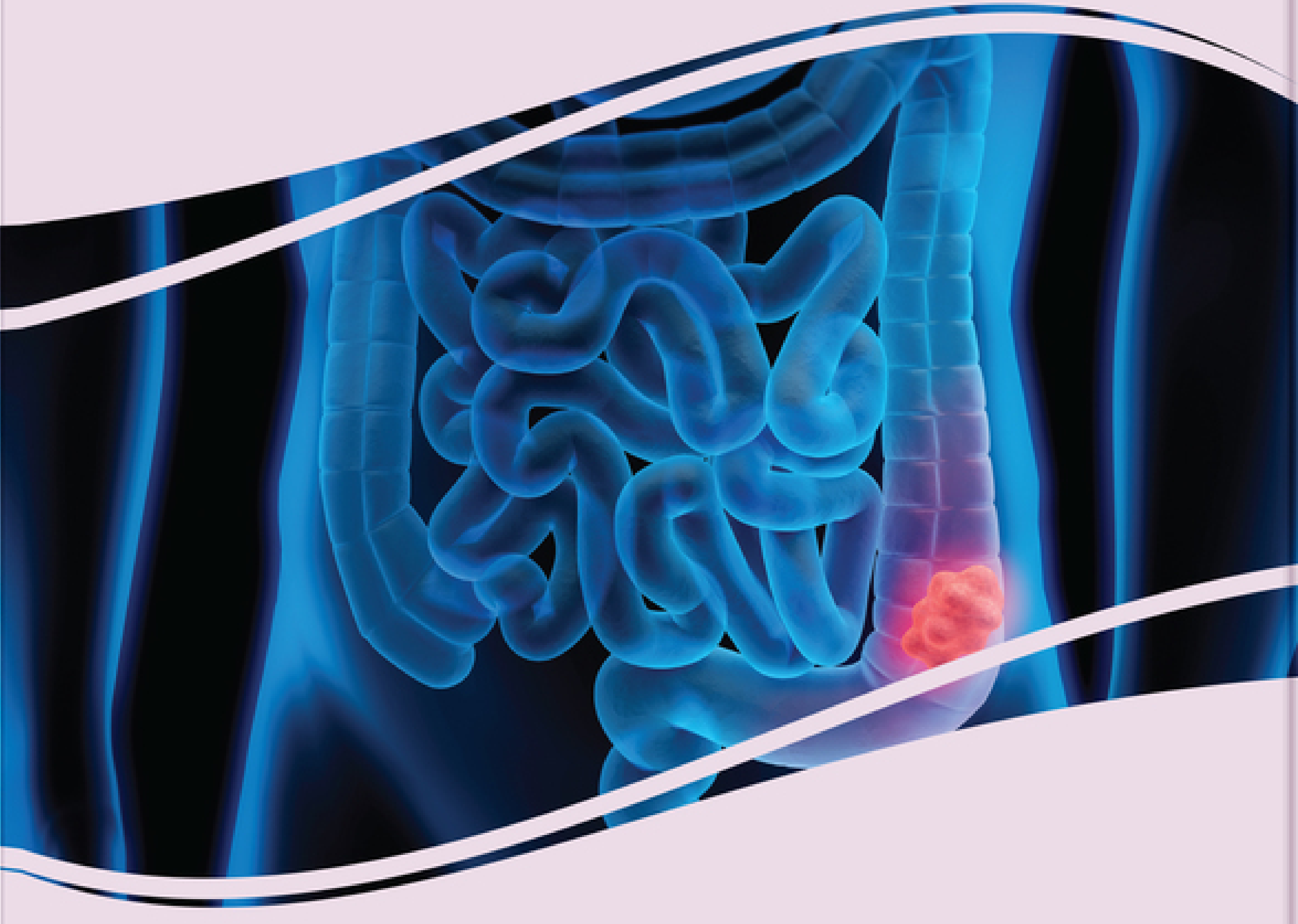
Premier Reference Source

Machine Learning in Cancer Research With Applications in Colon Cancer and Big Data Analysis



Zhongyu Lu, Qiang Xu, Murad Al-Rajab, and Lamogha Chiazor

IGI Global
PUBLISHER of TIMELY KNOWLEDGE

Machine Learning in Cancer Research With Applications in Colon Cancer and Big Data Analysis

Zhongyu Lu
University of Huddersfield, UK

Qiang Xu
University of Huddersfield, UK

Murad Al-Rajab
University of Huddersfield, UK & Abu Dhabi University, UAE

Lamogha Chiazor
University of Huddersfield, UK

A volume in the Advances in Medical Technologies and Clinical Practice (AMTCP) Book Series

IGI Global
PUBLISHER of TIMELY KNOWLEDGE

Published in the United States of America by
IGI Global
Medical Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA, USA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Names: Lu, Zhongyu, 1955- author. | Xu, Qiang, 1963- author. | Al-Rajab, Murad, 1983- author. | Chiazor, Lamogha, 1989- author.
Title: Machine learning in cancer research with applications in colon cancer and big data analysis / by Zhongyu Joan Lu and Qiang Xu, Murad Al-Rajab, Lamogha Chiazor.
Description: Hershey, PA : Medical Information Science Reference, [2021] | Includes bibliographical references and index. | Summary: "This book presents a general picture for the latest research output and the state of the art technology in the bioinformatics in cancer research, and big data, machine learning theory and technologies in the various applications"-- Provided by publisher.
Identifiers: LCCN 2020042092 (print) | LCCN 2020042093 (ebook) | ISBN 9781799873167 (hardcover) | ISBN 9781799873174 (ebook)
Subjects: LCSH: Cancer--Research. | Colon (Anatomy)--Cancer--Research. | Cancer--Diagnosis. | Bioinformatics. | Big data. | Problem solving--Data processing.
Classification: LCC RC267 .L8 2021 (print) | LCC RC267 (ebook) | DDC 616.99/40072--dc23
LC record available at https://lccn.loc.gov/2020042092
LC ebook record available at https://lccn.loc.gov/2020042093

This book is published in the IGI Global book series Advances in Medical Technologies and Clinical Practice (AMTCP) (ISSN: 2327-9354; eISSN: 2327-9370)

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Advances in Medical Technologies and Clinical Practice (AMTCP) Book Series

Srikanta Patnaik
SOA University, India
Priti Das
S.C.B. Medical College, India

ISSN:2327-9354
EISSN:2327-9370

Mission

Medical technological innovation continues to provide avenues of research for faster and safer diagnosis and treatments for patients. Practitioners must stay up to date with these latest advancements to provide the best care for nursing and clinical practices.

The **Advances in Medical Technologies and Clinical Practice (AMTCP) Book Series** brings together the most recent research on the latest technology used in areas of nursing informatics, clinical technology, biomedicine, diagnostic technologies, and more. Researchers, students, and practitioners in this field will benefit from this fundamental coverage on the use of technology in clinical practices.

Coverage

- Patient-Centered Care
- Biomechanics
- Biomedical Applications
- Neural Engineering
- Diagnostic Technologies
- Nursing Informatics
- Clinical Data Mining
- Medical Imaging
- Medical Informatics
- Biometrics

IGI Global is currently accepting manuscripts for publication within this series. To submit a proposal for a volume in this series, please contact our Acquisition Editors at acquisitions@igi-global.com or visit: https://www.igi-global.com/publish/.

Table of Contents

Preface

This book focuses on the chapters implemented with artificial intelligence and machine learning approaches and case studies to solve the predictive issues in colon cancer research. The book explains the concepts and techniques to run tasks in an automated manner, to improve a better accuracy in comparison with previous studies. So the proposed book will cover up the processes of research design, development and outcome analytics.

This book also presents the machine learning applied in the big data analysis. The machine learning or deep learning algorithms have been deployed in a wide range of applications. There are numerous benefits of big data, which have been discussed over the years in a wide range of applications including: increased efficiency, better and improved services in different sectors e.g. healthcare, e-commerce, etc. This book demonstrates a number of algorithms developed to evaluate big data approaches.

The book may benefit scientists, researchers and learners in the area of data science and healthcare practitioners, because there is always a challenge looking for and selecting the best performing ML algorithm(s) to use for a dataset in a short period of time. An automation of computation in machine learning could be a solution in next generation of big data science technology. Furthermore, Cancer kills millions of people worldwide each year. It is a growing problem and is the foremost cause of death worldwide. The numbers of people battling cancer is growing rapidly, owing to different reasons, such as lifestyle. Clinically, determining the cause of cancer is very challenging and often inaccurate. Increasing necessity to incorporate efficient and accurate algorithms to detect cancer cases, such as colon cancer, could be beneficial to the scientists in both sectors, i.e. computer science and healthcare, as well as a long term benefit could be extended to the doctors, patients and clinic practitioners, and other stakeholders.

THE TARGET AUDIENCE

The audience of this book could be cross sector readers, such as university postgraduates, final year undergraduates, doctors, researchers in the relevant subjects, such as computer scientists, healthcare practitioners, medical researchers and data workers/scientists.

THE IMPORTANCE OF EACH OF THE CHAPTER

This book is organised in 13 chapters that address different issues in the subject area. The importance of each chapter is highlighted as follows.

Chapter 1 introduces the importance of Information Science into Colon Cancer Research. According to the American Cancer Association, in the United States for 2018 include 97,220 new cases of colon cancer (CC). The research into this topic area is an immediate need for saving huge number of human life and improving people's living standards.

Chapter 2 gives an overview of bioinformatics in the cancer research. The work has described the historic development of topic area and state of the art technology in the research regime, such as biomedical data analysis algorithms, microarray dataset technology, colon cancer gene selection and classification algorithms, machine learning classification algorithms, discussion of accuracy and efficiency in previous studies.

Chapter 3 describes the detailed approaches in the underpinned theory and methods in machine learning. The algorisms involved are genetic algorisms (GA), and classification algorithms. The implementation methods and technology, such as support vector machine (SVM), Naive Bayers (NB), Decision Tree (DT) and K-Nearest Neighbour, are discussed as well.

Chapter 4 defines the research design and procedures for the investigation conducted. The different experimentations are carried out with two case studies. The chapter describes the data preparation, two stage hybrid multi-filter feature selection method, and two case studies for the further comparison and evaluation of research outcome.

Chapter 5 presents the findings for the research conducted. The significant discovery has been highlighted within the context, especially for the different case studies.

Chapter 6 analyses and discusses the research findings and evaluations with existing research done by the academic communities, especially for the classification accuracy and efficiency of two case studies.

Chapter 7 provides the final remarks for the research with advanced machine learning methods in colon-cancer analysis. The chapter discusses the achievements and limitations in the current investigation and indicate the future research directions in the subject area.

Chapter 8 provides background information for the research of big data and machine learning; highlights into the motivations, problems need to be resolved and then discusses the aims and contributions of the research conducted.

Chapter 9 discusses what is already known in the area of this research. Touching particularly on the key concepts, theories, and factors and how they are relevant to this research. Some inconsistencies, limitations and problem in existing literatures are discussed. Discussions on why some of these limitations and inconsistencies occur, how the knowledge relates to this research, as well as issues still yet to study effectively is carried out. Finally, it sets the basis for what contributions this research makes and who will benefit from such a study.

Chapter 10 gives a detailed and logically ordered plan of the approach, techniques, procedures and steps followed to achieve the research aims and objectives, and detailed description of methods employed, experiments conducted, algorithms developed and the knowledge gained from the research.

Chapter 11 presents the development process for hybrid-autoML system from the model design, to algorithms development. The tools, e.g. Weka, used to test and evaluate the system with case studies.

Chapter 12 discuss the evaluation process for the system developed with five cases. The comparison studies are also carried out and discussed with research evidence from this investigation.

Chapter 13 provides the final remarks and further directions for the Hybrid-AutoML System. Meanwhile, the limitations and achievements are also discussed.

CONCLUSION

This book presents a general picture for the latest research output and the state of the art technology in the bioinformatics in cancer research, and big data, machine learning theory and technologies in the various applications, in particular, it highlights that:

1. Colon cancer is critical and easy to misdiagnose. The new data analysing methods and technologies to enhance cancer detection and tumour classification is immediate needed to save human life and improve the quality of living style.
2. Big data science is another challenge for researchers working towards high automation of advanced system to achieve a high accuracy and efficiency for the real world applications.
3. The advanced machine learning theory and technology makes significant contributions to the above important areas.

Zhongyu Lu
University of Huddersfield, UK

Chapter 1 Importance of Information Working With Colon Cancer Research

ABSTRACT

Modern science helps us to understand the changing world around us, across fields such as biology, computer science, mathematics, statistics, chemistry, computational biology, biotechnology, biochemistry, and many others. An important branch of science that has had a large impact on the medical field is bioinformatics. This chapter introduces the importance of information science into colon cancer research. According to the American Cancer Association, in the United States in 2018, 97,220 new cases of colon cancer (CC) were identified. The research into this topic area is an immediate need to save many lives and improve people's living standards.

1. INTRODUCTION

Modern science helps us to understand the changing world around us, across fields such as biology, computer science, mathematics, statistics, chemistry, computational biology, biotechnology, biochemistry, and many others. An important branch of science that has had a large impact on the medical field is bioinformatics. This field has recently attracted the attention of academia and medical professionals (Chavan, 2008; Poe et al., 2009; Simon, 2005; Umarji et al., 2009). Bioinformatics, a new science, aims at assisting the medical field when choosing correct treatments, detecting diseases, and supporting drug development. A particularly challenging disease is cancer, a genomic disease in which human cells lose their ability to follow the chronological phases of a normal cell cycle. Thus, those cells lose their regulation process, divide uncontrollably, and alter their chromosomal constituency, forming cancer cells (Al-Rajab & Lu, 2012).

Computerised models can employ gene expression data and related information to estimate the clinical state of a patient or the amount of cancer tissues or cells inside the body (Al-Rajab & Lu, 2012; Ardekani, Aslani, & Lakpour, 2007, Guo et al, 2021). Bioinformatics has shown positive impact during

DOI: 10.4018/978-1-7998-7316-7.ch001

the treatment of cancer as professionals continually strive to control tumour growth in ways that were impossible in the past. Moreover, many studies have demonstrated that cancer-cell gene expression is essential to effective remediation (Fenstermacher, 2005; Kihara, Yang, & Hawkins, 2006). Cancer studies also employ bioinformatics to record cancer-cell expression from datasets and to identify tumour and drug responses (Kihara, Yang, & Hawkins, 2006). Until recently, bioinformatics studies have succeeded in identifying two types of cancer: ovarian and breast. The future of bioinformatics will certainly contribute to the therapy of other types of cancer (Ardekani, Aslani, & Lakpour, 2007). Additionally, with bioinformatics, it is possible for therapists to analyse immune system responses to permit a better understanding of the alterations in both controlled and uncontrolled tumours and to provide patients with better treatment. Moreover, bioinformatics, with the help of mathematical models, has demonstrated success in describing the effects of radiation therapy and chemotherapy on the human body.

Cancer is classified as a critical disease in modern civilisation that causes great suffering. There are many carcinogens and different causes of cancer (e.g., radiation, smoking, artificial chemicals, microbes, dirty water, polluted air), all of which accelerate changes in cells. Certainly, there are many causes yet undiscovered (Shah & Kusiak, 2007). However, traditional methods of diagnostics have depended mainly on the clinical appearance and the morphologic structure of cancer. However, these attributes have limited influence because cancer results from many different factors. Moreover, there is an overwhelming need to find informative genes from information in huge datasets, wherein researchers eliminate unrelated genes, reduce noise and complexity and present opportunities for disease detection (Shah & Kusiak, 2007). An informative gene is suitable and beneficial for classifying cancer (Mohamad et al., 2008). The most challenging task for cancer diagnosis is recognising relevant gene expressions from massive datasets, which can help determine the disease phenotype (Liu, Liu, & Zhang, 2010). The insertion and evolution of array diagnostic technologies aid the early prediction and accurate diagnosis of cancer (Al-Rajab & Lu, 2016). For example, the instantaneous retrieval and analysis of thousands of gene expression levels instantaneously and would greatly benefit cancer classification (Park & Cho, 2003). Cancer classification plays a vital role in enhancing patient health care and can greatly improve quality of life. Classification, therefore, is crucial for drug assignment (Liu, Liu, & Zhang, 2010). Cancer classification and detection uses a microarray dataset to discriminate samples within a given model (Mohamad et al., 2007). The aim of using microarray data in cancer classification is to distinguish body cells into relevant classes of observations, such as normal versus cancerous (Al-Rajab & Lu, 2016; Osareh & Shadgar, 2010).

2. SIGNIFICANCE OF THE RESEARCH

Colon cancer is the most common tumour-producing disease in the world. It is unfortunately well known that many hospitals and treatment centres lack the facilities and equipment to predict the disease. Additionally, the larger the queue of people waiting to be treated in a given region lacking requisite facilities, the larger the probability of missed diagnoses. For this reason, there is a great need for new methods and technologies to enhance cancer detection and tumour classification.

The main problems with microarray genetic data are the noisy nature of genes, the high redundancy of data and the irrelevant information that negatively affects classification accuracy. Because only a few genes are important for classification (Al-Rajab, Lu, & Xu, 2017; Nurminen, 2003), the need for algorithmic cancer identification has become critical. For practicality, methods and algorithms must be rapid, reliable, accurate, easily implemented, tested and maintained (Solorio-Fernández et al., 2012).

Thus, it is desirable to improve the research in these areas. Many researchers have tackled the selection and classification problem by examining machine-learning (ML) techniques for cancer classification (Yang & Zhang, 2007). Many studies have shown that different cancers result from changes in gene expression. Several of these studies implemented data size reducing gene selection methods to eliminate uninformative or irrelevant genes from datasets. Thus, ML has recently been applied to gene selection and cancer classification algorithms (Al-Rajab, Lu, & Xu, 2017). Although hopeful outcomes have been reported in the literatures, they are merely shown in the classification accuracy. There seems to be a lack of effort with measuring algorithm efficiency (i.e., time analysis) between the various gene selection cancer classifying methods (Al-Rajab, Lu, & Xu, 2017). More details about these shortcomings are presented in the literature review (Section 2.10). The significance of this research is described as follows.

- This study incorporates the simultaneous identification and classification of colon cancer tissues using accurate and efficient algorithms. It is critical that we expertly and accurately classify them fast and efficiently. The faster the diagnosis, the greater the probability of survival, including improved drug assignment.
- This study presents an improved methodology to enhance colon cancer cell identification and selection. The method comprises a two-stage multifilter selection process that guarantees the selection of more informative genes. The key advantage of the proposed method is its substantial reduction of features, which results in better classification performance.

3. RESEARCH QUESTIONS

Classifying genes is a challenging task because of its nature (Al Snousy et al., 2011). The first challenge relates to the availability of only small patient sample datasets, wherein the number of patients is very often less than 200, compared to multidimensional genetic datasets, in which the number of genes can be thousands or tens of thousands for each tuple. A second challenge reflects the fact that not all the genes are required for disease detection; only a few are relevant. A third challenge relates to the technological and biological noisy nature of the dataset. The final challenge relates to the application field, where classification accuracy is not the only goal; gene relevancy is also very important. According to Al Snousy et al. (2011) and Kwon and Sim (2013), there are many classification algorithms that can enhance data prediction. However, there is no single best algorithm for all cases. However, it remains possible to develop a powerful algorithm for an explicit data need in a specific area, because classification accuracy depends on the classification methodology, the gene or feature selection and the nature of the dataset. Recently, gene selection procedures and cancer classification methods using ML algorithms have been employed in cancer therapy. Whereas several algorithms have shown improved or better classification results compared to others, there exists no thorough research comparing the time efficiency between the diverse selection methods and classifiers. Thus, this study explores and examines the following research questions.

- What are the main algorithms/methods implemented and applied for colon cancer in terms of gene selection and classification?
- Which of these algorithms ensures high-performance results?
- Which of these algorithms achieves efficient results with better time analysis?

Does the combination or the hybridisation of multifilter gene selection algorithms achieve better classification performance results?

4. AIMS AND OBJECTIVES

The aims of this research study are as follows.

(1) A method of examining the detection accuracy and efficiency (i.e., time complexity) of high-performance gene selection and cancer classification algorithms for colon cancer therapy is sought, and (2) a proposed model for a two-stage hybrid multifilter feature selection system to augment the predictive outcome of ML is desired. To achieve these aims, the following objectives are specified.

- Examine and determine common feature selection algorithms to demonstrate better and more efficient performance with colon cancer data subset selection and gene reduction.
- Examine and determine common classification high-performance algorithms that demonstrate accuracy and efficiency (i.e., time complexity analysis) while examining the colon cancer dataset.
- Propose a model of applying a two-stage hybrid multifilter gene selection model for better colon cancer classification and cell detection.

5. SUMMARY OF THE CHAPTER

- Demonstrate a systematic characterisation method of examining accuracy and efficiency using time complexity analysis of high-performance gene selection and cancer classification algorithms for colon cancer (Chapter 3). It provides an alternative performance-based comparison of algorithms, rather than using classification accuracy on its own (Chapter 2). Although, there existing studies which evaluate the accuracy of cancer types, none of them had contributed to measure the efficiency using the time complexity analysis. Whereas, comparative results that depend on classification accuracy alone show limitations, as stated in Chapter 2.
- Propose a two-stage hybrid multifilter feature selection method to augment the prediction outcome of ML algorithms to better classify colon cancer datasets (Chapter 3). The proposed method demonstrates that the hybrid genetic algorithm (GA) and information gain (IG), combined with minimum redundancy maximum relevance (mRMR) ranking algorithms, classified by support vector machines (SVM), leads to better classification accuracy compared to previous works (Chapter 6).

REFERENCES

Al-Rajab, M., & Lu, J. (2012). *Bioinformatics: An Overview for Cancer Research*. Paper presented at the 2012 World Congress in Computer Science, Computer Engineering & Applied Computing, Las Vegas, NV.

Al-Rajab, M., Lu, J., & Xu, Q. i. (2017). Examining Applying High Performance Genetic Data Feature Selection and Classification Algorithms for Colon Cancer Diagnosis. *Computer Methods and Programs in Biomedicine*, *146*, 11–24. doi:10.1016/j.cmpb.2017.05.001 PMID:28688481

Al-Rajab, M. M., & Lu, J. (2016). A Study on the Most Common Algorithms Implemented for Cancer Gene Search and Classifications. *International Journal of Data Mining and Bioinformatics*, *14*(2), 159–176. doi:10.1504/IJDMB.2016.074685

Al Snousy, M. B., El-Deeb, H. M., Badran, K., & Al Khlil, I. A. (2011). Suite of Decision Tree-Based Classification Algorithms on Cancer Gene Expression Data. *Egyptian Informatics Journal*, *12*(2), 73–82. doi:10.1016/j.eij.2011.04.003

Ardekani, A. M., Aslani, F., & Lakpour, N. (2007). Application of Genomics and Proteomics Technologies to Early Diagnosis of Reproductive Organ Cancers. *Journal of Reproduction & Infertility*, *8*(3), 259–278.

Chavan, P. R. (2008). *Application of Bioinformatics in the Field of Cancer Research.* Paper presented at the 11th Workshop on Medical Informatics & CME on Biomedical Communication.

Fenstermacher, D. (2005). Introduction to Bioinformatics: Research Articles. *Journal of the American Society for Information Science and Technology - Bioinformatics, 56*(5), 440-446. doi:10.1002/asi.v56:5

Guo, C., Wang, J., Yongming, W., Qu, X., Shi, Z., Meng, Y., Qiu, J., & Hua, K. Novel artificial intelligence machine learning approaches to precisely predict survival and site-specific recurrence in cervical cancer: A multi-institutional study, Translational Oncology 14 (2021) 101032, ISSN: 1936-5233

Kihara, D., Yang, Y. D., & Hawkins, T. (2006). Bioinformatics Resources for Cancer Research with an Emphasis on Gene Function and Structure Prediction Tools. *Cancer Informatics*, *2*, 25–35. doi:10.1177/117693510600200020 PMID:19458756

Kwon, O., & Sim, J. M. (2013). Effects of Data Set Features on the Performances of Classification Algorithms. *Expert Systems with Applications*, *40*(5), 1847–1857. doi:10.1016/j.eswa.2012.09.017

Liu, H., Liu, L., & Zhang, H. (2010). Ensemble Gene Selection for Cancer Classification. *Pattern Recognition*, *43*(8), 2763–2772. doi:10.1016/j.patcog.2010.02.008

Mohamad, M. S., Omatu, S., Deris, S., & Hashim, S. Z. M. (2007). A Model for Gene Selection and Classification of Gene Expression Data. *Artificial Life and Robotics*, *11*(2), 219–222. doi:10.100710015-007-0432-1

Mohamad, M. S., Omatu, S., Yoshioka, M., & Deris, S. (2008). *An Approach Using Hybrid Methods to Select Informative Genes from Microarray Data for Cancer Classification*. Paper presented at the Second Asia International Conference on Modeling & Simulation (AICMS 08), Kuala Lumpur, Malaysia. 10.1109/AMS.2008.71

Nurminen, J. K. (2003). Using Software Complexity Measures to Analyze Algorithms—An Experiment with the Shortest-Paths Algorithms. *Computers & Operations Research*, *30*(8), 1121–1134. doi:10.1016/S0305-0548(02)00060-6

Osareh, A., & Shadgar, B. (2010). *Microarray Data Analysis for Cancer Classification*. Paper presented at the 5th International Symposium on Health Informatics and Bioinformatics (HIBIT), Antalya, Turkey. 10.1109/HIBIT.2010.5478893

Park, C., & Cho, S.-B. (2003). *Evolutionary Ensemble Classifier for Lymphoma and Colon Cancer Classification*. Paper presented at the 2003 Congress on Evolutionary Computation (CEC'03), Canberra, Australia. 10.1109/CEC.2003.1299385

Poe, D., Venkatraman, N., Hansen, C., & Singh, G. (2009). Component-Based Approach for Educating Students in Bioinformatics. *IEEE Transactions on Education*, *52*(1), 1–9. doi:10.1109/TE.2007.914943

Shah, S., & Kusiak, A. (2007). Cancer Gene Search with Data-Mining and Genetic Algorithms. *Computers in Biology and Medicine*, *37*(2), 251–261. doi:10.1016/j.compbiomed.2006.01.007 PMID:16616736

Simon, R. (2005). Bioinformatics in Cancer Therapeutics-Hype or Hope? *Nature Clinical Practice. Oncology*, *2*(5), 223–224. doi:10.1038/ncponc0176 PMID:16264939

Solorio-Fernández, S., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & Zhang, Y.-Q. (2012). *Hybrid Feature Selection Method for Biomedical Datasets*. Paper presented at the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), San Diego, CA. 10.1109/CIBCB.2012.6217224

Umarji, M., Seaman, C., Koru, A. G., & Liu, H. (2009). *Software Engineering Education for Bioinformatics*. Paper presented at the 22nd Conference on Software Engineering Education and Training (CSEET'09), Hyderabad, India. 10.1109/CSEET.2009.44

Yang, P., & Zhang, Z. (2007). Hybrid Methods to Select Informative Gene Sets in Microarray Data Classification. *AI 2007. Advances in Artificial Intelligence*, *4830*, 810–814. doi:10.1007/978-3-540-76928-6_97

Chapter 2
An Overview on Bioinformatics

ABSTRACT

This chapter presents a thorough background and deep literature review of the current topic of study. It also presents and defines the key concepts utilised throughout this investigation. It consists of ten sections: (1) a background on bioinformatics, (2) a discussion of colon cancer, (3) an overview of the microarray technology that is used to extract the dataset, (4) an overview of the colon cancer dataset, (5) a review of the most prevalent algorithms employed for gene selection and cancer classification, (6) a presentation of related works from the literature, (7) identification of feature selection approaches and procedures, (8) an investigation of the ML concept, (9) a review of algorithm efficiency and time complexity analysis, and (10) identification of current problems in the research area.

1. INTRODUCTION

This chapter presents a thorough background and deep literature review of the current topic of study. It also presents and defines the key concepts utilised throughout this investigation. It consists of ten sections: (1) a background on bioinformatics; (2) a discussion of colon cancer; (3) an overview of the microarray technology that is used to extract the dataset; (4) an overview of the colon cancer dataset; (5) a review of the most prevalent algorithms employed for gene selection and cancer classification; (6) a presentation of related works from the literature; (7) identification of feature selection approaches and procedures; (8) an investigation of the ML concept; (9) a review of algorithm efficiency and time complexity analysis; and (10) identification of current problems in the research area.

2. BIOINFORMATICS: AN OVERVIEW OF CANCER RESEARCH

Bioinformatics is the integration of the fields of biology, computer science, statistics, and mathematics (Al-Rajab & Lu, 2012), with each field playing a significant role in gathering, forming, analysing, and digitising genetic data. Moreover, it aids the efficient categorisation and storage of data (Al-Rajab & Lu, 2012; Bayat, 2002; Cohen, 2005; Jawdat, 2006; Jena et al., 2009; Ng & Wong, 2004).

DOI: 10.4018/978-1-7998-7316-7.ch002

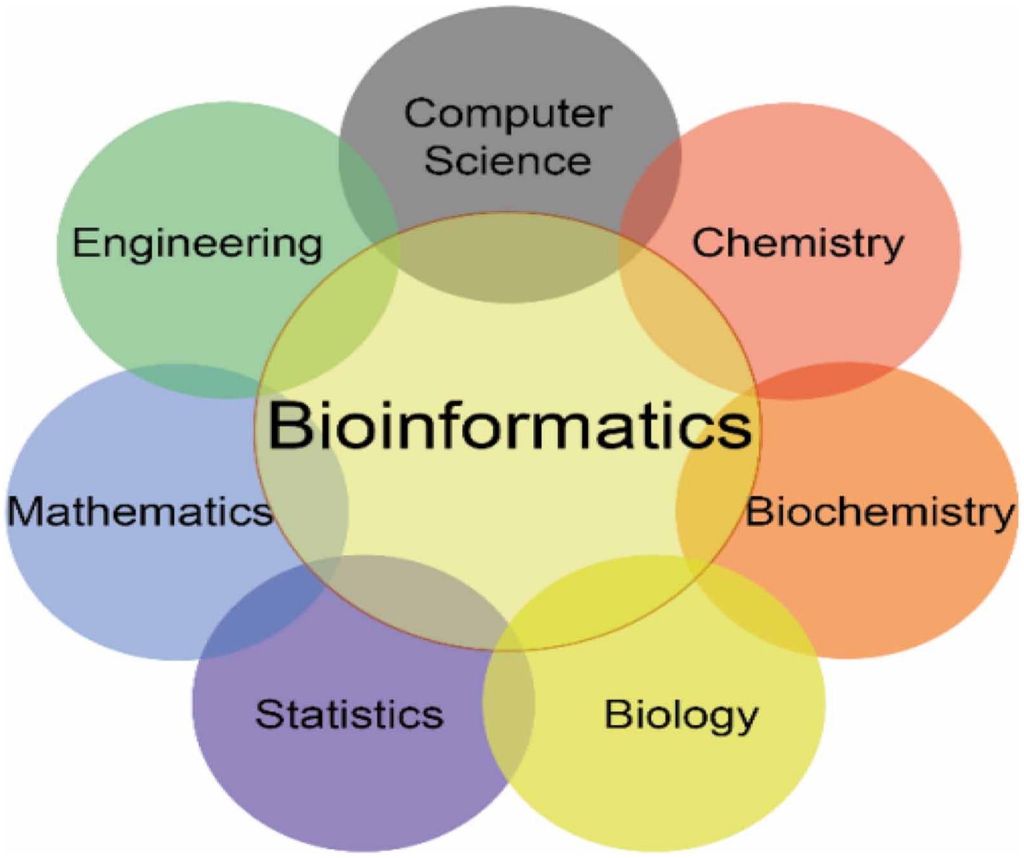
Bioinformatics was first introduced in 1979 by Paulien Hogeweg to study the informatic procedures of biological systems (Raza, 2012). This section elaborates bioinformatics using a variety of scientific papers to establish the basis for the current research, to identify core bioinformatics applications, to present the data structure and the central databases employed, to provide an overview of the most popular algorithms applied to this field, and to advocate for the employment of bioinformatics in the area of cancer research.

2.1 Background of Methodologies

Several scientific papers, books, articles, and websites provide information regarding bioinformatics. However, these resources do not present a unified, official, or integrated description for bioinformatics as a science. Most works simply present basic descriptions. Therefore, a deep search was conducted to gather as much knowledge as possible to understand the field and connect its significance to cancer research.

Jawdat (2006) described bioinformatics as a process using specific methods, algorithms, and computer software to store and interpret biological data. It is the design, implementation, and employment of computer tools to produce, store, interpret, access, and analyse molecular biology data. (Raut, Sathe, & Raut, 2010) stated that bioinformatics is fundamentally the study to design, arrange, grasp and explore useful information linked to a large amount biotic data. The term, 'bio' signifies the molecular biology aspect, and 'informatics' reflects the information technology used to control, interpret, and employ vast amounts of genetic data. In (Ng & Wong, 2004), the authors claimed that bioinformatics manages, organises and analyses genetic data, which is the raw, basic essence of any organism. Bioinformatics connects computer science, biology and mathematics to address the many computational challenges of modern medical research. Chavan (2008) maintained that genetic data contains a wide range of information concerning genetic sequences regulating diversity, evolutionary changes and alterations of proteins. Bioinformatics is a product of the need to systematically filter data for classification and indexing. Thus, it can be defined as the field that combines a diversity of sciences for this purpose (Al-Rajab & Lu, 2012). It is the discipline of controlling, interpreting and analysing a massive amount of genetic data using progressive computing methods and techniques. Furthermore, other authors claimed that the difference between bioinformatics and computational biology cannot be simply deduced (Ackovska & Madevska-Bogdanova, 2005). Both topics involve several scientific disciplines, including molecular biology, computer science, mathematics, statistics, physics, biochemistry and genetics (Al-Rajab & Lu, 2012, Guo et al, 2021). Zadeh (2006) defined bioinformatics as a modern field developed from the domains of biology, computer science and biochemistry. Bioinformatics is a multidisciplinary study and a fast-growing topic evolved from the domains of biology, computer science and chemistry. Additionally, Kasabov (2004) claimed that bioinformatics included the development of information science applications needed to analyse, demonstrate and discover knowledge of biological activities in living organisms. Moreover, in (Fulekar & Sharma, 2008), the authors focused on the mixture of information technology and biology, insisting that bioinformatics concentrated on the molecular cell level of biotechnology. Fenstermacher (2005) defined bioinformatics as a multidimensional topic joining many specialised disciplines, such as computational biology, mathematics, statistics, molecular chemistry and biology. Additionally, Nair (2007) stated that bioinformatics answers biologists' questions about the ambiguities of life via the application of computer science and associated technologies. Other authors stated that bioinformatics is a modern and fast-developing topic that combines molecular biology, biochemistry, artificial intelligence (AI), databases, pattern recognition and computer science algorithms (Al-Rajab & Lu, 2012; Doom et al., 2003). Finally, in (Jena et al., 2009) the authors defined bioinformatics as the employment of computer technology to address biological information management. Figure 1 summarises the variety of fields that provide meaning for bioinformatics. For the purposes of this research, bioinformatics comprises four core elements: databases, computational tools, algorithms, and software (Al-Rajab & Lu, 2012). Biologists and other concerned experts should be mindful of the differences between bioinformatics and computational biology. The latter is concerned with using computers to understand biological hypotheses (Al-Rajab & Lu, 2012; Fenstermacher, 2005, Nahum et al, 2019, SHukla, 2020), whereas bioinformatics is more concerned about the information itself Domokos (2008).

Figure 1. Bioinformatics multidisciplinary sciences



2.2 Aims of Bioinformatics

There exist five mains aims and objectives of bioinformatics, as discussed in (Al-Rajab & Lu, 2012; Jena et al., 2009).

1. To easily arrange biological data supporting biologists and other scientists to save and access related extant information.

2. To design, develop and implement software algorithms and applications to assist managing and analysing genetic data. entropy-based discretisation
3. To utilise genetic data in the interpretation, expression and analysis of biological results.
4. To assist the drug industry by aiding researchers' ability to recognise protein structures.
5. To assist medical fields in the recognition of genetic data structures to assist detection and diagnosis of diseases, including cancer.

2.3 Biological Data, Data Types and Databases

Biological genetic data is distinguished by its enormous volume. There exist four significant kinds of data produced and extracted via biological processes (Nair, 2007): deoxyribonucleic acid (DNA), ribonucleic acid (RNA), protein sequences and microarray images. The first three are classified as text data types, but the last one is a digital image data type. When various biological genetic data are produced, they are typically characterised using diverse data structures.

There are four types of biological data structures: (1) strings that signify DNA, RNA and protein sequences; (2) trees that signify protein structures; (3) graphs that characterise metabolic and signalling pathways and (4) long strings (e.g., words and phrases) that capture researchers' observations. Furthermore, biologists and researchers are also concerned with substrings, subtrees and subgraphs (Al-Rajab & Lu, 2012; Cohen, 2005).

Massive and composite volumes of genetic data are required to be stored, analysed, retrieved and manipulated in effective and efficient ways. Thus, bioinformatics databases have been established. These databases are categorised into microarray databases, protein structure databases, genome databases, sequence databases and others (Raut, Sathe, & Raut, 2010).

Sequence information about all organisms are represented by sequence databases (Al-Rajab & Lu, 2012). Examples of extant large databanks include DNA sequence databases (e.g., European Molecular Biology Laboratory, Bethesda, DNA Data Bank Japan, the National Centre for Biotechnology Information, GeneBank and the Protein Sequence Database at the Swiss Institute of Bioinformatics, Geneva (Swiss-Port)) (Al-Rajab & Lu, 2012; Jawdat, 2006). Microarray gene expressions are contained in microarray databases under various biological circumstances. Some database examples of this group are the Gene Expression Omnibus and the Array Express. Gene databases gather cells' DNA sequences. DrugBank is a cheminformatics database that contains drug entries (Al-Rajab & Lu, 2012).

2.4 Bioinformatics Common Algorithm Tendencies

This section explores popular algorithm tendencies utilised by bioinformaticians and other researchers. Some of the most common algorithmic objectives are listed below (Al-Rajab & Lu, 2012; Cohen, 2005; Jena et al., 2009).

1. Distinguishing specific paradigms within strings (e.g., genes).
2. Discovering similarities between strings (e.g., proteins of different organisms).
3. Classifying new data based on formerly interpreted clustered sets.
4. Understanding microarray data and equivalent path behaviours.
5. Discovering similarities between spatial structures (e.g., motifs).

6. Building phylogenetic trees that express the growth of living organisms with known DNA or proteins.

2.5 Applications of Bioinformatics in Cancer Research

Cancer is a genetic disease in which cells lose their regulator sequence and split uncontrollably. Thus, cancer cell chromosomes organise incorrectly or lose large genetic components.

Extensive efforts have been made in medical fields to reveal, identify and cure cancers. The Human Genome Project, completed in 2003, set the stage for bioinformatics being employed in cancer research and treatment (Al-Rajab & Lu, 2012). Recently bioinformatics research trials have been applied to cancer investigation and therapy (Simon, 2005). Researchers have rapidly implemented bioinformatic tools for cancer therapy. One of these measures utilises computer paradigms linked to biological genetic data to explore and investigate cancer-cell volumes inside the human body and the clinical states of patients (Goldin, 2006). Moreover, bioinformatics can access cancer-cell expression information in databases to help researchers investigate drug responses and determine tumour statuses (Al-Rajab & Lu, 2012; Kihara, Yang, & Hawkins, 2006).

Multiple data repositories and a variety of search engines, such as Google, have been utilised by researchers to extract biological data and to employ bioinformatics in cancer studies. Unfortunately, this is a 'stovepiping' procedure that limits access and restricts information-sharing among experts. Thus, bioinformatics databases require integration, because the unified data types are all vital factors for adopting bioinformatics in cancer therapy. Consequently, bioinformatics has been applied in several domains to investigate and solve related problems (Chavan, 2008; Nair, 2007), such as

- Predicting and analysing protein sequence data;
- Predicting and analysing DNA sequence data to detect genes;
- Predicting and analysing RNA sequence data;
- Interpreting images of gene expression;
- Examining genetic diseases, such as cancer and anaemia;
- Designing and enhancing drugs for better treatment and side-effect reduction.

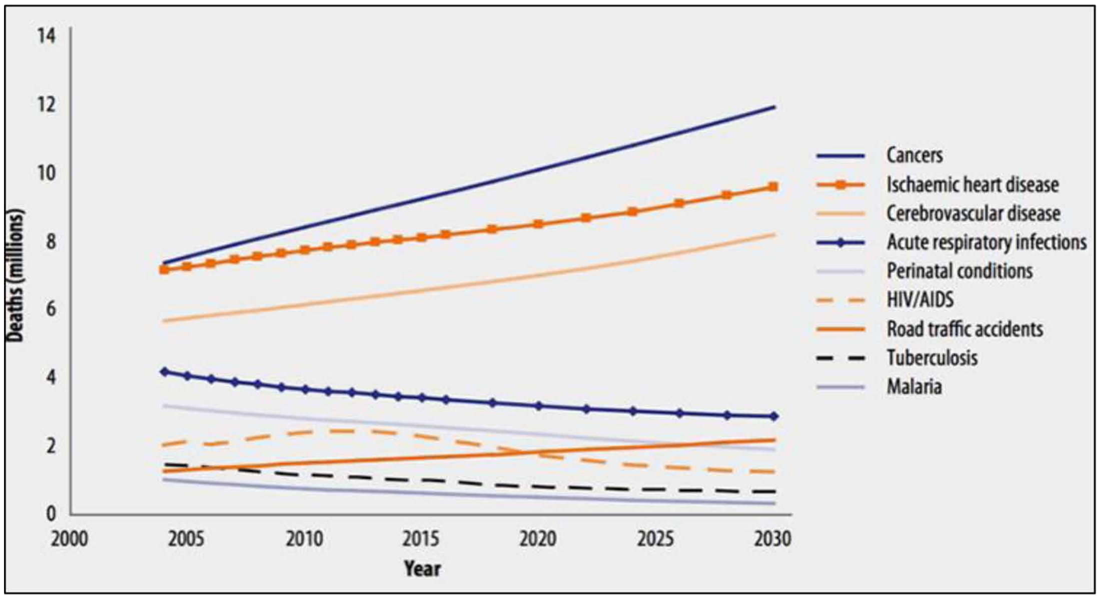
2.6 Colon Cancer

According to the World Health Organisation (WHO), cancer is the top cause of death worldwide, with around 14-million cancer cases each year. 8.2-million cases result in death within the same year as diagnosis (Stewart & Wild, 2014). Additionally, the WHO has predicted that cancer will continue to be the top cause of death through 2030. See Figure 2. Cancer presents primarily by a rotatory alteration of regular cells having the ability to damage DNA, spoiling cell replication and resulting in malignant tumours (i.e., cancers).

Owing to fast and dramatic advances in medical research, intensive efforts have been applied to discover possible methods of diagnosing and treating cancer. As mentioned, the Human Genome Project enabled bioinformatics to be employed in cancer treatment (Al-Rajab & Lu, 2012). However, researchers still face many difficulties in choosing the best methods.

Cancer is an enemy that kills indiscriminately. It is one of the most serious health problems in human history (Akbar, Gopi, & Babu, 2015). However, there are different types of cancer, and they all start with an out-of-control evolution of cells. Regular cells of the body can grow, divide and die in a regulated way. Throughout the early years of a human's life, normal body cells divide very fast until adulthood. Later, most of these cells divide only to substitute damaged or dying cells. Cancer cells are dissimilar from regular cells in this way. Instead of dividing to replace old cells, they continue reproducing with new irregular cells, as shown in Figure 3 and 4.

Figure 2. Projected deaths worldwide by selected causes, 2005-2030
Source [Adapted from]: World Health Organisation (WHO, 2008) 'The global burden of disease: 2004 update, Health statistics and information systems' [Image]. Retrieved from https://www.who.int/healthinfo/global_burden_disease/GBD_report_2004update_full.pdf?ua=1



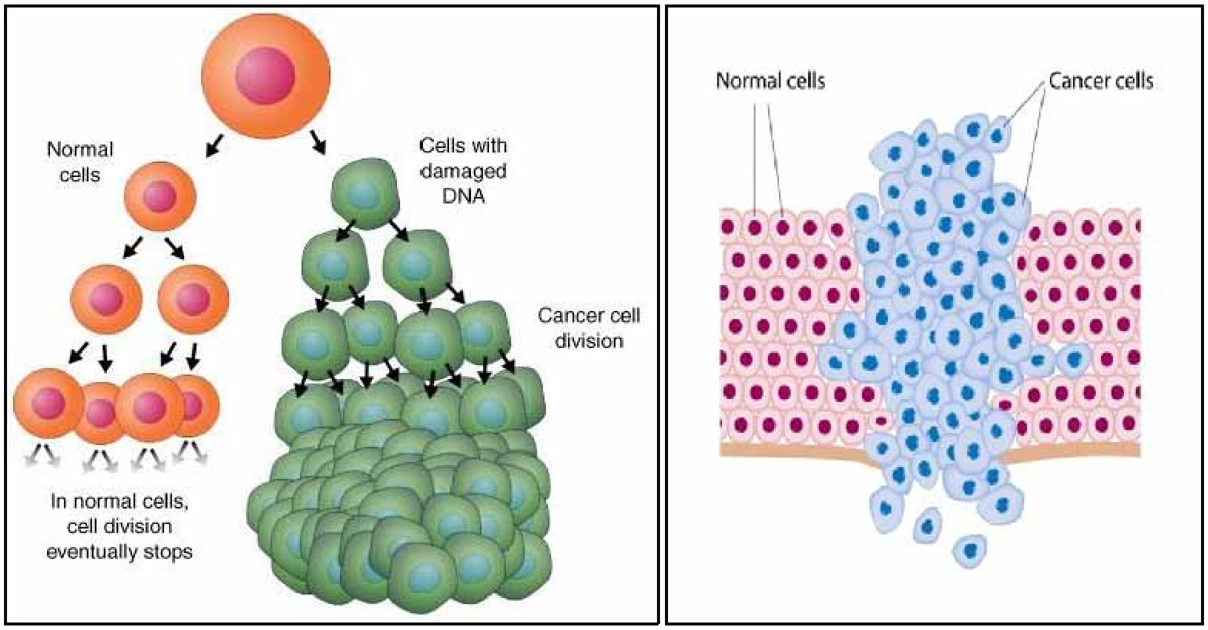
Damaged DNA causes cancer cells to develop. DNA is found in each cell and controls all cell activities. Whenever DNA is damaged, the body tries to rebuild it. However, cancer cells cannot be repaired or rebuilt. Thus, damaged DNA can be inherited, leading to congenital cancers.

About half a million people die every year because of colon cancer (Rathore et al., 2013). Colon cancer is ranked the fourth major cause of mortality in the world (Stewart & Wild, 2014). The colon is an important part of the large intestine, as shown in Figure 5. A colon tumour is formed when an abnormal growth of cells is found on the inner wall of the colon. Non-cancerous or benign tumours of the colon are called polyps (a mushroom-like growth). However, malignant tumours of the same type are cancerous.

Normally, benign polyps do not divide and spread to nearby organs and can be safely removed. However, if these polyps are not removed, they may become cancerous, developing the ability to invade and damage adjacent cells, tissue and organs. Once a polyp turns malignant, it grows rapidly and enlarges very quickly. It can even block the colon and breach the colon wall. Additionally, it may spread to other organs. There are many causes of colon cancer. Some are genetic and others lifestyle-based (e.g., age, red meat, high fat, smoking and low intake of fruits) (Akbar, Gopi, & Babu, 2015).

Figure 3. Colon cancer cells
Source [Adapted from]: (Kaulitzki, 2018) Kaulitzki, S. (2018). '3-D rendered illustration of a colon tumour' [Image]. Retrieved from https://www.shutterstock.com/image-illustration/3d-rendered-illustration-colon-tumor-130089656?src=EDizE28-03Vl_dYxHGwmBQ-2-30

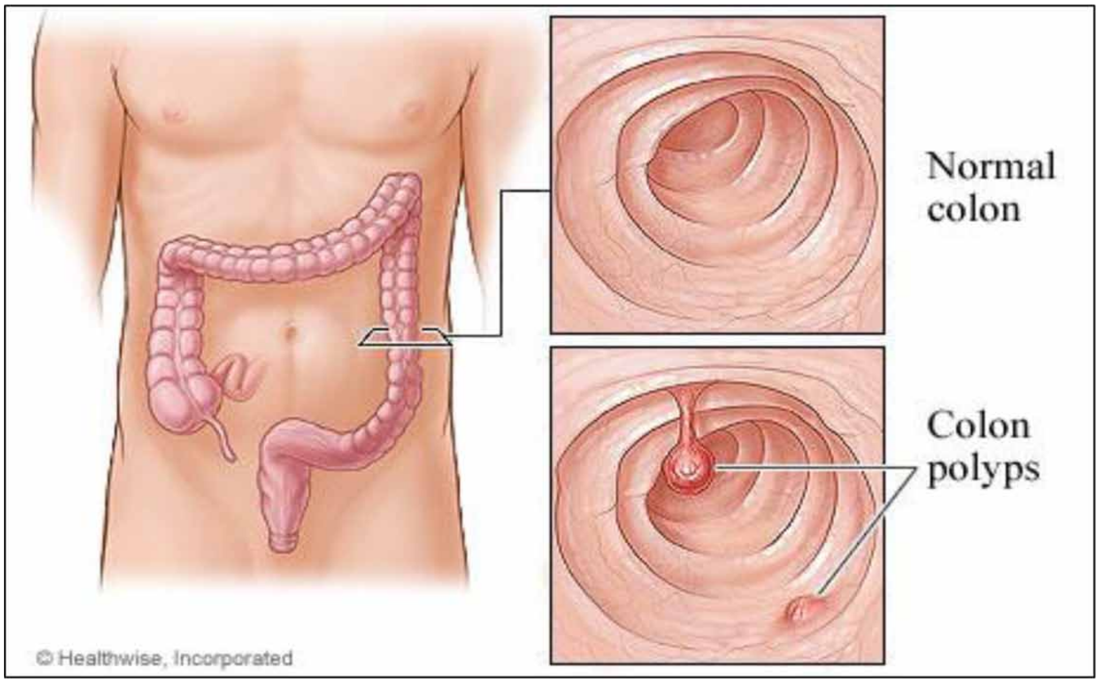
Figure 4. Normal cells versus cancer cells
Source [Adapted from]: (Wilkin, 2017). 'Gene Regulation and Cancer – Advanced' [Image]. Retrieved from https://www.ck12.org/biology/gene-regulation-and-cancer/lesson/Gene-Regulation-and-Cancer-Advanced-BIO-ADV/



2.7 Microarray Dataset Technology

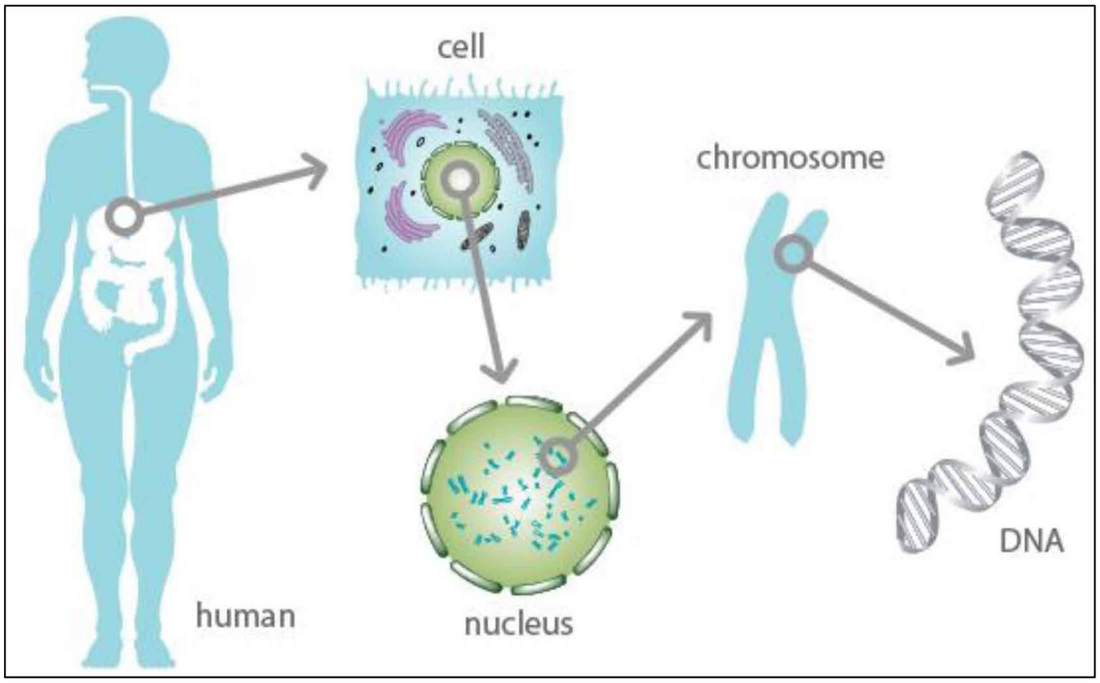
Cells are the basic composition of all living organisms. A cell is the basic unit and building block of life, because it exhibits all life characteristics. Each cell has a nucleus containing DNA, as shown in **Figure 6**. The DNA is the genetic material that encodes the program of the RNA and other proteins, constituting an organism (Alshamlan, Badr, & Alohali, 2013; Babu & Sarkar, 2016; Bolón-Canedo et al., 2014). DNA produces proteins through a well-known two-step process (Babu & Sarkar, 2016) as illustrated in Figure 7. Step one is commonly recognised as transcription, in which the gene inside the DNA is expressed by the information coded to the messenger RNA (mRNA). Step two is commonly known as translation, in which the mRNA information is decoded by ribosomes to produce proteins. These two steps result in a specific protein (i.e., gene expression). That is, the gene expression is the level measurement of an active gene inside a specific tissue of the body via the mRNA (Alshamlan, Badr, & Alohali, 2013).

Figure 5. A hypothetical image for the colon, normal colon and colon polyps
Source [Adapted from]: MyHealth.Alberta.ca. (2018). 'Colon Polyps: Care Instructions' [Image]. Retrieved from https://myhealth.alberta.ca/Health/aftercareinformation/pages/conditions.aspx?hwid=ut2896



Gene expression is a vital biological process, as indicated by the quantity of mRNA generated in each cell during protein composition. If genes mutate, they may express in an uncontrolled manner, generating a tumour: an expression of altered genes. There are many new technologies that give researchers a full view of the cell and to measure tens of thousands of genes instantaneously. Microarrays are part of this new technology, which have been extensively used for medical diagnoses (Lee & Leu, 2011). This technology has proven effective in bioinformatic gene identification, cell differentiation, disease diagnosis, disease prediction, cancer classification and pharmaceutical development (Ali & Gupta, 2006; Chuang et al., 2009; Horng et al., 2009; Hsieh & Chou, 2016, Nahum et al, 2019). An exemplary microarray contains information about a huge amount of DNA molecules. Based on the DNA spot number, arrays can be classified into microarrays whenever the DNA spot diameter is less than 250 μm and can be classified into macro-arrays when the same diameter is greater than 300 μm. Small solid substrate arrays are known as DNA chips. Microarrays are influential in that gene information can be investigated in less time, because large numbers (e.g., hundreds or thousands of genes) can be examined when placed on a DNA microarray (Cho & Won, 2003).

Figure 6. DNA location inside the nucleus of each cell of the living organism
Source [Adapted from]: GB-HealthWatch (2018), 'Genetics 101' [Image]. Retrieved from: https://www.gbhealthwatch.com/Trait-Genetics-101.php



The main component of microarray technology is hybridisation. The technology starts with the removal of mRNA from a probe or a sample. Then, the mRNA is labelled with red fluorescent nucleotide: complementary DNA (cDNA). Green fluorescent nucleotides can also be used. The labelled probe and the reference are placed together and organised onto the microarray surface. A laser processes the populated microarray, and each significant fluorescent spot is measured. If no hybridisation exists in the probe or reference sample, and the gene is marked on the slide, then this mark or spot will appear black. However, if the hybridisation occurs mainly with the probe, the mark or the spot will appear red (*Cy*5). However, if the hybridisation occurs mainly between the reference and the DNA, then the spot will appear green (*Cy*3). If both the probe and reference samples hybridise at the same specified spot, then the colour appears yellow (George & Raj, 2011). Thus, gene expression can be computed using the log ratio between the two concentrations of red and green, as shown below.

$$gene\_expression = log_2 \frac{Intensity\ of\ (Cy5)}{Intensity\ of\ (Cy3)} \tag{2-1}$$

where *Intensity of* (*Cy*5) and *Intensity of* (*Cy*3) are the concentrations of red and green colours.

Figure 7. Production of proteins by DNA expression (gene expression)
Source [Adapted from]: Yourgenome (2016). 'What is the 'Central Dogma'?' [Image]. Retrieved from: https://www.yourgenome.org/facts/what-is-the-central-dogma

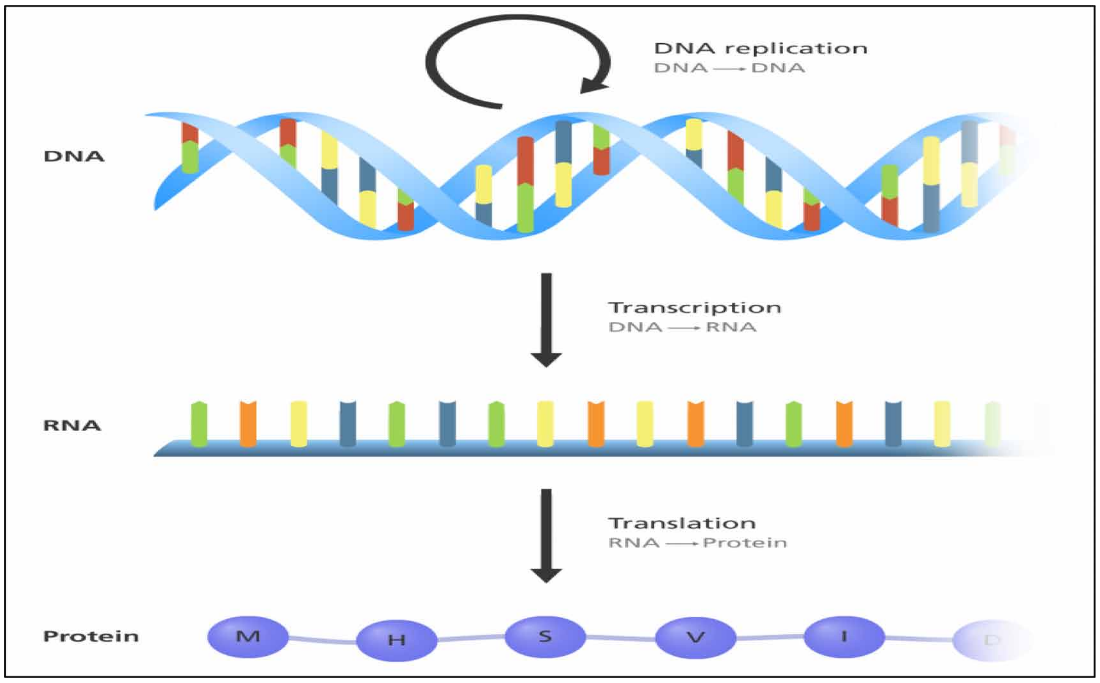


Figure 8 mirrors the core stages that recognise and analyse DNA microarray technology. The microarray is scanned, analysed and transformed into digital data after hybridisation, facilitating analysis. This can be generated when thousands of distinct DNA sequences are printed on a high concentration array on a glass using a robotic printing device (Banerjee, Mitra, & Banka, 2007; Cho & Ryu, 2002, Ghosh et al, 2019). Thereafter, the gene expression data is formulated by a matrix template where genes are represented in columns, and the different samples are represented in rows (e.g., tissues or experimental condition). The numeric values inside each cell characterise a particular gene's expression level for a particular sample (Lavanya et al., 2014). *N* represents the number of tissue/samples, and *M* represents a gene's expression level. Then, microarray data can be stored using a matrix template, $N\times(M+1)$, as illustrated in Figure 9, where $g(i,j)$ is the expression level value of gene *j* in a tissue/sample, *i*.

The main objective of DNA microarray studies is categorised into three main objective sets: gene searching, class discovery, and class prediction (Alshamlan, Badr, & Alohali, 2013) Table 1, illustrates each of these groups with their related approach and purpose.

Figure 8. Main steps of DNA microarray technology analysis
Source [Adapted from]: (Lowery et al., 2011)

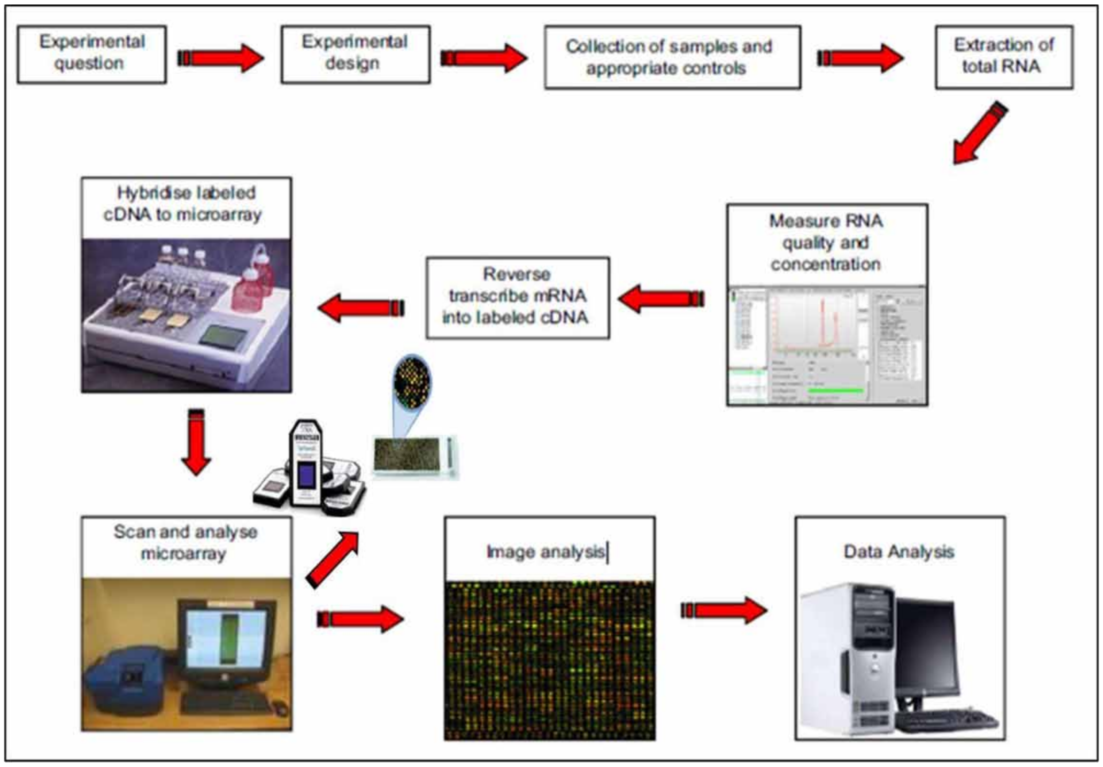

Experimental question
Experimental design
Collection of samples and appropriate controls
Extraction of total RNA
Measure RNA quality and concentration
Reverse transcribe mRNA into labeled cDNA
Hybridise labeled cDNA to microarray
Scan and analyse microarray
Image analysis
Data Analysis

Figure 9. Microarray data matrix template

$$\begin{array}{c|cccc|c}
 & \multicolumn{4}{c|}{\textbf{M genes}} & \textbf{Class label} \\
\hline
 & g_{1,1} & g_{1,2} & \cdots & g_{1,M} & l_1 \\
 & g_{2,1} & g_{2,2} & \cdots & g_{2,M} & l_2 \\
\textbf{N Samples} & \vdots & \vdots & \vdots & \vdots & \vdots \\
 & g_{N,1} & g_{N,2} & \cdots & g_{N,M} & l_N
\end{array}$$

2.8 Colon Dataset

As discussed earlier, many promising methods exist for cancer detection. For colon cancer, the microscopic inspection of colon polyps is the most common method used. However, this method is very time-consuming. Additionally, it is very difficult for physicians to discern the variations using this method. Thus, there is an urgent need to automate a faster approach to colon cancer detection. Many approaches and methods have been proposed. Generally, however, there exist two main types of detection methodologies that use datasets: biopsy image analysis and physical sample analysis (Figure 2-10). The cDNA microarray technology presents a promising method that allows better understanding of cell regulation activities and tumours at different states, as discussed in Section 2.3 (Kim & Cho, 2004; Rathore, Hussain, & Khan, 2014).

Table 1. Microarray gene expression profile main objectives

Objective	Approach	Purpose
Gene Searching	Gene Selection	Search and select the most related and informative genes form the microarray dataset
Class Discovery	Clustering	Detect new disease/cancer
Class Prediction	Classification	To classify samples into normal or cancerous types

Currently, there exists no central database for human genome expression data (Lu & Han, 2003). There exists however, several publicly accessible colon cancer repositories, which include DNA microarray data table 2, lists the common ones.

Figure 10. Main colon-cancer detection techniques
Source [Adapted from]: (Rathore et al., 2013)

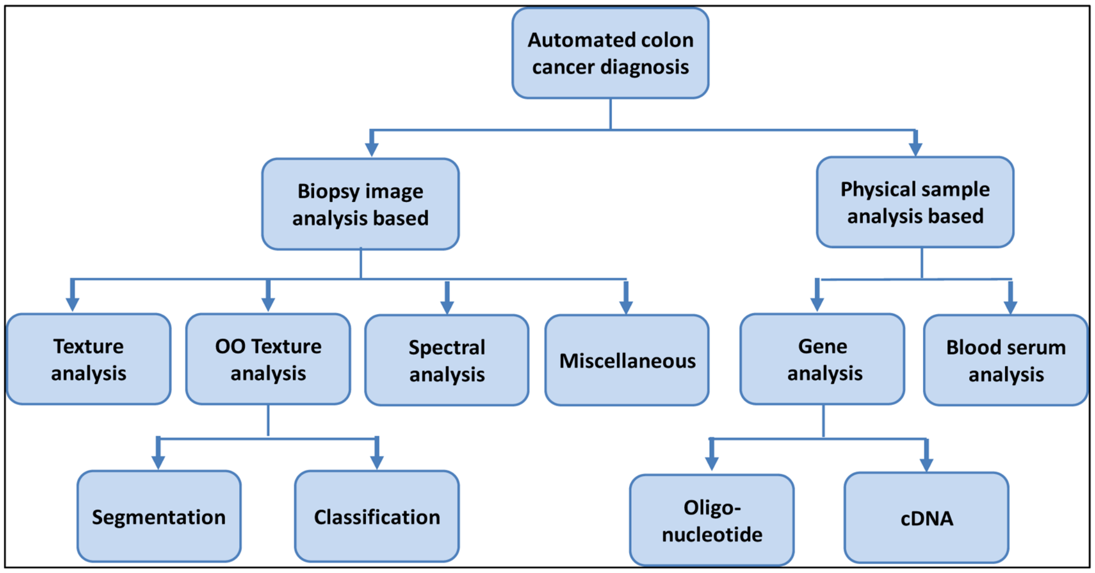


Table 2. Colon cancer public datasets

Reference	Dataset	Description
(Venkatesh, Tangaraj, & Chitra, 2010)	Kent Ridge Biomedical Dataset	An online database for high-dimensional genetic datasets; contains data for genetic expression, profiling protein and genomic sequence.
(Jourdan, Dhaenens, & Talbi, 2001)	Artificial Database	Used for the Genetic Analysis Workshop (Workshop Challenge GAW11).
(Wang, Chu, & Xie, 2007)	GCM Dataset	Contains 14 types of cancer, including 16,306 genes and 198 samples total.
(Rhodes et al., 2004)	ONCOMINE	A centralised gene database, contains information about bioinformatics from resources such as Swiss-Port, LocusLink, Unigene and others.
(Chitode & Nagori, 2013)	KEGG, Gene Ontology (GO)	Provides gene observations.

The data utilised in this research were obtained from one of the most common colon cancer datasets featuring tissues (Alon et al., 1999), available from the Princeton University gene expression database (Table 2). This dataset has been widely used for numerous colon-cancer studies. Furthermore, it is publicly available. Appendix 1 contains a reference list of these studies.

The colon cancer dataset is a binary-class microarray dataset and a gathering of diverse expressions comprising 62 patient samples. These colon dataset samples were collected within 20 min of removal from the colon adenocarcinoma snap specimens that were frozen in liquid nitrogen (Alon et al., 1999). Additionally, tumour biopsies were extracted from tumour cells, and normal biopsies were extracted from healthy cells from similar patients. The pathological examination demonstrates the status of the final biopsy specimen distribution. Among them, 40 biopsies were only tumours, and the 22 remaining ones were normal biopsies from the colon of the same patient. From these 62 samples, gene expression levels were computed using high-concentrated oligonucleotide arrays. Out of approximately 6500 human genes recorded in these arrays, only 2000 were chosen, based on the computed expression levels confidence. The dataset is publicly available at http://microarray.princeton.edu/oncology/affydata/index.html.

The microarray dataset in this study is represented by an $N \times M$ matrix, where N represents the number of samples (i.e., 62), and M is 2000, representing the number of genes (Archetti et al., 2008). Table 3 presents all related dataset information employed for this research, including class numbers, total samples, number of genes and classification types.

Table 3. Gene expression dataset description employed for this investigation

Dataset Type	Number of Classes	Total Number of Samples	Number. of Genes	Type Classification	Number of Samples
Colon Cancer	2	62	2000	Tumour (Positive)	22
				Normal/Healthy (Negative)	40

Because the extracted data contain variations and noise, pre-processing is required. Two popular and well-known pre-processing approaches are employed with the dataset samples.

1. A normalisation technique is utilised for this target, where classification algorithms apply the same gene expression measurements. The advantage of normalisation is that it regulates bias originating from the variation of the microarray technology, instead of that coming from the biological differences of RNA or the printed probes (Alladi et al., 2008). Once the data are prepared, the next step is reducing the dimensionality of the dataset using the feature selection process. Thus, prior to employing colon data in this research, it is normalised using min–max normalisation (Han, Pei, & Kamber, 2011). This process rescales the genes to a new range of values between 0 and 1. The min–max normalisation is evaluated using the following equation.

$$x[i]=\frac{(x[i]-minValue)}{(maxValue-minValue)} \tag{2-2}$$

where x represents the attribute, i indicates the number of the samples, *minValuw* indicates each attribute minimum value and *maxValue* indicates each attribute maximum value. This approach is widely used by researchers in the field (Li et al., 2003; Lorena, Costa, & de Souto, 2008; Mishra, Devi, & Shrivastava, June 2015; Pattanateepapon, Suwansantisuk, & Kumhom, 2016; Shah, Saad, & Othman, 2009; Wang et al., 2013; Yang, Zhou, et al., 2010; Yeh et al., 2007; Zhang et al., 2009).

2. Another popular method involves pre-processing the data and discerning them using entropy-based discretisation, as suggested by Fayyad and Irani (1993). This method is common as a global discretisation method and has been used by many researchers for the same topic (Dash & Patra, 2012; Dash, Patra, & Tripathy, 2012; El Akadi et al., 2009a, 2009b; Fang, Mustapha, & Sulaiman, 2010; Ong, Mustapha, & Sulaiman, 2011; Wang & Gotoh, 2009b). The data are discretised categorically to reduce noise. This method implements an entropy minimisation heuristic, which recursively discretises the continuous attributes by dividing their range into intervals. In general, assuming the cut point T for f features is going to cut or split the S sample set into two subsets of S_1 and S_2. Thus, the split entropy class information is given by Fayyad and Irani (1993) and Huiqing (2004).

$$E(f,T,S)=\frac{|S_1|}{|S|}E(S_1)+\frac{|S_2|}{|S|}E(S_2) \tag{2-3}$$

where $E(S_i),(i=1,2)$ represents the class entropy of the s subset. Suppose there are k classes $(C_1,C_2,\ldots,C_k)$, then the samples proportion in S_i which have the class C_j is given by $P(C_j,S_i)$. So, the entropy of S_i with reference to the k classes given by.

$$E(S_i)=-\sum_{j=1}^{k}P(C_j,S_i)\times\log_2 P(C_j,S_i) \tag{2-4}$$

The f features will be binary discretized through selecting the cutting point of T_f by which $E(f,T_f,S)$. is computed to be the smallest amount over all the candidate cutting points. The T_f selection process recursively splits the ranges of S_1 and S_2. This method halts the recursive step, depending on the principle of minimum description length (Wang & Gotoh, 2009a).

2.9 Common Cancer-Gene Selection and Classification Aorithms Implemented

Array technology development provides the opportunity for early and accurate cancer detection. Applying these technologies, one collects thousands of gene expression levels instantaneously via arrays. It also enables classification of cancer cells (Park & Cho, 2003). Therefore, there is an increasing demand to demonstrate informative genes that directly indicate a carcinogenic situation (Al-Rajab & Lu, 2016). An informative gene will certainly be helpful and relevant to cancer classification (Al-Rajab & Lu, 2016; Mohamad et al., 2008). Cancer classification helps enhance patient healthcare and supports individuals through life-quality enhancements. It is also important for diagnosing cancer and discovering better drugs. Cancer classification is the procedure of establishing a paradigm using a microarray dataset. It discriminates different samples within the developed paradigm (Liu, Liu, & Zhang, 2010). The objective of cancer classification using microarray data technology is to discriminate tissue samples into categories of relevant phenotypes versus normal cells (Al-Rajab & Lu, 2016; Mohamad et al., 2007). The main issue with these microarray sets, when employed with many genes, is the high redundancy and their noisy nature, negatively affecting cancer classification accuracy. Only a few limited genes may be important (Osareh & Shadgar, 2010). Accurate cancer discovery and classification at an early stage is important to patients. The need for a cancer identification method or algorithm is crucial and will critically influence treatment and medical services. For practical uses, any proposed algorithm or method must be fast, accurate and easily designed, implemented, applied, tested and maintained. The optimal and best algorithm for a given task in cancer classification should show good performance with low implementation complexity (Al-Rajab & Lu, 2016; Nurminen, 2003).

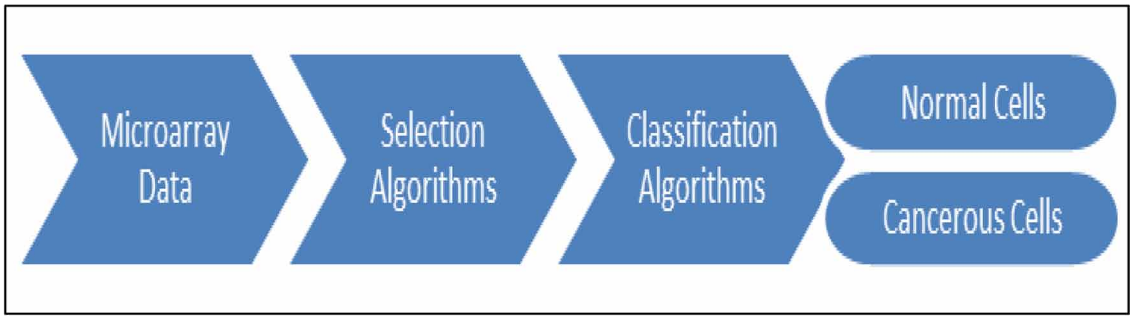
Common algorithms are used for gene search and cancer classification. Thus, an extensive literature review is conducted, beginning with understanding bioinformatics, followed by algorithmic interpretation and analysis. Then, the focus turns to how algorithms are implemented efficiently to classify cancer tissues (Al-Rajab & Lu, 2016). There has, consequently, been a major development of modern techniques and tools, such as microarray data technology and datasets loaded with a huge number of genes/features (Al-Rajab & Lu, 2014, 2016). Therefore, gene selection and extraction has become a very significant method of eliminating noise and useless data. Many researchers have examined methods and algorithms to enhance genesearch and cancer classification results, realising that methods having high cancer classification accuracy will most certainly supply more relevant information for disease detection and treatment enhancement (Al-Rajab & Lu, 2016; Jyun-Jie & Pei-Chann, 2010).

2.9.1 Summary of Algorithms Implemented for Different Cancer Categories

This section examines several popular algorithms implemented in the research area for different cancer types. Gene selection algorithms should be applied prior to classification to minimise the number of genes and facilitate the classification process (Al-Rajab & Lu, 2016), as illustrated by Figure 11.

Several gene selection and cancer classification algorithms were investigated for the literature review. Cancer gene selection is a pre-processing stage used to produce a minimised microarray data subset (Al-Rajab & Lu, 2014, 2016). Table 2-4, summarises the employment of each algorithm per the relevant

Figure 11. Consistent gene selection and cancer classification principle



cancer type, as investigated from the literature review in (Al-Rajab & Lu, 2016; Chen, 2003; Lee, 2008; Osareh & Shadgar, 2010; Shah & Kusiak, 2007; Yeh et al., 2007).

From Table 4 the GA was practically applied to all cancer types with good performance, except for brain cancer. On the other hand, the DT and SVM algorithms were applied to all cancer types, resulting in good performance results (Al-Rajab & Lu, 2014). Moreover, from the contributions of several researchers, Table 5 categorises the most popular algorithms employed for gene search and cancer-cell classification. Many of these algorithms were implemented with other models to produce higher performance results (Al-Rajab & Lu, 2014, 2016).

Table 4. Different Algorithms Implemented for different cancers feature selections and classifications. Source [Adapted from]: (Al-Rajab & Lu, 2016)

Cancer Algorithm	**Bladder**	**Brain**	**Breast**	**CNS**	**Colon**	**Leukaemia**	**Lung**	**Lymphoma**	**Ovarian**	**Prostate**
Analysis of variance (ANOVA)	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗
Correlation based heuristics (CFS)	✓	✗	✓	✗	✓	✓	✓	✗	✓	✓
Decision tree (DT)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fuzzy Model	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗
Genetic Algorithm (GA)	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Information Gain (IG)	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓
K nearest Neighbour (KNN)	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓
Naïve Bayes (NB)	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓
Neural Network (NN)	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓
Particle Swarm Optimisation (PSO)	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗
Relief Algorithm (RA)	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓
Support Vector Machine (SVM)	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓
t-statistics (T-Test)	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 5. Most popular gene selection and cancer classifications algorithms

Selection Algorithms	Classification Algorithms
Analysis of Variance (ANOVA)	Bagging and Stacking Algorithms
Correlation-based feature selection (CFS)	Bootstrapped SVM
Genetic Algorithm (GA)	Decision Tree (DT)
Information Gain (IG)	Fuzzy Model
Particle Swarm Optimisation (PSO)	K-Nearest Neighbours (KNN)
Relief Algorithm (RA)	Naïve Bayes
t-statistics (TA)	Neural Networks (NN)
Minimum Redundancy Maximum Relevance (mRMR)	Support Vector Machine (SVM)
	Genetic Programming (GP)

2.10 Related Work and Previous Research Findings

High-performance algorithms for gene selection and cancer classification have enjoyed massive use in the ML domain in support of medical studies (Al-Rajab, Lu, & Xu, 2017; Hassan, Siuly, & Zhang, 2016; Hassan & Subasi, 2016; Kirar & Agrawal, 2017). Hassan and Subasi employed gene selection and proposed a leaner programming boosting (LPBoost) method as a classifier method that allowed controlling epileptic seizures, contributing to patient treatment (Hassan & Subasi, 2016). Kirar and Agrawal (2017) differentiate electro-encephalogram brain signals via ML. They ranked brain-segment regions using the recursive gene exclusion selection technique with the composite kernel SVM classifier algorithm. In (Hassan & Haque, 2015), the authors proposed an efficient real-time algorithm to detect small-intestine bleeding by employing wireless endoscopy capsule videos to produce a vast database of digital images. SVMs were used to classify these images and to detect digestive system outflows, greatly assisting physicians.

Furthermore, huge amounts of genetic data can be processed instantaneously and can be applied as a root for all gene selection (Mukkamala et al., 2005). This technology can be applied to study gene expression levels from a vast amount of genes simultaneously to produce gene expression data that can be more easily interpreted (Al-Rajab, Lu, & Xu, 2017; Mohamad et al., 2008). Shah and Kussaik asserted that collecting genetic data is a costly process. They observed that not all extracted genes are useful. However, there must be a process to extract the most proper genes from the enormous dataset. This process eliminates useless and repeated gene selection, eliminates noise and decreases complexity, maintaining active genes (Shah & Kusiak, 2007). A gene classification process typically follows a series of actions: pre-processing (i.e., normalisation and reduction of gene expression), gene selection and gene-cancer classification (Al-Rajab, Lu, & Xu, 2017). Jäger, Sengupta, and Ruzzo (2002) recognised that, if an arrangement of associated microarray genes is inspected across diverse situations, they will express differently and will be altered under conditions producing a phenomenon known as feature selection. New techniques for determining the best genes while differentiating the cancer cells is a major problem for ML to help solve (Jäger, Sengupta, & Ruzzo, 2002). Khobragade and Vinayababu (2012) investigated cancer tumour categorisation procedures to organise tissues into different categories, such as normal versus cancer. Hence, selecting useful and relevant subsets of genes, which decreases computational time and efforts and enhance the classification accuracy, results in better efficiency (Bennet, Ganaprakasam, & Kumar, 2015; Khobragade & Vinayababu, 2012).

Furthermore, redundancy is found in most genes. Thus, gene selection is applied and executed for the sake of selecting and extracting a small subset of genes (Bonilla-Huerta et al., 2016). Jeyachidra and Punithavalli highlighted the existing collection of cancer gene selections and classification algorithms implemented in the field of ML (Jeyachidra, 2013). A set of these algorithms achieved good results over others in terms of accuracy (Al-Rajab, Lu, & Xu, 2017). However, efforts still need to be taken to compare gene selection and cancer classification algorithms in terms of temporal performance with relation to accuracy as applied to cancer prediction (Al-Rajab, Lu, & Xu, 2017). Thus, time-complexity analysis is a key factor of comparison between the different algorithms.

The authors in (Lee & Leu, 2011; Zhu, Ong, & Dash, 2007) found that there exists various feature selection and classification algorithms that predicate microarray data analysis. However, none suggest the number of genes needed for analysis. Plus, these algorithms cost too much computational time. The authors in (Sun et al., 2013) concluded that there is a problem with most feature selection algorithms, because they often ignore important and informative features having strong effective power related to cancer detection. The recent study of (Zeng et al., 2015) concluded that feature selection methods should be simple, robust and efficient.

The overview recently provided by Al Snousy et al. (2011), Hsieh and Chou (2016), Pattanateepapon, Suwansantisuk, and Kumhom (2016), and Peng et al. (2011) found that no best algorithm exists for any classification. However, it is possible to develop an optimum method or algorithm in the domain of one specific problem. According to Rathore, Hussain, and Khan (2014), and Sharma, Imoto, and Miyano (2012), not all gene expressions are related, but can well-contribute to cancer classification. Therefore, there exists a persistent need for efficient computational data analysis techniques and methods to discard unrelated genes to keep only the informative ones. Several inferences (Ammu & Preeja, 2013; Elyasigomari et al., 2017; Mohamad et al., 2010) suggested that many studies utilised filtered methods of feature selection algorithms for the purpose of gene selection, because they were more efficient computationally than the hybrid method. Though, hybrid methods resulted in better classification accuracy compared to filter methods.

2.10.1 Feature Selection and Cancer Classification Background

Several studies have been conducted to examine colon cancer classification processes utilising microarray genetic data. A range of the most relevant and recent studies are summarised in the following sections.

2.10.1.1 Algorithms Reviewed

This section briefly summarises several works that implemented a variety of gene selection and cancer classification algorithms in the area of colon cancer (Al-Rajab, Lu, & Xu, 2017). Table 6 summarises the accuracy results for these recent studies.

Table 6 shows a summary of recent results from background studies on gene selection and colon cancer classification prediction accuracy. 13 out of 31 methods achieved an accuracy of 90% or greater when implementing the colon dataset, whereas the other methods accomplished a prediction classification accuracy between 69% and 89%. The algorithms reflecting high-impact accuracy in terms of colon-cancer classification were SVM, GP, and DT. It can be observed that SVM and GP classifier algorithms produce

Table 6. Literature review for colon-cancer classification accuracy results

No.	Reference	Method		Accuracy [%]
		Feature Selection	**Classifier**	
1.	(Osareh & Shadgar, 2010)	● Information Gain (IG) ● Relief Algorithm (RA) ● t-statistics (TA)	Support Vector Machine (SVM)	99.90
2.	(Huerta, Duval, & Hao, 2006)	● Genetic Algorithm (GA)	Support Vector Machine (SVM)	99.41
3.	(Mohamad, Deris, & Illias, 2005)	● New Genetic Algorithm (New-GA)	Support Vector Machine (SVM)	98.39
4.	(Kulkarni et al., 2011)	● t-statistics (TA)	Genetic Programming (GP)	98.33
5.	(Yang & Zhang, 2007)	● Genetic Algorithm (GA)	Decision Tree (DT)	96.79
6.	(Tan & Gilbert, 2003)	● ---	Single C4.5 (DT)	95.16
7.	(Yang & Zhang, 2007)	● Genetic Algorithm (GA)	Neural Network (NN)	94.92
8.	(Mohamad, Omatu, Deris, & Yoshioka, 2009)	● Improved Particle Swarm Optimization (IPSO)	---	94.19
9.	(Cho & Won, 2003)	● Euclidean Distance (ED)	● Cosine Kernel KNN ● Pearson Kernel KNN	93.90
10.	(Li, Wu, & Hu, 2008)	● Genetic Algorithm (GA)	Polynomial Kernel SVM RBF Kernel SVM	93.60
11.	(Fang, Mustapha, & Sulaiman, 2010)	● Information Gain (IG) with Association Analysis	Support Vector Machine (SVM)	93.55
12.	(Fang, Mustapha, & Sulaiman, 2010)	● Information Gain (IG)	Support Vector Machine (SVM)	90.33
13.	(Mohamad, Deris, & Illias, 2005)	● Genetic Algorithm (GA)	Support Vector Machine (SVM)	90.32
14.	(Yeh et al., 2007)	● t-GA	Decision Tree (DT)	89.24
15.	(Yeh et al., 2007)	● Genetic Algorithm (GA)	Decision Tree (DT)	88.8
16.	(Mohamad, Omatu, Deris, & Yoshioka, 2009)	● Binary Particle Swarm Optimization (BPSO)	---	86.94
17.	(Mohamad, Deris, & Illias, 2005)	● Genetic Algorithm (GA)	Support Vector Machine (SVM)	85.48
18.	(Salem, Attiya, & El-Fishawy, 2017)	● Information Gain (IG) & Standard Genetic Algorithm	Genetic Programming (GP)	85.48
19.	(Salem, Attiya, & El-Fishawy, 2015)	● Information Gain	Genetic Programming (GP)	85.48
20.	(Kulkarni et al., 2011)	● t-statistics (TA)	Decision Tree (DT)	85.00
21.	(Alladi et al., 2008)	● t-statistic (TA)	RBF Kernel SVM	84.09
22.	(Wang & Gotoh, 2010)	● ---	Support Vector Machine (SVM)	83.12
23.	(Pradhananga, 2007)	● ---	Naïve Bayes (NB)	83.07
24.	(Hu et al., 2006)	● ---	Decision Tree (DT)	82.30
25.	(Pradhananga, 2007)	● ---	Decision Tree (DT)	81.95
26.	(Tan, Dowe, & Dix, 2007)	● ---	Decision Tree (DT)	80.90
27.	(Hengpraprohm & Chongstitvatana, 2007)	● K-Means Clustering ● SNR Ranking	Genetic Programming (GP)	79.80
28.	(Yeh et al., 2007)	● t-statistics (TA)	Decision Tree (DT)	77.42
29.	(Yeh et al., 2007)	● Information Gain (IG)	Decision Tree (DT)	77.26
30.	(Cho & Won, 2003)	● Information Gain (IG) ● Mutual Information (MI)	Linear Kernel SVM RBF Kernel SVM	71.00
31.	(Yeh et al., 2007)	● GS Method	Decision Tree (DT)	69.35

a high percentage of accurate results when employed with IG and GA (90%–100%). The integration of GA and SVM provided better accuracy than the combination of GA and DT. The table also presents different outcome accuracy results using the same methods of classification. More elaboration on the existing problems and limitations are discussed in Section 2.15.

2.10.2 Hybrid Feature Selection and Classification Background

The hybrid model is an approach originating from a combination of selection methods. It is used to solve high computational complexity problems of selection methods and slowness algorithms. This approach reflects the advantages of multiple selection algorithms that have played a significant role in cancer classification (Hoque, Bhattacharyya, & Kalita, 2014).

Many researchers have focused on the implementation of hybrid algorithms to increase classification accuracy. Ammu, Siva Kumar, and Mundayoor (2014) applied a hybrid feature selection algorithm that calculated information gain values, then applied a biography-based optimisation algorithm. Chuang et al. (2011) suggested a hybrid method for gene selection composed of a dynamic parameter genetic algorithm with the x^2 test to choose a suitable number of top-ranked genes. Later, they applied SVM to estimate the selected genetic classification accuracy.

Dash and Patra (2012) A hybrid method composed of filter and wrapper approaches. They employed a correlation-based feature selection approach (CFS) to determine genetic subsets and three gene selection wrappers (i.e., J48, random forest (RF) and random trees). Their performance was evaluated with K-nearest neighbour (K-NN) and SVM classifiers. El Akadi et al. (2009a) proposed a two-stage selection algorithm composed of mRMR and GA. The proposed method was tested with the colon cancer dataset using the SVM and NB classifier. Another contribution of Wang et al. (2004) explored a hybrid method that integrated gene-ranking and clustering methods. Their filtering algorithms were applied to extract top-ranked genes before applying a hierarchical clustering approach. They tested the results with four classification algorithms (i.e., K-NN, SVM, C4.5 and NB). Tan et al. (2006) proposed a hybrid method using a GA to enhance gene-subset selection via the integration of better outcomes from several gene selection methods using the colon cancer dataset, testing with an SVM. Additionally, another algorithm, suggested by (Kim & Cho, 2004), used an evolutionary neural network for colon-cancer classification. Mohamad et al. (2010), implemented a cyclic-GA/SVM hybrid method for gene selection using microarray data.

Moreover, Mohamad, Omatu, Deris, Misman, et al. (2009), implemented an approach based on the hybridisation of feature selection, based on GA/SVM, called GASVM-II + GASVM. Another contribution was provided by Salem, Attiya, and El-Fishawy (2017), who presented a methodology combining both information gain and the standard GA for feature selection. They also evaluated the results of GP. Elyasigomari et al. (2017), proposed a method called mRMR–COA–HS. In their method, the mRMR selection technique was used to produce a subset of related genes. Then, the chosen genes were input into a wrapper system that integrated an algorithm (COA–HS) using the SVM classifier. Additionally, Alshamlan, Badr, and Alohali (2015) proposed an innovative feature selection algorithm composed of mRMR and an ABC algorithm for feature selection, tested by an SVM.

2.10.2.1 Algorithms Reviewed

Table 7 shows a summary of colon-cancer classification accuracy results for recent studies using the hybridisation, multifilter and similar algorithms of feature selection methods. Three methods were found to achieve high classification accuracies above 90% when implemented to the colon dataset, whereas the rest achieved a cancer classification accuracy between 66 and 89%. Among these algorithms, 11 applied an SVM as the ML algorithm, noted as having the highest classification accuracy. However, NB, K-NN and RF algorithms presented the lowest classification accuracies. GA, PSO and mRMR methods showed noticeable selection contributions for high classification results.

2.11 Feature Selection Approaches for ML

This section explores the ML concept with its applications and goals. Furthermore, this section discusses the concept of feature selection using ML, its advantages and its characteristics. Finally, the section concludes with the feature selection procedures and categories.

2.11.1 Machine Learning (ML)

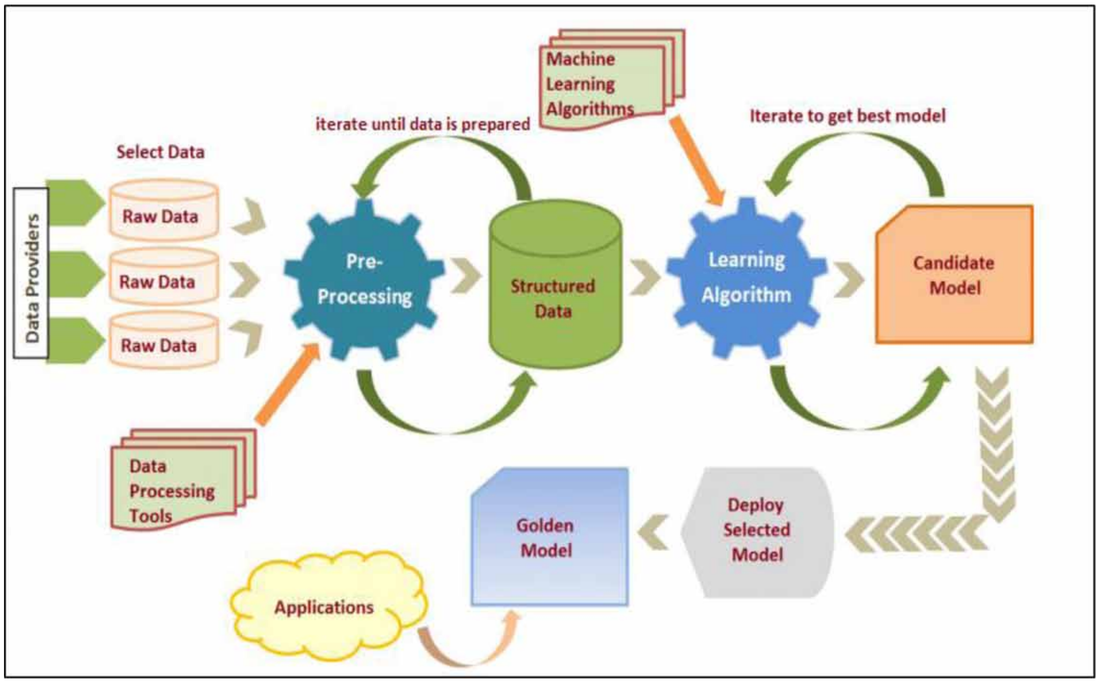
The ML field is an area of AI that has shown enormous contributions in the last decades, owing to its significant applications (Pradhananga, 2007). ML can be defined in multiple ways, but these ways intersect in a main definition. ML is interested in the study of methods, design and algorithm development, which automatically allows computers to learn and improve their performance from experience and to make decisions based on data (Hall, 1999; Janecek, 2009; Tan, 2007). ML is a data-mining process and gains advantages from fields such as AI, statistics, mathematics, biology, physics, information theory, cognitive science and philosophy. Data mining is the application of ML algorithms to large databases (Kilany, 2013). This employs the same idea as gold mining, where a large volume of raw material is extracted, mined and refined. Then, analytical processes can be applied to obtain the gold. The same mechanism is applied to data mining, where a massive amount of data is processed and analysed to build informative and predicting models with better accuracy, thus improving performance time.

Recently, large amounts of data have been stored, quickly processed, and accurately interpreted. Additionally, the data was accessed from distinct network locations. This occurs because of the revolution of computer technologies and huge advanced computing performances. Row data is stored in a digital format to make data mining easy, fast, and reliable. Recent computer breakthroughs should be utilised. However, the vast amount of data requires tremendous storage space, and accessing or gaining a useful information is a complex problem. To overcome this issue, algorithms should be developed to convert the massive amounts of data into useful information (Kilany, 2013). These data become useful when they can be smoothly analysed and converted into meaningful and relevant information to make predictions. There must always be a way to explain the data retrieved, but we also need to know how it can be processed. The main role of data mining is implementing ML algorithms to discover how the data can be retrieved, related to each other to provide useful information and assist in making predictions.

Table 7. Colon-Cancer hybrid methods for classification accuracy literature review

No.	Reference	Method		Accuracy [%]
		Hybrid Feature Selection	Classifier	
1.	(Li, Wu, & Tan, 2008)	PSO+GA	Support Vector Machine (SVM)	91.90
2.	(Abdi, Hosseini, & Rezghi, 2012)	mRMR + PSO	Support Vector Machine (SVM)	90.32
3.	(Mohamad, Deris, & Illias, 2005)	Genetic Algorithm (GA)	Support Vector Machine (SVM)	90.32
4.	(Lu et al., 2017)	MIM+AGA	Extreme Learning Machine (ELM)	89.09
5.	(Dash & Patra, 2012)	CFS + Wrapper (J48)	Support Vector Machine (SVM)	89.03
6.	(Yeh et al., 2007)	Genetic Algorithm (GA)	Decision Tree (DT)	88.80
7.	(Sreepada, Vipsita, & Mohapatra, 2015)	Filter (F-Score + IG) + Wrapper (SBE)	Support Vector Machine (SVM)	87.50
8.	(Sreepada, Vipsita, & Mohapatra, 2015)	F-Score + IG SBE + SBS	Support Vector Machines	87.50
9.	(Dash & Patra, 2012)	CFS + Wrapper (Random Forest)	K- Nearest Neighbour (KNN)	87.10
10.	(Dash & Patra, 2012)	CFS + Wrapper (Random Forest)	Support Vector Machine (SVM)	87.10
11.	(Li, Wu, & Tan, 2008)	PSO+GA	Naïve Bayes (NB)	85.50
12.	(Dash & Patra, 2012)	CFS + Wrapper (J48)	K- Nearest Neighbour (KNN)	85.48
13.	(Dash & Patra, 2012)	CFS + Wrapper (Random Trees)	Support Vector Machine (SVM)	85.48
14.	(El Akadi et al., 2009a)	mRMR	Support Vector Machine (SVM)	85.48
15.	(Salem, Attiya, & El-Fishawy, 2017)	Information Gain (IG) & Standard Genetic Algorithm (SGA)	Genetic Programming (GP)	85.48
16.	(Li, Wu, & Hu, 2008)	mRMR + GA-SVM	Support Vector Machine (SVM)	85.48
17.	(El Akadi et al., 2011)	mRMR + GA	Support Vector Machine (SVM)	85.48
18.	(Zhang et al., 2009)	GE Hybrid	7-Nearest Neighbour	85.34
19.	(Zhang et al., 2009)	GE Hybrid	Naïve Bayes (NB)	84.96
20.	(Zhang et al., 2009)	GE Hybrid	3-Nearest Neighbour	84.93
21.	(Li, Wu, & Tan, 2008)	PSO+GA	Decision Tree (DT)	83.9
22.	(Zhang et al., 2009)	GE Hybrid	Decision Tree (DT)	83.41
23.	(Dash & Patra, 2012)	CFS + Wrapper (Random Trees)	K- Nearest Neighbour (KNN)	82.26
24.	(Zhang et al., 2009)	GE Hybrid	Random Forests	81.67
25.	(Ammu, Siva Kumar, & Mundayoor, 2014)	Information Gain (IG) + phase biogeography-based optimisation algorithm with immigration refusal	---	80.00
26.	(Yang, Zhou, et al., 2010)	MF-GE	3-Nearest Neighbour	77.01
27.	(Yang, Zhou, et al., 2010)	MF-GE	Decision Tree (DT)	76.64
28.	(Yang, Zhou, et al., 2010)	MF-GE	Naïve Bayes (NB)	75.07
29.	(Yang, Zhou, et al., 2010)	MF-GE	Random Forests (RF)	74.35
30.	(Yang, Zhou, et al., 2010)	MF-GE	7-Nearest Neighbour	68.78
31.	(El Akadi et al., 2009a)	mRMR	Naïve Bayes (NB)	66.13

Figure 12. ML Process
Source [Adapted from]: Mittal (2017). 'ML Process and Scenarios' [Image]. Retrieved from: https://elearningindustry.com/machine-learning-process-and-scenarios



An overview of the ML process is illustrated in Figure 12. It is a complex and challenging process that begins with the selection of working data from data providers. Because these data are unstructured in rows, they can contain redundant, duplicate and irrelevant data; they come from multiple sources. Thus, the second step is pre-processing, which prepares and structures the data. Next, it enters an iterative process that operates on the data to refine and clean the data, leaving the data structured and ready. Thereafter, the data will be available, and ML algorithms can be applied. The final step is the formulation of a candidate model (i.e., initial model). This step is iteratively processed to produce other candidate models, until it results in the best model. Finally, the best model is used by ML applications.

2.11.1.1 Applications of ML

ML depends mainly on computer science and statistical theory. Computer science plays two major tasks in ML: training via algorithms, which solve optimisation problems; and massive data manipulation. After learning the model, the second task ensures the provided solution is efficient. Thus, the algorithmic interpretation of efficiency and time complexity is as significant as the accuracy prediction in some applications.

This research focuses on feature selection and classification problems aiming to classify the colon cancer cells into a fixed set of categories/classes and to evaluate prediction performance. However, ML can be found and applied in a variety of examples (Kilany, 2013).

- In medical fields, medical and disease diagnostics of patients.

- In telecommunication fields, network optimisation to enhance services, spam filtering.
- In science fields, quick analysis of physical and biological data.
- In AI fields, face detection and speech recognition systems.
- In industrial and mechanical fields, optimisation and control product performance.
- In business and finance fields, financial data analysis that assists fraud dedication, stock market analysis and other financial applications.
- In weather forecasting, predicting weather situations.

2.11.1.2 ML Goal

The main goal and objective of ML is to design and develop efficient algorithms in time and space. In the context of ML, these algorithms should deal with a massive amount of data. Additionally, ML algorithms must be general purpose and employed to a variety of learning applications, ideas and problems (Section 2.7.1.1). They must also handle multiple datasets. Still, ML algorithms must produce results that are as accurate as possible, so they can be easily interpreted by experts.

2.11.2 Feature Selection for ML

Feature selection is a fundamental to ML. It searches through all possible combinations of attributes in the entire dataset. Then, it determines which attribute subset works better for prediction. Feature selection extracts the most relevant features for learning, enabling a learning algorithm that focuses on the most useful data for further analysis and selects an optimal subset for better classification.

2.11.2.1 Feature Selection Concept

It is found from Section 2.3 that microarray technologies produce enormous amounts of high-dimensional data in less time (e.g., a few days), compared to the examination of genes inside a biological lab (e.g., several months) (Babu & Sarkar, 2016). Therefore, disease diagnosis can be performed quicker, and suitable medications and treatments can be prescribed earlier. Discriminant analysis is important for cancer tissue distinction (i.e., normal or cancer). However, if informative genes can be selected at earlier stages, prediction and treatment can be more efficient.

As discussed in Section 2.3 and Section 2.5, the microarray data key problem involves a massive amount of data (e.g., genes and features) associated with the small number of samples. Usually, there exist very few samples, often less than 100 or 200 patients, in the microarray data, with reference to the number of genes/features generated from the samples (6000 to 60000) (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2012). This huge number of features can spur many types of data analyses; classification prediction becomes significantly difficult as a result, and data increasingly becomes scattered in the space of study. This phenomena is known as the 'curse of dimensionality' (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2012; Janecek, 2009). In this phenomenon, the massive number of genes increase data noise, causing the learning algorithm to demonstrate low performance, thereby increasing the possibility of errors. Figure 13 explains this phenomenon and shows that if the dimensionality increases, the performance decreases. Thus, computational costs will increase when dimensionality increases.

As stated in Section 2.6, there exist several approaches and methods in the literature to resolve this problem by investigating different ways to decrease data dimensionality or minimise the number of genes by selecting related genes. Thus, to break the 'curse of dimensionality', gene selection is applied for

DNA microarray analysis. Feature selection, known as gene subset selection, is the process applied to ML to settle the issue of data having high dimensionality. This is the procedure of searching and choosing a subset of highly correlated, important and informative genes while eliminating those redundant, irrelevant or noisy (Anaissi, Kennedy, & Goyal, 2011; Bolón-Canedo et al., 2014; Fiori, 2010; Hall, 1999; Karabulut, Özel, & Ibrikci, 2012; Saeys, Abeel, & Van de Peer, 2008). This process minimises the computational time of the classifier (i.e., ML algorithm) and enhances the prediction accuracy, because it influences those redundant, noisy and irrelevant data to be removed or discarded and it provides a better understanding of data. Furthermore, feature selection provides a deep understanding of diseases at the molecular level.

Feature selection can be defined as: 'the search for the best subset of *d* features which best contributes to a class discrimination, given a set of *f* features on *n* labelled samples.' Thus, the possible number of subsets is $\frac{f!}{d!(f-d)!}$, which is considered very massive and large. Feature selection is a major pre-processing step in microarray dataset analysis and plays a significant role in identifying the most informative genes that cause disease. Feature selection is always applied as an ML method of pre-processing to influence and enhance accuracy. Thus, feature selection provides data filtering for ML (Pradhananga, 2007). Figure 14 presents the feature selection process.

Figure 13. Relation between the performance of ML model and the dimensionality of feature space
Source [Adapted from]: Spruyt (2014). 'About the Curse of Dimensionality' [Image]. Retrieved from: https://www.datascience-central.com/profiles/blogs/about-the-curse-of-dimensionality

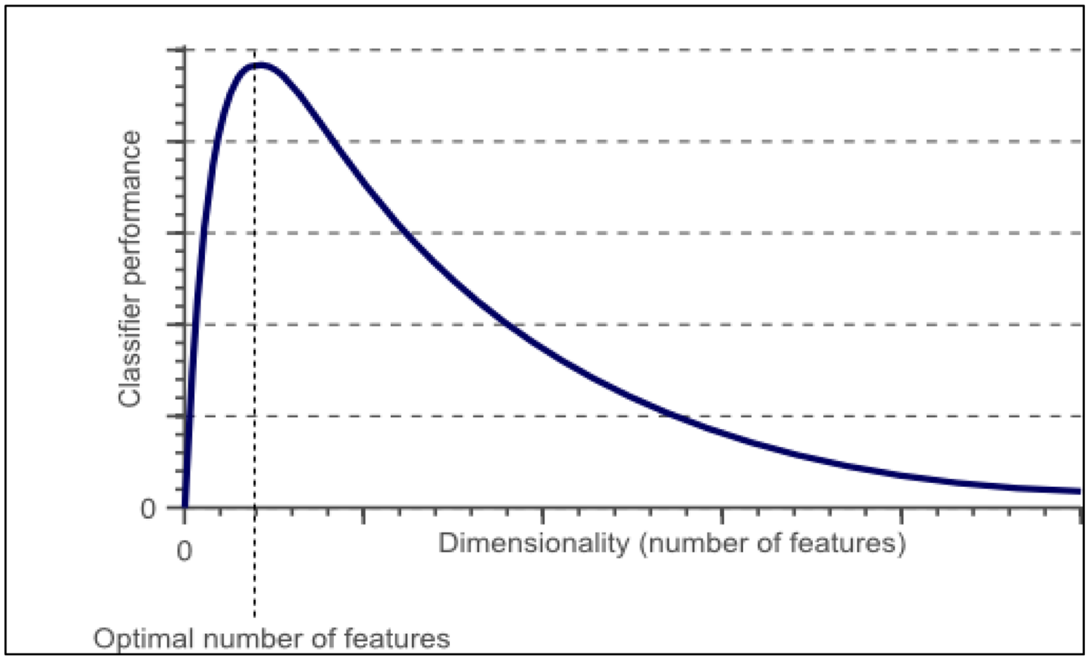
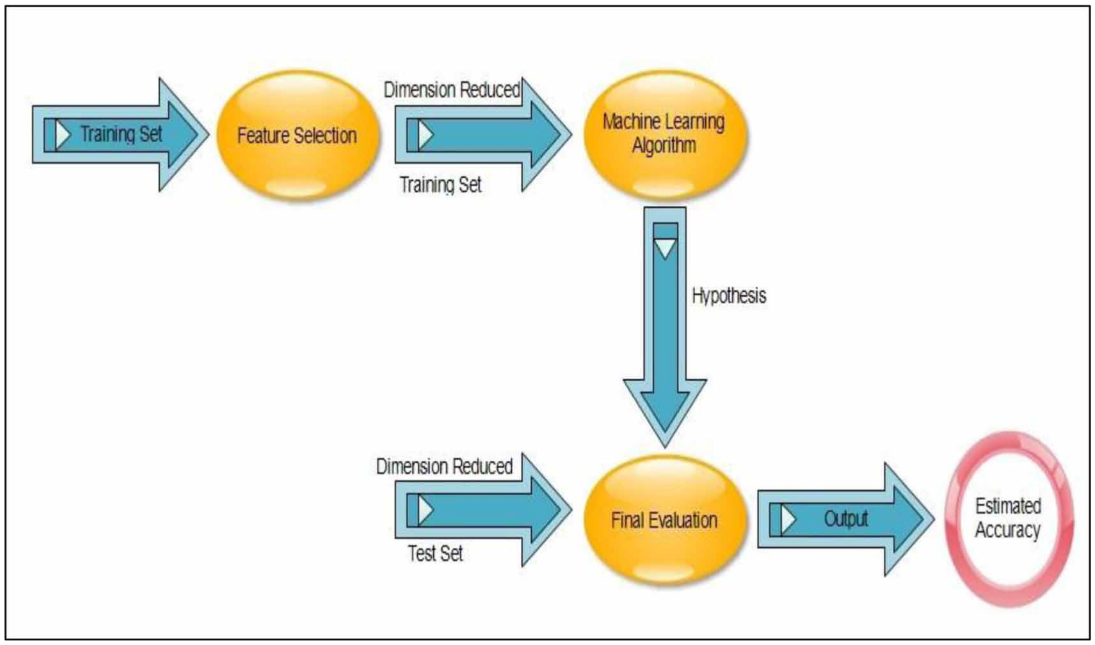


Figure 14. Feature selection process



2.11.3 Advantages of Feature Selection

Several manifold advantages of feature selection exist as ML pre-processing steps.

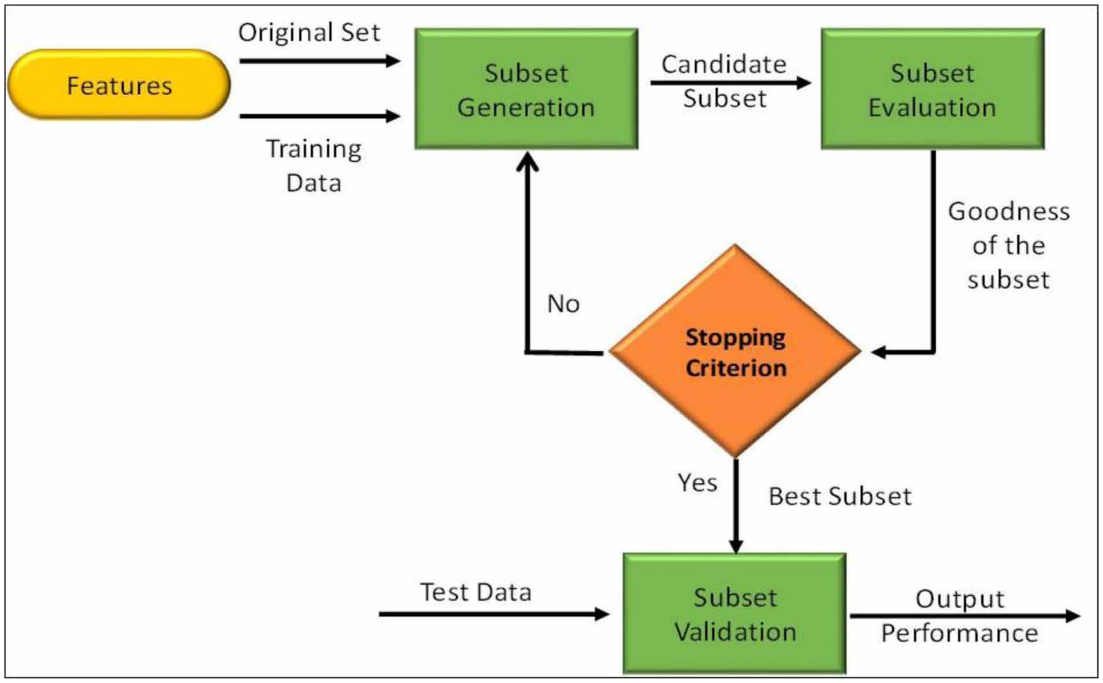
- Filtering out and limiting the number of genes for the ML algorithm enhances the classification accuracy and improves detection performance.
- Consuming a subset of genes is always better for data analysis than using the entire set of genes, because it removes and eliminates unwanted features.
- Resulting in less computationally intensive models, reducing the computation costs of training and prediction.
- Providing cheaper medical diagnosis costs, because feature selection focuses on a subset of genes rather than the entire dataset (i.e., thousands of genes).
- Resulting in easier and faster modelling for classification prediction.
- Helping biologists gain significant understanding of molecular level and genetic data to assist early disease diagnosis and drug discovery.
- Eliminating the curse-of-dimensionality phenomenon.

2.11.4 The Feature Selection Procedure

A distinctive feature selection procedure comprises four components, as illustrated by Figure 2-15, listed below (Zhu, Ong, & Dash, 2007).

1. **Subset generation** is an empirical search process that generates a candidate feature subset of the evaluation process. The origin of this step is identified by two elementary steps. The first step includes the starting point decision, which affects the searching direction. There exist three types of starting-points based on direction: *forward* considers an empty set to start with, then successfully adds features; *backward* considers a full set of data to start with and successfully eliminates genes and *bi-directional* considers both ends to start with and removes or adds features at the same time. Sometimes the search can start with a random subset. The second step includes the search strategy. There exist different search approaches, such as sequential, complete and random searches.
2. **Subset evaluation** evaluates each candidate feature in the subset and compares them with the best former one based on an evaluation measure. If the new candidate feature subset is evaluated better, it replaces the previous candidate. Thus, this step measures the goodness of features.
3. **Stopping criterion:** steps one and two are conditionally repeated until reaching a stopping criterion.
4. **Subset validation:** the new best subset generated must be validated by measuring the results using former information about the data or using different tests of real datasets.

Figure 15. Feature selection procedure
Source [Adapted from]: (Fahrudin, Syarif, & Barakbah, 2016; Zhu, Ong, & Dash, 2007)

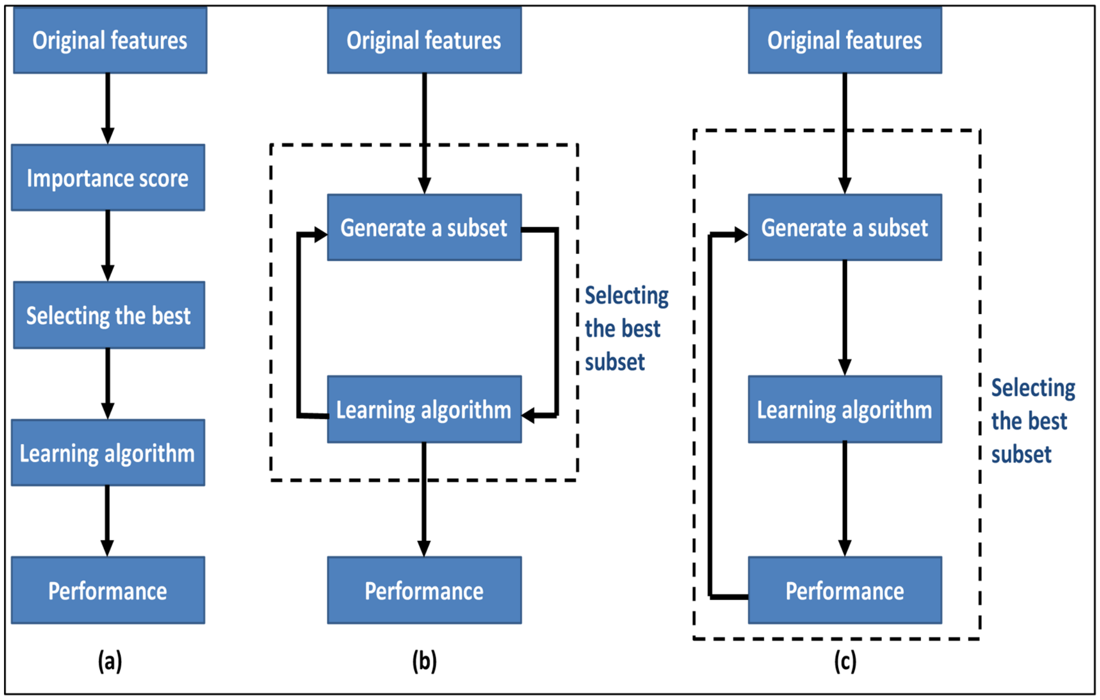


2.11.5 Feature Selection Categories (Models)

The gene/feature selection methods are normally grouped into three main categories, based on how the gene selection search method is involved with the structured model of the classifier: filter, wrapper and embedded methods (Saeys, Inza, & Larrañaga, 2007).

A filter method takes place when the selection method runs independently from the classifier (Figure 16 (a)), otherwise it is said to use a wrapper approach (Figure 16 (b)). Therefore, filter methods estimate the significance of genes/features by investigating only the main characteristics of the data. Generally, a gene-related score is measured, and the genes with low scores are eliminated. Thereafter, the gene subset is entered as an input into the classifier. Filtering methods have several advantages, because they are simple and computationally rapid. They can easily scale to very-high-dimensional datasets and are independent from the classifier. Thus, feature selection must be completed only once. Then, diverse classifiers can be estimated (Saeys, Inza, & Larrañaga, 2007). The segregation from the classification algorithm is the main filter approach disadvantage, which may lead to poor classification accuracy.

Figure 16. Categories of feature selection algorithms
Source [Adapted from]: (Wang et al., 2016)



Wrapper methods leverage gene selection and classification methods together using the same procedure, associating a learning algorithm to compute the classification accuracy (Alba et al., 2007). The computational expense is the main disadvantage of wrapper techniques. In some situations, gene selection is combined with the classifier design (e.g., recursive ridge or boosting design), known as an embedded technique (Hua, Tembe, & Dougherty, 2009). The advantage of embedded methods is their direct interaction with the classifier. Additionally, they are considered to have lower computation intensity when compared to wrapper methods (Figure 16 (c)).

Table 8 specifies a comparison between the three different categories of feature selection models in terms of the importance, advantages, disadvantages and other common problems. Both filter and wrapper methods have a significant impact in gene selection, but each has advantages and disadvantages (Al-

Rajab & Lu, 2014, 2016). The filter method is rapid, but it may result in lower classification accuracy, whereas the wrapper method has more time and considered more computationally complex. It could result in a higher accuracy output. Furthermore, embedded techniques incorporate the advantages of both filter and wrapper methods.

Table 8. Specifications of feature selection algorithms

Methods/ Technology involved	Importance	Advantages	Disadvantages	Problems
Filter Techniques	The importance of each gene is computed, and then genes that are top ranked will be selected	● Very fast ● Scales easily to data with very high dimension ● Independent of the classifier ● Less computational complexity ● Selection algorithms are to be implemented once only, then diverse ML algorithms can be assessed ● Time complexity is $O(n)$, which is considered low in comparison to other types.	● Univariate; where each feature is treated separately ● Disregards any relationship between features ● Ignores the interaction with the classifier ● Produce redundancy.	● Low classification accuracy
Wrapper Techniques	Select genes subset which are suitable to formulate a good classifier	● Simple and easy to implement in supervised learning. ● The correlation between features is being considered ● Ability to interact with the classifier ● Producing optimal solutions (good classification accuracy)	● Risk of overfitting ● It involves very thorough computation. ● Classifier is dependent on the selection. ● Time complexity is exponential in most cases. ● Slow	● Impracticable for high-dimensional data gene selection in ● Complexity is high
Embedded Techniques		● Ability to interact with the classifier ● Provides better computational complexity than wrappers ● Combines the benefits of filter and wrapper approaches.	● Classifier is dependent on the selection ● Time complexity may increase in some situations.	● It depends heavily on the model.

2.12 ML Classification Algorithms

This section probes the concept of feature classification in ML and the importance of applying classification to cancer diagnosis. The section describes the general steps of classification. Finally, it presents the main ML classification challenges and the different classification categories.

2.12.1 ML Classification Concept

Classification is a promising, important and widely studied topic by researchers in the fields of ML, statistics, pattern recognition and data mining, particularly in the healthcare sector. It assists scientists and other clinicians with better disease diagnoses and automated decision-making (Ang et al., 2016; Hong & Cho, 2006; Lu & Han, 2003; Patil & Sane, 2014; Porkodi & Suganya, 2015). This process is useful in cancer prediction and treatment. The main classification objective is to distinguish the data into categories of class labels. Classification can be defined formally as a given set of instances/samples

$F= \{f_1,f_2,f_3,\ldots,f_n\}\}$, each of which belongs to a finite set of classes $C= \{c_1,\ldots,c_n\}$, such that the classification will produce a function X which is going to map the elements of F to C as X: $F®C$. A wide range of algorithms are developed as solutions to the classification problems in the ML domain. They are applied to current studies of cancer gene expression detection and classification. These algorithms should ostensibly meet some standards (Kilany, 2013), such as collecting related data to provide high prediction accuracy. They should also be feasible in terms of computation, and they should provide results that can be easily interpretable.

2.12.2 General Classification Process

Classification is a process of predicting an outcome from a given input. The general classification process in ML is determining an effective and efficient model to predict class membership data from a set of samples. This process uses two main steps to predict an accurate outcome. First, the classification algorithm takes as input the training set, which contains a set of attributes, and produces an outcome hypothesis or goal. The algorithm learns the input and produces a model builder (i.e., learning model). The model built from the relationship between the attributes is expected to generate correct labels and to produce an accurate classifier. Next, the algorithm utilises the learning model to analyse the predicted accuracy and determine the percentage of correctly classified samples from the unseen test dataset (Kilany, 2013).

Generally, classification works by exploring the training dataset to figure out a collection of rules that specify the class of each datum according to its attributes. Thereafter, these rules are utilised to predict or evaluate the class or the missing attribute value from the hidden test dataset. It must not be seen during the training phase. They correctly distinguish the given training samples patterns and classify test samples using the trained classifier (Cho & Won, 2003; Elsheikh et al., 2011; Li, Zhang, & Ogihara, 2004). The group of samples in which the class labels are identified is known as the training set, whereas the test is defined as the set of samples in which the class labels are unseen or unknown to the algorithm, used to predict which samples belong to which class (Yeung & Bumgarner, 2005). The classification rules include the process of categorising each element of the population into one of several predefined classes. A classification process is supposed to be perfect, once each element is classified into the desired category or class it belongs to. This achieves high classification accuracy. This process is illustrated in Figure 17. Generally, the classification accuracy is computed by the percentage ratio of the correctly predicted samples over the test set via the function class trained across the training samples.

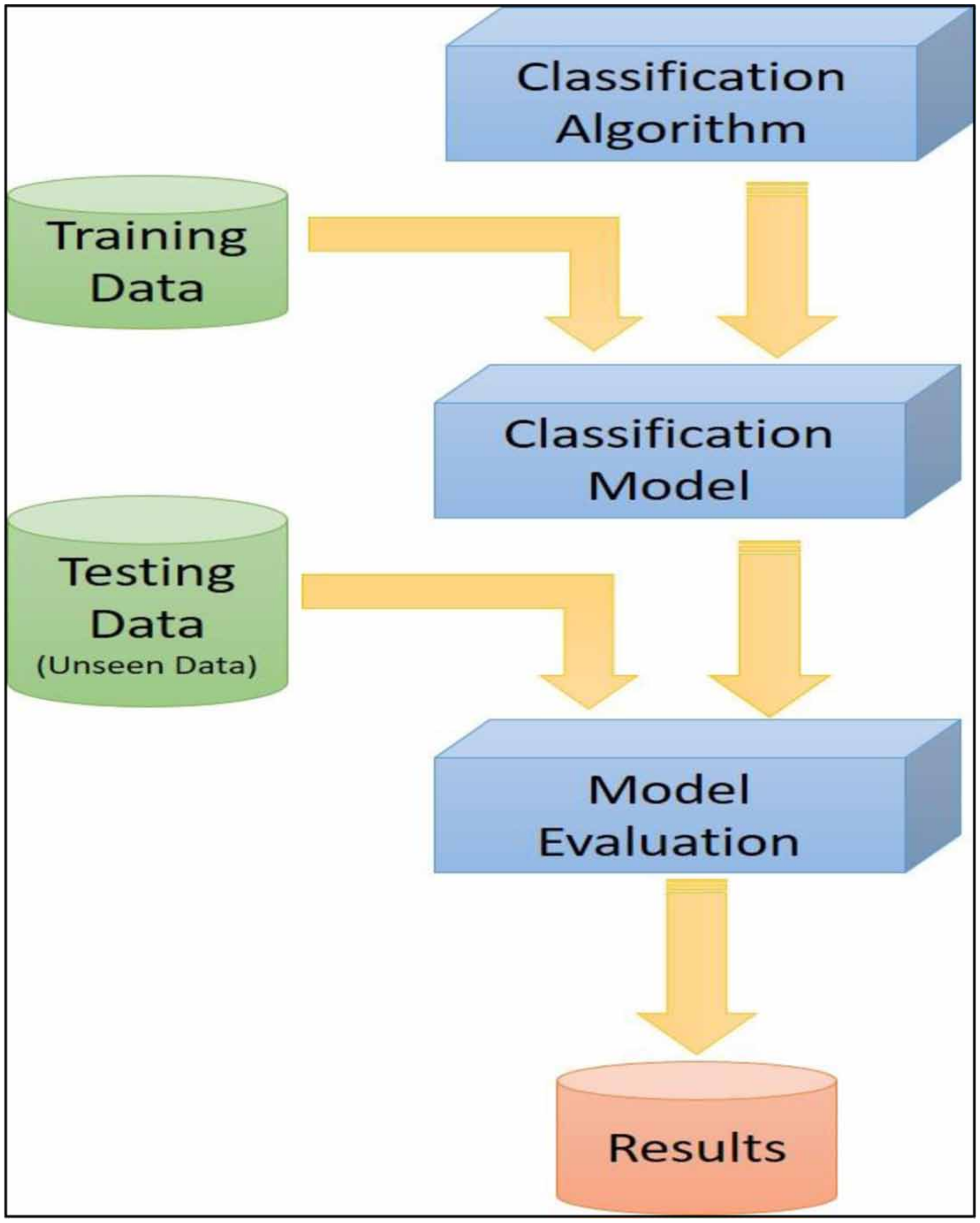
2.12.3 Challenges of Cancer Classification Methods

Microarray gene classification for cancer detection poses challenges in the area of ML because of problem uniqueness and characteristics (Babu & Sarkar, 2016; Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2010; Lu & Han, 2003; Salem, Attiya, & El-Fishawy, 2017). A list of these challenges is listed below.

- Very high dimensionality (thousands or hundreds-of-thousands of genes) compared to the small number of samples (less than 200) causes difficulties when estimating and reporting classification accuracy.
- The high dimensionality of features or attributes results from a small samples size or instances (less than 200 patients).

- There exist many irrelevant, uninformative and redundant genes in the dataset.
- Many genes are noisy, which increase computational complexity and negatively affects the classification model. This noise can be characterised as biological or technical.
- If the set used for training is relatively small compared to the testing set, it may cause the algorithm to become less likely to select the basic distribution of data.

Figure 17. General ML classification process

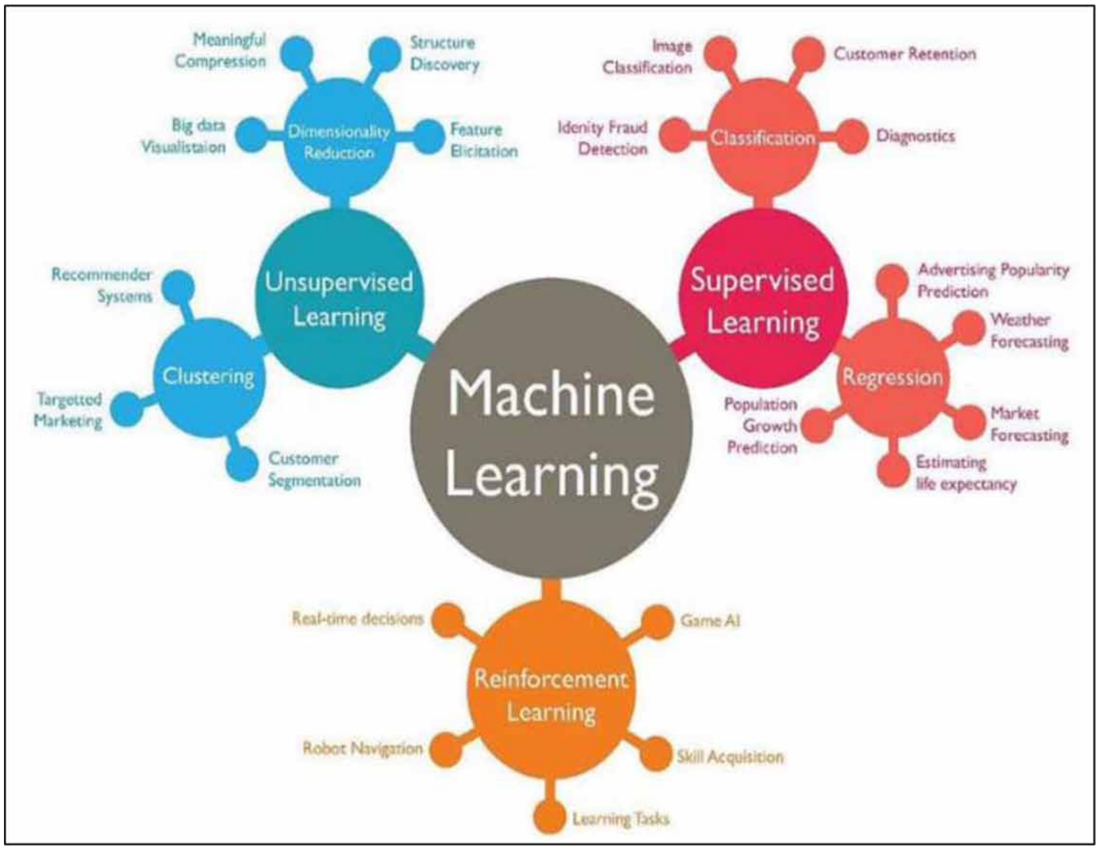


2.12.4 Categories of ML Algorithms for Classification

ML algorithms are grouped into diverse categories based on the algorithm's outcome. Figure 18 presents these diverse types. The focus of this research is on the categories that are related to cancer-gene selection and classification, as discussed below.

- Supervised learning (predictive): the algorithm produces a function that maps the input to the desired output (class label). In this category, the learner has previous information about the class label of each training datum that helps the learner learn the function and produce a predictive function. This type of classification is a useful data mining technique for solving medical diagnostic problems.
- Unsupervised learning (descriptive): this is opposite to supervised learning, and the learner has no previous information about the input or the desired output.

Figure 18. Types of ML algorithms
Source [Adapted from]: Granville (2017). 'Types of ML Algorithms in One Picture' [Image]. Retrieved from: " https://www.datasciencecentral.com/profiles/blogs/types-of-machine-learning-algorithms-in-one-picture



There exists other ML types, such as semi-supervised learning, transduction, reinforcement learning and learning to learn (Jumali et al., 2009). These types are outside the scope of this research and are not discussed here. Alshamlan, Badr, and Alohali (2013) and Dougherty, Kohavi, and Sahami (1995) argued that supervised methods are better than unsupervised ones for several reasons. In supervised methods, the classifier is trained before the classification process starts. However, in the unsupervised methods, the classification process starts without training. Additionally, the supervised methods are considered more effective in the domain of cancer classification.

The classification process is found in real-world applications. Some are used to classify modules that includes a two-class label problem (i.e., binary classification). Others contain more than two class labels (i.e., multi-class classification). As investigated by (Alshamlan, Badr, & Alohali, 2013), the problem of binary cancer classification has been extensively applied to colon, lung, prostate, leukaemia, ovarian and breast cancers.

In this research, the binary cancer classification is utilised, because the dataset applied is for the colon cancer microarray data. In summary, the objective of this research is categorised into gene finding (i.e., feature selection) and class prediction (i.e., classification). An illustration of the objective, approach and aim of these categories is presented in table 9.

Table 9. Objectives, approaches and aims for the microarray gene expression profile

Objective	Approach	Aim
Gene Finding	Feature Selection	Decrease the dimensionality of the microarray dataset by searching and selecting the related and informative genes.
Class Prediction	Classification	Classify the class labels (samples) and discriminate them in cancerous or normal cells

2.13 Efficiency of Algorithms and Time Complexity Analysis

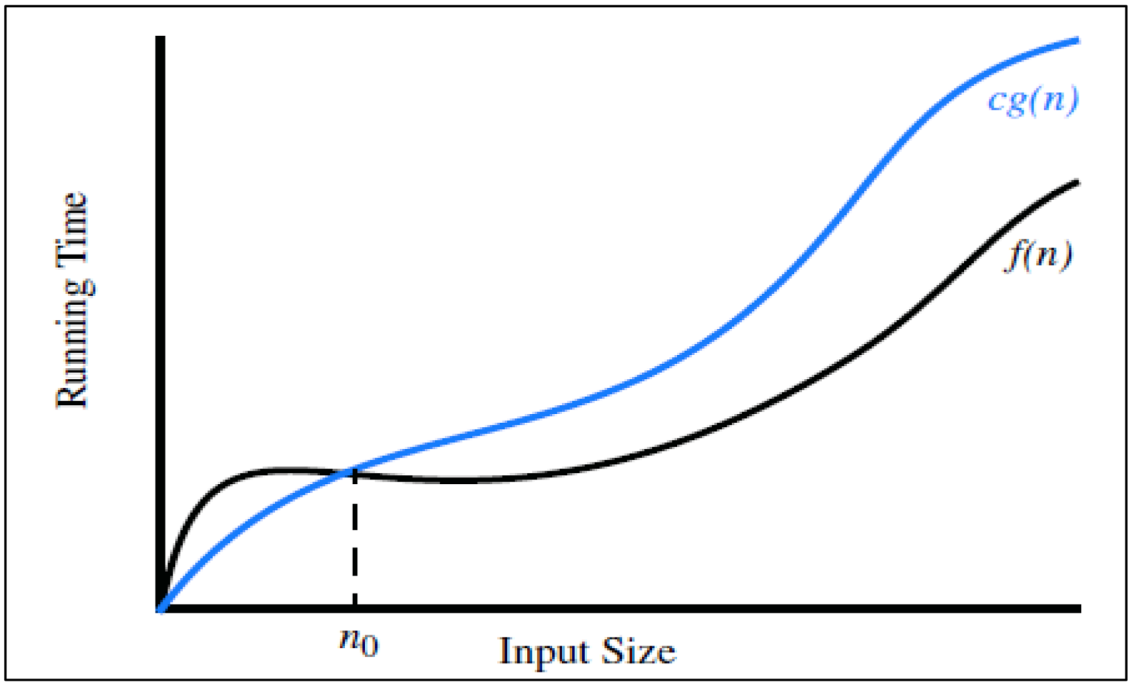
According to Jones and Pevzner (2004), the term *algorithm* is defined as an arrangement of instructions that must be performed very precisely to provide a solution for a particular problem. The problem must be specified in terms of input and output. Thus, the algorithm is the process or the procedure of translating inputs into outputs following a logical procedure. There can be more than one algorithm to solve the same problem, but we need to know which one is better. There exist many criteria to compare algorithms, such as speed, memory usage, correctness and usability. Algorithm analysis is a common approach of computer science that offers tools to examine the efficiency of different solutions methods (Goodrich, Tamassia, & Goldwasser, 2014). The efficiency or the performance of an algorithm can be expressed by the amount of memory space and time consumed. To analyse an algorithm, one must first understand how efficiently it solves a problem. The memory space is a machine-dependent criterion and cannot be considered in the case of this research study, because all experiments are conducted using the same data structures on the same machine. Thus, it uses a fixed amount of memory space. However, time complexity analysis is a composition theory of computational complexity that defines how computer resources can be used by an algorithm. That is, it determines how much time each basic operation is required for each input size value (Al-Rajab, Lu, & Xu, 2017). The *BigO* (big Omicron) notation is used to express the upper-bound growth rate of a function, which is the worst-case running time, expressed by the following notation.

$$O\left(g\left(n\right)\right) = [f\left(n\right) \mid \exists c \rangle 0, \exists n_0 > 0, \forall n \geq n_0 : 0 \leq \left|f\left(n\right)\right| \leq c\left|g\left(n\right)\right|] \tag{2-5}$$

In other words, $f \in O(g(n))$ if and only if there are positive constants, c and n_0, for all $n^3 n_0$. In that case, f is the big O of $g(n)$, or $g(n)$ is an asymptotic upper bound of f (Black, 2008*)*. Figure 19 illustrates the big O notation, and Figure 20 presents a set of the most common growth-rate functions. The function grows while the input size increases. The time used to establish the process model across the employed ML tool is considered the time computed and analysed across the experiments (See Section 4.3).

Figure 21. summarises the types of different growth rate complexity functions. Table 10 describes each type, whereas Table 11 demonstrates the time complexity (*BigO*) for most gene selection algorithms. Table 12 demonstrates the same for the gene cancer classification algorithms.

Figure 19. Big O notation



2.14 Problems Identified in Previous Studies

Whereas research on gene selection and cancer classification has achieved considerable development, there are still problems and limitations that need to be resolved. This section probes into those and highlights the foremost problems and limitations investigated from previous research, as stated in Section 2.10.

Figure 20. Common growth rate complexity functions
Source [Adapted from]: (Szaboj, 2015)

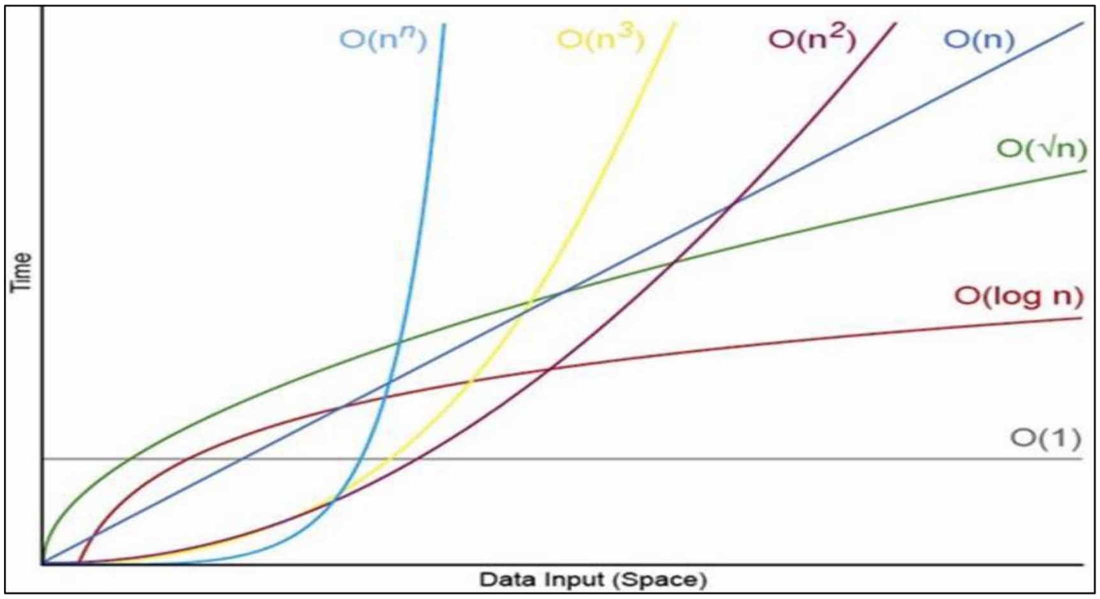


Figure 21. Types of growth-rate functions

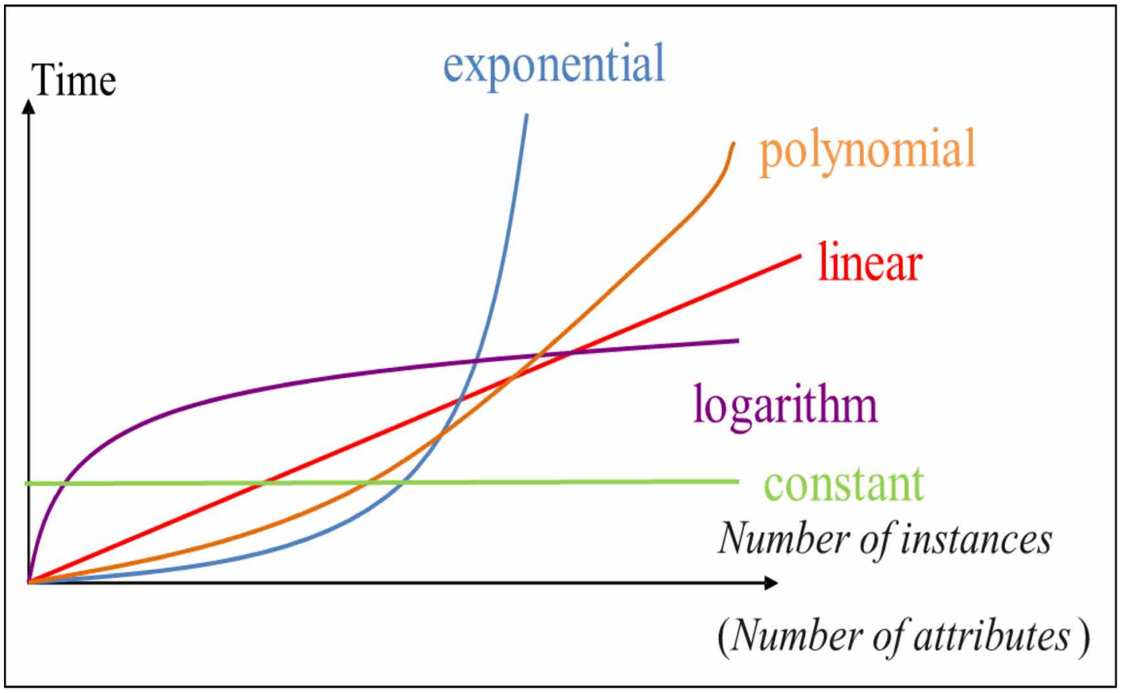


Table 10. Summary of Big (O) notation

Notation	Type	Description
$O(1)$	Constant	The notation continues to be constant nevertheless the input size.
$O(log\ n)$	Logarithmic	The notation increases in constant. If n doubles, the time grows in constant which is smaller than the quantity of n.
$O(n)$	Linear	The notation increases in proportion to n; if n doubles the time doubles.
$O(n^2)$	Polynomial (Quadratic)	The notation increases by the proportion of the product of $n \times n$.
$O(n^3)$	Polynomial (Cubic)	The notation increases by the proportion of the product of $n \times n \times n$.
$O(c^n)$	Exponential	The notation increases depending on the exponent n of a constant, c.

Source [Adapted from]: (Al-Rajab, Lu, & Xu, 2017)

Table 11. Gene selection algorithms time complexity function

Algorithm	Reference	Type	Time Complexity Notation
Genetic Algorithm	(Tsai et al., 2014; Tseng & Yang, 2001)	Polynomial	$O(n^2)$ or $O(gens \times n \times m)$, where *gens* is the generation, n represents the size of population and m represent the individual size.
Particle Swarm Optimisation	(Lakshmana, Botella, & Svensson, 2012)	Polynomial	$O(m \times n)$; here, m expresses initial number of particles and n expresses the number of iterations.
Information Gain	(Yang, Chuang, & Yang, 2010)	Logarithmic	$O(n \times log\ n)$; here, n represents the number of samples.
minimum Redundancy Maximum Relevance	(Ammu & Preeja, 2013; Rodriguez-Lujan et al., 2010)	Polynomial	$O(nm^2)$, where n represents the remaining set size and m is the number of variables.

Source [Adapted from]: (Al-Rajab, Lu, & Xu, 2017)

Table 12. Feature classification algorithms time complexity

Algorithm	Reference	Type	Time Complexity Notation
Support Vector Machine	(Abdiansah & Wardoyo, 2015; Tsang, Kwok, & Cheung, 2005)	Polynomial (Cubic)	$O(n^3)$; here, n is the number of a classical SVM training points.
Naïve Bayes	(Webb, Boughton, & Wang, 2005)	Polynomial	$O(m \times n)$; here, m is the number of samples and n is the number of features or attributes.
Decision Tree	(Su & Zhang, 2006; Sumam, Sudheep, & Joseph, 2013)	Polynomial	$O(m \times n^2)$; here, m is the number of training data and n is the number or attributes.
Genetic Programming	(Kötzing, Neumann, & Spöhel, 2011; Urli, Wagner, & Neumann, 2012)	Logarithmic	$O(n \times log\ n)$; here, n is the number of samples.
K-Nearest Neighbour	(Ammu & Preeja, 2013)	Linear	$O(nm)$; where n is the number of features or attributes and m is the number of samples

Source [Adapted from]: (Al-Rajab, Lu, & Xu, 2017)

2.14.1 High-Performance Feature Selection and Classification Algorithms Issues

Many studies have stated that early cancer diagnoses can increase the chance of survival. For colon cancer, regular screenings play a significant role in preventing disease. Doctors can apply surgical treatments for that purpose when necessary. Doctors can also use chemotherapy or radiation to reduce the cancer or stop the growth of cells. However, some studies have proven that colon cancer can be detected early and prevented when the cancerous polyps can be well-identified (Cotilho, 2011; Khan, 2009).

There are remarkable discrepancies between current and previous studies (Section 2.6.1) on the topic of colon cancer selection and classification across the performance of classification accuracy (Al-Rajab, Lu, & Xu, 2017). The limitations of existing works reflect the advantage of the current study, are expressed in the following points.

- Table 6 (Section 2.10) presents 31 diverse gene selection algorithms and different cancer classification algorithms. 13 exhibited an accuracy of 90% or greater, utilising different tools when implemented over the desired colon dataset. The rest attained a classification accuracy between 69 and 89%.
- Some proposed systems, such as that of (Huerta, Duval, & Hao, 2006) applied GA+SVM with an accuracy of 99%. Mohamad, Deris, and Illias (2005) applied GA+SVM with an accuracy of 98%. Mohamad et al. (2010) applied GA+SVM with an accuracy of 99%. Mohamad, Omatu, Deris, Misman, et al. (2009) applied GA+SVM with an accuracy of 99.5%. Tan and Gilbert (2003) applied DT with an accuracy of 95%. Porkodi and Suganya (2015) applied RF with an accuracy of 95% and an NB of 91%. Sujatha and Rani (2013) applied DT with an accuracy of 95%. All these studies achieved high classification accuracy results of 95% or above using the colon cancer dataset. However, these results were not produced from an independent test/validating set. Their results came from the same dataset used for training the classifier. They employed the same dataset to evaluate the results of the training set, which was the same dataset. Thus, the procedure of their experiments was limited and biased. Thus, classification accuracy alone is not sufficient to measure algorithm performance.
- Table 13 shows that, by using the same dataset and algorithms or methods, different accuracy outputs are obtained. See Table 6 (Section 2.10). This also implies that classification accuracy alone is not sufficient to measure the performance of algorithms.
- To the best of knowledge, no previous studies have examined the direct relationship between the accuracy results of algorithms and the analysis of the time taken to perform gene selection and cancer classification, except for one study by Salem, Attiya, and El-Fishawy (2017), who mentioned the time complexity of his proposed system, but without a comparative relation to other systems.

2.14.2 Hybrid Feature Selection Issues

There are remarkable discrepancies between the current study and previous studies (Section 2.10) on the topic of hybrid and multifilter selection algorithms. The limitations of this work are elaborated by following points.

Table 13. Summary highlights from literature review

No.	Method		References	Accuracy [%]
	Feature Selection	Classifier		
1.	IG	SVM	(Osareh & Shadgar, 2010)	99.90
			(Fang, Mustapha, & Sulaiman, 2010)	93.55
			(Fang, Mustapha, & Sulaiman, 2010)	90.33
			(Cho & Won, 2003)	71.00
2.	GA	SVM	(Huerta, Duval, & Hao, 2006)	99.41
			(Mohamad, Deris, & Illias, 2005)	98.39
			(Li, Wu, & Hu, 2008)	93.60
			(Mohamad, Deris, & Illias, 2005)	90.32
			(Mohamad, Deris, & Illias, 2005)	85.48
3.	GA	DT	(Yang & Zhang, 2007)	96.79
			(Yeh et al., 2007)	89.24
			(Yeh et al., 2007)	88.8
4.	t-Statistics	SVM	(Osareh & Shadgar, 2010)	99.90
			(Alladi et al., 2008)	84.09
5.	t-Statistics	DT	(Kulkarni et al., 2011)	85.00
			(Yeh et al., 2007)	77.42

- Wrapper methods produce high classification accuracy, but have very high computational complexity, compared with filter approaches, which are categorised as fast but less accurate.
- Some approaches in the literature have applied the hybridisation algorithms for gene selection and cancer classification. To the best of knowledge, no one in the literature has touched this problem using a two-stage hybrid multifilter selection method.
- From Table 7 (Section 2.10), the contribution of GA, PSO and mRMR algorithms as pre-selection methods has achieved very good performance results, approaching 90% or above when applied in a hybrid fashion.
- From Table 7 (Section 2.10), those who achieved high classification accuracy (above 90%) such as Li, Wu, and Tan (2008), great accuracy was achieved when only the top 10 genes were selected. In Abdi, Hosseini, and Rezghi (2012), achieved great accuracy when the mRMR was used to select only the top 50 genes. Mohamad, Deris, and Illias (2005), identified different methods, but when they applied the hybrid method of selection on all dataset genes using the SVM, they arrived at a 90.32% accuracy. Thus, no one has employed the selection across the entire genetic dataset.

REFERENCES

Abdi, M. J., Hosseini, S. M., & Rezghi, M. (2012). A Novel Weighted Support Vector Machine Based on Particle Swarm Optimization for Gene Selection and Tumor Classification. *Computational and Mathematical Methods in Medicine*, *2012*, 1–7. doi:10.1155/2012/320698 PMID:22924059

Abdiansah, A., & Wardoyo, R. (2015). Time Complexity Analysis of Support Vector Machines (Svm) in Libsvm. *International Journal of Computers and Applications*, *128*(3), 28–34. doi:10.5120/ijca2015906480

Ackovska, N., & Madevska-Bogdanova, A. (2005). *Teaching Bioinformatics to Computer Science Students*. Paper presented at the International Conference on Computer as a Tool, Belgrade, Serbia. 10.1109/EURCON.2005.1630056

Akbar, B., Gopi, V. P., & Babu, V. S. (2015). *Colon Cancer Detection Based on Structural and Statistical Pattern Recognition*. Paper presented at the 2nd International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India. 10.1109/ECS.2015.7124883

Al-Rajab, M., & Lu, J. (2012). *Bioinformatics: An Overview for Cancer Research*. Paper presented at the 2012 World Congress in Computer Science, Computer Engineering & Applied Computing, Las Vegas, NV.

Al-Rajab, M., Lu, J., & Xu, Q. i. (2017). Examining Applying High Performance Genetic Data Feature Selection and Classification Algorithms for Colon Cancer Diagnosis. *Computer Methods and Programs in Biomedicine*, *146*, 11–24. doi:10.1016/j.cmpb.2017.05.001 PMID:28688481

Al-Rajab, M. M., & Lu, J. (2014). *Algorithms Implemented for Cancer Gene Searching and Classifications*. Paper presented at the International Symposium on Bioinformatics Research and Applications, Zhangjiajie, China. 10.1007/978-3-319-08171-7_6

Al-Rajab, M. M., & Lu, J. (2016). A Study on the Most Common Algorithms Implemented for Cancer Gene Search and Classifications. *International Journal of Data Mining and Bioinformatics*, *14*(2), 159–176. doi:10.1504/IJDMB.2016.074685

Al Snousy, M. B., El-Deeb, H. M., Badran, K., & Al Khlil, I. A. (2011). Suite of Decision Tree-Based Classification Algorithms on Cancer Gene Expression Data. *Egyptian Informatics Journal*, *12*(2), 73–82. doi:10.1016/j.eij.2011.04.003

Alba, E., Garcia-Nieto, J., Jourdan, L., & Talbi, E.-G. (2007). *Gene Selection in Cancer Classification Using Pso/Svm and Ga/Svm Hybrid Algorithms*. Paper presented at the IEEE Congress on Evolutionary Computation (CEC), Singapore, Singapore. 10.1109/CEC.2007.4424483

Ali, S., & Gupta, P. (2006). Classification and Rule Generation for Colon Tumor Gene Expression Data. *Proceedings of the 2006 Information Resources Management Association Conference in Emerging Trends and Challenges in Information Technology Management*, 281-284

Alladi, S. M., Shinde Santosh, P., Ravi, V., & Murthy, U. S. (2008). Colon Cancer Prediction with Genetic Profiles Using Intelligent Techniques. *Bioinformation*, *3*(3), 130–133. doi:10.6026/97320630003130 PMID:19238250

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., & Levine, A. J. (1999). Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. *Proceedings of the National Academy of Sciences of the United States of America*, *96*(12), 6745–6750. doi:10.1073/pnas.96.12.6745 PMID:10359783

Alshamlan, H., Badr, G., & Alohali, Y. (2015). Mrmr-Abc: A Hybrid Gene Selection Algorithm for Cancer Classification Using Microarray Gene Expression Profiling. *BioMed Research International*, *2015*, 1–15. doi:10.1155/2015/604910 PMID:25961028

Alshamlan, H. M., Badr, G. H., & Alohali, Y. (2013). A Study of Cancer Microarray Gene Expression Profile: Objectives and Approaches. *Proceedings of the World Congress on Engineering*, 1-6.

Ammu, P., & Preeja, V. (2013). Review on Feature Selection Techniques of DNA Microarray Data. *International Journal of Computers and Applications*, *61*(12), 39–44. doi:10.5120/9983-4814

Ammu, P., Siva Kumar, K., & Mundayoor, S. (2014). A Bbo Based Feature Selection Method for DNA Microarray. *International Journal of Research Studies in Biosciences*, *3*(1), 201–204.

Anaissi, A., Kennedy, P. J., & Goyal, M. (2011). *Feature Selection of Imbalanced Gene Expression Microarray Data*. Paper presented at the 12th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Sydney, NSW, Australia. 10.1109/SNPD.2011.12

Ang, J. C., Mirzal, A., Haron, H., & Hamed, H. N. A. (2016). Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *13*(5), 971–989. doi:10.1109/TCBB.2015.2478454 PMID:26390495

Archetti, F., Castelli, M., Giordani, I., & Vanneschi, L. (2008). Classification of Colon Tumor Tissues Using Genetic Programming. *Proceedings of the Annual Italian Workshop on Artificial Life and Evolutionary Computation*, 49-58. 10.1142/9789814287456_0004

Babu, M., & Sarkar, K. (2016). *A Comparative Study of Gene Selection Methods for Cancer Classification Using Microarray Data*. Paper presented at the Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN). 10.1109/ICRCICN.2016.7813657

Banerjee, M., Mitra, S., & Banka, H. (2007). Evolutionary Rough Feature Selection in Gene Expression Data. *IEEE Transactions on Systems, Man and Cybernetics. Part C, Applications and Reviews*, *37*(4), 622–632. doi:10.1109/TSMCC.2007.897498

Bayat, A. (2002). Science, Medicine, and the Future: Bioinformatics. *British Medical Journal*, *324*(7344), 1018–1022. doi:10.1136/bmj.324.7344.1018 PMID:11976246

Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., & Yakhini, Z. (2000). Tissue Classification with Gene Expression Profiles. *Journal of Computational Biology*, *7*(3-4), 559–583. doi:10.1089/106652700750050943 PMID:11108479

Bennet, J., Ganaprakasam, C., & Kumar, N. (2015). A Hybrid Approach for Gene Selection and Classification Using Support Vector Machine. *The International Arab Journal of Information Technology*, *12*(6A), 695–700.

Black, P. E. (Ed.). (2008). *Dictionary of Algorithms and Data Structures. National Institute of Standards and Technology (NISTIR)*. US Department of Commerce.

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2010). *On the Effectiveness of Discretization on Gene Selection of Microarray Data*. Paper presented at the 2010 International Joint Conference on Neural networks (IGCNN), Barcelona, Spain. 10.1109/IJCNN.2010.5596825

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2012). An Ensemble of Filters and Classifiers for Microarray Data Classification. *Pattern Recognition*, *45*(1), 531–539. doi:10.1016/j.patcog.2011.06.006

Bolón-Canedo, V., Sánchez-Marono, N., Alonso-Betanzos, A., Benítez, J. M., & Herrera, F. (2014). A Review of Microarray Datasets and Applied Feature Selection Methods. *Information Sciences*, *282*, 111–135. doi:10.1016/j.ins.2014.05.042

Bonilla-Huerta, E., Hernández-Montiel, A., Morales-Caporal, R., & Arjona-López, M. (2016). Hybrid Framework Using Multiple-Filters and an Embedded Approach for an Efficient Selection and Classification of Microarray Data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *13*(1), 12–26. doi:10.1109/TCBB.2015.2474384 PMID:26336138

Chavan, P. R. (2008). *Application of Bioinformatics in the Field of Cancer Research.* Paper presented at the 11th Workshop on Medical Informatics & CME on Biomedical Communication.

Chen, X.-w. (2003). Gene Selection for Cancer Classification Using Bootstrapped Genetic Algorithms and Support Vector Machines. *Proceedings of the 2003 IEEE Bioinformatics Conference (CSB2003)*, 504-505. 10.1109/CSB.2003.1227389

Chitode, K., & Nagori, M. (2013). A Comparative Study of Microarray Data Analysis for Cancer Classification. *International Journal of Computers and Applications*, *81*(15), 14–18. doi:10.5120/14198-2392

Cho, S.-B., & Ryu, J. (2002). Classifying Gene Expression Data of Cancer Using Classifier Ensemble with Mutually Exclusive Features. *Proceedings of the IEEE*, *90*(11), 1744–1753. doi:10.1109/JPROC.2002.804682

Cho, S.-B., & Won, H.-H. (2003). Machine Learning in DNA Microarray Analysis for Cancer Classification. *Proceedings of the First Asia-Pacific Bioinformatics Conference on Bioinformatics*, 189-198.

Chuang, L.-Y., Ke, C.-H., Chang, H.-W., & Yang, C.-H. (2009). A Two-Stage Feature Selection Method for Gene Expression Data. *OMICS: A Journal of Integrative Biology*, *13*(2), 127–137. doi:10.1089/omi.2008.0083 PMID:19182978

Cohen, J. (2005). Computer Science and Bioinformatics. *Communications of the ACM*, *48*(3), 72–78. doi:10.1145/1047671.1047672

Cotilho, R. A. F. (2011). *Detection and Classification of Human Colorectal Polyps: An Analysis of Image Curvature and Edge Detection in Wce Images* (MSc Dissertation). Technical University of Lisbon.

Cueto-López, N., García-Ordás, M. T., Dávila-Batista, V., Moreno, V., Aragonés, N., & Alaiz-Rodríguez, R. (2019). A comparative study on feature selection for a risk prediction model for colorectal cancer. *Computer Methods and Programs in Biomedicine*, *177*, 219–229. doi:10.1016/j.cmpb.2019.06.001 PMID:31319951

Dang, V. Q. (2014). *Evolutionary Approaches for Feature Selection in Biological Data* (PhD Thesis), Edith Cowan University, Perth, Western Australia.

Dash, S., & Patra, B. (2012). Study of Classification Accuracy of Microarray Data for Cancer Classification Using Hybrid, Wrapper and Filter Feature Selection Method. *Proceedings of the International Conference on Bioinformatics & Computational Biology (BIOCOMP)*, 268-275.

Dash, S., Patra, B., & Tripathy, B. (2012). Study of Classification Accuracy of Microarray Data for Cancer Classification Using Multivariate and Hybrid Feature Selection Method. *IOSR Journal of Engineering*, *2*(8), 112–119. doi:10.9790/3021-0281112119

Domokos, A. D. (2008). Bioinformatics and Computational Biology. *Bulletin of University of Agricultural Sciences and Veterinary Medicine. Horticulture*, *65*(2), 571–574. doi:10.15835/buasvmcn-hort:517

Doom, T., Raymer, M., Krane, D., & Garcia, O. (2003). Crossing the Interdisciplinary Barrier: A Baccalaureate Computer Science Option in Bioinformatics. *IEEE Transactions on Education*, *46*(3), 387–393. doi:10.1109/TE.2003.814593

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the Twelfth International Conference on Machine Learning*. 10.1016/B978-1-55860-377-6.50032-3

El Akadi, A., Amine, A., El Ouardighi, A., & Aboutajdine, D. (2009a). Feature Selection for Genomic Data by Combining Filter and Wrapper Approaches. *INFOCOMP Journal of Computer Science*, *8*(4), 28–36.

El Akadi, A., Amine, A., El Ouardighi, A., & Aboutajdine, D. (2009b). *A New Gene Selection Approach Based on Minimum Redundancy-Maximum Relevance (Mrmr) and Genetic Algorithm (Ga)*. Paper presented at the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2009), Rabat, Morocco. 10.1109/AICCSA.2009.5069306

El Akadi, A., Amine, A., El Ouardighi, A., & Aboutajdine, D. (2011). A Two-Stage Gene Selection Scheme Utilizing Mrmr Filter and Ga Wrapper. *Knowledge and Information Systems*, *26*(3), 487–500. doi:10.100710115-010-0288-x

Elsheikh, A., Jarada, T., Nagi, M., Naji, G., Karampelas, P., Sair, O., Peng, P., Kianmehr, K., Özyer, T., Ridley, M., Rokne, J., & Alhajj, R. (2011). *Effectiveness of Feature Selection and Classification Techniques for Gene Expression Data Analysis*. Paper presented at the 5th International Conference on Information Technology (ICIT), Amman, Jordan.

Elyasigomari, V., Lee, D., Screen, H. R., & Shaheed, M. H. (2017). Development of a Two-Stage Gene Selection Method That Incorporates a Novel Hybrid Approach Using the Cuckoo Optimization Algorithm and Harmony Search for Cancer Classification. *Journal of Biomedical Informatics*, *67*, 11–20. doi:10.1016/j.jbi.2017.01.016 PMID:28163197

Fahrudin, T. M., Syarif, I., & Barakbah, A. R. (2016). *Ant Colony Algorithm for Feature Selection on Microarray Datasets*. Paper presented at the International Electronics Symposium (IES), Denpasar, Indonesia. 10.1109/ELECSYM.2016.7861030

Fang, O. H., Mustapha, N., & Sulaiman, M. N. (2010). *Integrating Biological Information for Feature Selection in Microarray Data Classification*. Paper presented at the Second International Conference on Computer Engineering and Applications (ICCEA), Bali Island, Indonesia. 10.1109/ICCEA.2010.215

Fayyad, U., & Irani, K. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, 1022-1027.

Fenstermacher, D. (2005). Introduction to Bioinformatics: Research Articles. *Journal of the American Society for Information Science and Technology - Bioinformatics, 56*(5), 440-446. doi:10.1002/asi.v56:5

Fiori, A. (2010). *Extraction of Biological Knowledge by Means of Data Mining Techniques* (PhD Thesis). Polytechnic of Turin.

Fujarewicz, K., & Wiench, M. (2003). Selecting Differentially Expressed Genes for Colon Tumor Classification. *International Journal of Applied Mathematics and Computer Science*, *13*, 327–335.

Fulekar, M., & Sharma, J. (2008). Bioinformatics Applied in Bioremediation. *Innovative Romanian Food Biotechnology*, *2*, 28–36.

Gao, S., Addam, O., Qabaja, A., ElSheikh, A., Zarour, O., Nagi, M., Triant, F., Almansoori, W., Özyer, O. Ş. T., & Zeng, J. (2012). *Robust Integrated Framework for Effective Feature Selection and Sample Classification and Its Application to Gene Expression Data Analysis*. Paper presented at the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), San Diego, CA. doi: 10.1109/CIBCB.2012.6217219

García-Nieto, J., Alba, E., Jourdan, L., & Talbi, E. (2009). Sensitivity and Specificity Based Multiobjective Approach for Feature Selection: Application to Cancer Diagnosis. *Information Processing Letters*, *109*(16), 887–896. doi:10.1016/j.ipl.2009.03.029

GB-HealthWatch. (2018). *Genetics*, *101*. https://www.gbhealthwatch.com/Trait-Genetics-101.php

George, G., & Raj, V. C. (2011). Review on Feature Selection Techniques and the Impact of Svm for Cancer Classification Using Gene Expression Profile. *International Journal of Computer Science & Engineering Survey*, *2*(3), 16–26. doi:10.5121/ijcses.2011.2302

Ghosh, M., Begum, S., Sarkar, R., Chakraborty, D., & Maulik, U. (2019). Recursive Memetic Algorithm for gene selection in microarray data. *Expert Systems with Applications, 116*, 172-185. doi:10.1016/j.eswa.2018.06.057

Goldin, L. (2006). *Bioinformatics Integration for Cancer Research-Goal Question Analysis*. Paper presented at the International Conference on Information Technology: Research and Education (ITRE'06), Tel-Aviv, Israel. 10.1109/ITRE.2006.381575

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data Structures & Algorithms in Java*. John Wiley and Sons, Inc.

Granville, V. (2017). *Types of Ml Algorithms in One Picture*. Retrieved from: https://www.datasciencecentral.com/profiles/blogs/types-of-machine-learning-algorithms-in-one-picture

Guo, C., & Wang, J. (2021). Novel artificial intelligence machine learning approaches to precisely predict survival and site-specific recurrence in cervical cancer: A multi-institutional study. *Translational Oncology*, ●●●, 14.

Hall, M. A. (1999). *Correlation-Based Feature Selection for Machine Learning* (PhD Thesis). The University of Waikato, Hamilton, New Zealand.

Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Elsevier.

Hassan, A. R., & Haque, M. A. (2015). Computer-Aided Gastrointestinal Hemorrhage Detection in Wireless Capsule Endoscopy Videos. *Computer Methods and Programs in Biomedicine*, *122*(3), 341–353. doi:10.1016/j.cmpb.2015.09.005 PMID:26390947

Hassan, A. R., Siuly, S., & Zhang, Y. (2016). Epileptic Seizure Detection in Eeg Signals Using Tunable-Q Factor Wavelet Transform and Bootstrap Aggregating. *Computer Methods and Programs in Biomedicine*, *137*, 247–259. doi:10.1016/j.cmpb.2016.09.008 PMID:28110729

Hassan, A. R., & Subasi, A. (2016). Automatic Identification of Epileptic Seizures from Eeg Signals Using Linear Programming Boosting. *Computer Methods and Programs in Biomedicine*, *136*, 65–77. doi:10.1016/j.cmpb.2016.08.013 PMID:27686704

Hengpraprohm, S., & Chongstitvatana, P. (2007). *Selecting Informative Genes from Microarray Data for Cancer Classification with Genetic Programming Classifier Using K-Means Clustering and Snr Ranking*. Paper presented at the Frontiers in the Convergence of Bioscience and Information Technologies, 11-13 Oct. 2007, Jeju City, South Korea. 10.1109/FBIT.2007.84

Hong, J.-H., & Cho, S.-B. (2006). The Classification of Cancer Based on DNA Microarray Data That Uses Diverse Ensemble Genetic Programming. *Artificial Intelligence in Medicine*, *36*(1), 43–58. doi:10.1016/j.artmed.2005.06.002 PMID:16102956

Hoque, N., Bhattacharyya, D., & Kalita, J. K. (2014). Mifs-Nd: A Mutual Information-Based Feature Selection Method. *Expert Systems with Applications*, *41*(14), 6371–6385. doi:10.1016/j.eswa.2014.04.019

Horng, J.-T., Wu, L.-C., Liu, B.-J., Kuo, J.-L., Kuo, W.-H., & Zhang, J.-J. (2009). An Expert System to Classify Microarray Gene Expression Data Using Gene Selection by Decision Tree. *Expert Systems with Applications*, *36*(5), 9072–9081. doi:10.1016/j.eswa.2008.12.037

Hsieh, S.-Y., & Chou, Y.-C. (2016). A Faster Cdna Microarray Gene Expression Data Classifier for Diagnosing Diseases. [TCBB]. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *13*(1), 43–54. doi:10.1109/TCBB.2015.2474389 PMID:26336139

Hu, H., Li, J., Wang, H., Daggard, G., & Shi, M. (2006). A Maximally Diversified Multiple Decision Tree Algorithm for Microarray Data Classification. *Proceedings of the 2006 Workshop on Intelligent Systems for Bioinformatics*, 35-38.

Hu, Y. (2010). *Personalised Modelling Framework and Systems for Gene Data Analysis and Biomedical Applications* (PhD Thesis). Auckland University of Technology.

Hua, J., Tembe, W. D., & Dougherty, E. R. (2009). Performance of Feature-Selection Methods in the Classification of High-Dimension Data. *Pattern Recognition*, *42*(3), 409–424. doi:10.1016/j.patcog.2008.08.001

Huang, T.-M., & Kecman, V. (2005). *Gene Extraction for Cancer Diagnosis by Support Vector Machines*. Paper presented at the International Conference on Artificial Neural Networks (ICANN), Warsaw, Poland. 10.1007/11550822_96

Huerta, E. B., Duval, B., & Hao, J.-K. (2006). *A Hybrid Ga/Svm Approach for Gene Selection and Classification of Microarray Data*. Paper presented at the Workshops on Applications of Evolutionary Computation, Budapest, Hungary. 10.1007/11732242_4

Huiqing, L. (2004). *Effective Use of Data Mining Technologies on Biological and Clinical Data* (PhD Thesis). National University of Singapore.

Jäger, J., Sengupta, R., & Ruzzo, W. L. (2002). *Improved Gene Selection for Classification of Microarrays.* Paper presented at the Pacific Symposium on Biocomputing, Lihue, HI. 10.1142/9789812776303_0006

Janecek, A. (2009). *Efficient Feature Reduction and Classification Methods, Applications in Drug Discovery and Email Categorization* (PhD Thesis), University of Vienna. (A 786 880)

Jawdat. (2006). *The Era of Bioinformatics*. Paper presented at the 2nd International Conference on Information & Communication Technologies, Damascus, Syria. doi:10.1109/ICTTA.2006.1684672

Jena, R. K., Aqel, M. M., Srivastava, P., & Mahanti, P. K. (2009). Soft Computing Methodologies in Bioinformatics. *European Journal of Scientific Research*, *26*(2), 189–203.

Jeyachidra, J., & Punithavalli, M. (2013). *A Comparative Analysis of Feature Selection Algorithms on Classification of Gene Microarray Dataset*. Paper presented at the International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India. 10.1109/ICICES.2013.6508165

Jones, N. C., & Pevzner, P. (2004). *An Introduction to Bioinformatics Algorithms*. MIT Press.

Jourdan, L., Dhaenens, S., & Talbi, E.-G. (2001). A Genetic Algorithm for Feature Selection in Data-Mining for Genetics. *Proceedings of the 4th Metaheuristics International ConferencePorto (MIC'2001).*

Jumali, D., & Hashim, M. (2009). *A Study of Network-Based Approach for Cancer Classification*. Paper presented at the International Conference on Information Management and Engineering, Kuala Lumpur, Malaysia. 10.1109/ICIME.2009.104

Jyun-Jie, L., & Pei-Chann, C. (2010). *A Particle Swarm Optimization Based Classifier for Liver Disorders Classification.* Paper presented at the International Conference on Computational Problem-Solving (ICCP).

Karabulut, E. M., Özel, S. A., & Ibrikci, T. (2012). A Comparative Study on the Effect of Feature Selection on Classification Accuracy. *Procedia Technology*, *1*, 323–327. doi:10.1016/j.protcy.2012.02.068

Kasabov, N. (2004). Bioinformatics: A Knowledge Engineering Approach. *Proceedings of the 2nd International IEEE Conference on Intelligent Systems*, 19-24. 10.1109/IS.2004.1344630

Kaulitzki, S. (2018). *3-D Rendered Illustration of a Colon Tumour*. Retrieved from: https://www.shutterstock.com/image-illustration/3d-rendered-illustration-colon-tumor-130089656?src=EDizE28-03Vl_dYxHGwmBQ-2-30

Khan. (2009). *Colon Cancer Classification Using Microarray Data* (MSc Dissertation). The British University in Dubai (BUiD).

Khobragade, V., & Vinayababu, A. (2012). A Comparative Analysis of Hybrid Approach for Gene Cancer Classification Using Genetic Algorithm and Ffbnn with Classifiers Anfis and Fuzzy Nn. *IOSR Journal of Engineering*, *2*(11), 44–52. doi:10.9790/3021-021134452

Kihara, D., Yang, Y. D., & Hawkins, T. (2006). Bioinformatics Resources for Cancer Research with an Emphasis on Gene Function and Structure Prediction Tools. *Cancer Informatics*, *2*, 25–35. doi:10.1177/117693510600200020 PMID:19458756

Kilany, R. M. (2013). *Efficient Classification and Prediction Algorithms for Biomedical Information* (PhD Thesis). University of Connecticut.

Kim, K.-J., & Cho, S.-B. (2004). Prediction of Colon Cancer Using an Evolutionary Neural Network. *Neurocomputing*, *61*, 361–379. doi:10.1016/j.neucom.2003.11.008

Kim, K.-J., & Cho, S.-B. (2008). An Evolutionary Algorithm Approach to Optimal Ensemble Classifiers for DNA Microarray Data Analysis. *IEEE Transactions on Evolutionary Computation*, *12*(3), 377–388. doi:10.1109/TEVC.2007.906660

Kirar, J. S., & Agrawal, R. (2017). Composite Kernel Support Vector Machine Based Performance Enhancement of Brain Computer Interface in Conjunction with Spatial Filter. *Biomedical Signal Processing and Control*, *33*, 151–160. doi:10.1016/j.bspc.2016.09.014

Kötzing, T., Neumann, F., & Spöhel, R. (2011). Pac Learning and Genetic Programming. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2091-2096. 10.1145/2001576.2001857

Kulkarni, A., Kumar, B. N., Ravi, V., & Murthy, U. S. (2011). Colon Cancer Prediction with Genetics Profiles Using Evolutionary Techniques. *Expert Systems with Applications*, *38*(3), 2752–2757. doi:10.1016/j.eswa.2010.08.065

Lakshmana, T. R., Botella, C., & Svensson, T. (2012). Partial Joint Processing with Efficient Backhauling Using Particle Swarm Optimization. *EURASIP Journal on Wireless Communications and Networking*, *2012*(1), 182–200. doi:10.1186/1687-1499-2012-182

Lavanya, C., Nandihini, M., Niranjana, R., & Gunavathi, C. (2014). Classification of Microarray Data Based on Feature Selection Method. *International Journal of Innovative Research in Science, Engineering and Technology*, *3*(1), 1261–1264.

Lee, C.-P., & Leu, Y. (2011). A Novel Hybrid Feature Selection Method for Microarray Data Analysis. *Applied Soft Computing*, *11*(1), 208–213. doi:10.1016/j.asoc.2009.11.010

Lee, C.-P., Lin, W.-S., Chen, Y.-M., & Kuo, B.-J. (2011). Gene Selection and Sample Classification on Microarray Data Based on Adaptive Genetic Algorithm/K-Nearest Neighbor Method. *Expert Systems with Applications*, *38*(5), 4661–4667. doi:10.1016/j.eswa.2010.07.053

Lee, J. W., Lee, J. B., Park, M., & Song, S. H. (2005). An Extensive Comparison of Recent Classification Tools Applied to Microarray Data. *Computational Statistics & Data Analysis*, *48*(4), 869–885. doi:10.1016/j.csda.2004.03.017

Lee, Z.-J. (2008). An Integrated Algorithm for Gene Selection and Classification Applied to Microarray Data of Ovarian Cancer. *Artificial Intelligence in Medicine*, *42*(1), 81–93. doi:10.1016/j.artmed.2007.09.004 PMID:18006289

Li, J., Liu, H., Ng, S.-K., & Wong, L. (2003). Discovery of Significant Rules for Classifying Cancer Diagnosis Data. *Bioinformatics (Oxford, England)*, *19*(2), 93–102. doi:10.1093/bioinformatics/btg1066 PMID:14534178

Li, S., Wu, X., & Hu, X. (2008). Gene Selection Using Genetic Algorithm and Support Vectors Machines. *Soft Computing-A Fusion of Foundations. Methodologies and Applications*, *12*(7), 693–698. doi:10.100700500-007-0251-2

Li, S., Wu, X., & Tan, M. (2008). Gene Selection Using Hybrid Particle Swarm Optimization and Genetic Algorithm. *Soft Computing*, *12*(11), 1039–1048. doi:10.100700500-007-0272-x

Li, T., Zhang, C., & Ogihara, M. (2004). A Comparative Study of Feature Selection and Multiclass Classification Methods for Tissue Classification Based on Gene Expression. *Bioinformatics (Oxford, England)*, *20*(15), 2429–2437. doi:10.1093/bioinformatics/bth267 PMID:15087314

Liu, H., Liu, L., & Zhang, H. (2010). Ensemble Gene Selection for Cancer Classification. *Pattern Recognition*, *43*(8), 2763–2772. doi:10.1016/j.patcog.2010.02.008

Liu, J., & Iba, H. (2002). Selecting Informative Genes Using a Multiobjective Evolutionary Algorithm. *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02),* 297-302 10.1109/CEC.2002.1006250

Lorena, A. C., Costa, I. G., & de Souto, M. C. (2008). *On the Complexity of Gene Expression Classification Data Sets*. Paper presented at the Eighth International Conference on Hybrid Intelligent Systems (HIS'080), Barcelona, Spain. 10.1109/HIS.2008.163

Lowery, A., Lemetre, C., Ball, G., & Kerin, M. (2011). *Microarray Technology - Expression Profiling of Mrna and Microrna in Breast Cancer*. IntechOpen. doi:10.5772/25176

Lu, H., Chen, J., Yan, K., Jin, Q., Xue, Y., & Gao, Z. (2017). A Hybrid Feature Selection Algorithm for Gene Expression Data Classification. *Neurocomputing*, *256*, 56–62. doi:10.1016/j.neucom.2016.07.080

Lu, Y., & Han, J. (2003). Cancer Classification Using Gene Expression Data. *Information Systems*, *28*(4), 243–268. doi:10.1016/S0306-4379(02)00072-8

Mishra, A., Devi, R., & Shrivastava, S. (2015, June). Gene Expression Data Analysis Using Data Mining Algorithms for Colon Cancer. *International Journal of Advance Research in Science & Engineering*, *4*(6), 171–177.

Mittal, A. (2017). *Ml Process and Scenarios*. Retrieved from: https://elearningindustry.com/machine-learning-process-and-scenarios

Mohamad, M. S., Deris, S., & Illias, R. M. (2005). A Hybrid of Genetic Algorithm and Support Vector Machine for Features Selection and Classification of Gene Expression Microarray. *International Journal of Computational Intelligence and Applications*, *5*(01), 91–107. doi:10.1142/S1469026805001465

Mohamad, M. S., Omatu, S., Deris, S., & Hashim, S. Z. M. (2007). A Model for Gene Selection and Classification of Gene Expression Data. *Artificial Life and Robotics*, *11*(2), 219–222. doi:10.100710015-007-0432-1

Mohamad, M. S., Omatu, S., Deris, S., Misman, M. F., & Yoshioka, M. (2009). Selecting Informative Genes from Microarray Data by Using Hybrid Methods for Cancer Classification. *Artificial Life and Robotics*, *13*(2), 414–417. doi:10.100710015-008-0534-4

Mohamad, M. S., Omatu, S., Deris, S., & Yoshioka, M. (2009). Particle Swarm Optimization for Gene Selection in Classifying Cancer Classes. *Artificial Life and Robotics*, *14*(1), 16–19. doi:10.100710015-009-0712-z

Mohamad, M. S., Omatu, S., Deris, S., & Yoshioka, M. (2010). *Selecting Informative Genes from Microarray Data by Using a Cyclic Ga-Based Method*. Paper presented at the International Conference on Intelligent Systems, Modelling and Simulation (ISMS), Liverpool, UK. 10.1109/ISMS.2010.14

Mohamad, M. S., Omatu, S., Yoshioka, M., & Deris, S. (2008). *An Approach Using Hybrid Methods to Select Informative Genes from Microarray Data for Cancer Classification*. Paper presented at the Second Asia International Conference on Modeling & Simulation (AICMS 08), Kuala Lumpur, Malaysia. 10.1109/AMS.2008.71

Mukkamala, S., Liu, Q., Veeraghattam, R., & Sung, A. H. (2005). Computational Intelligent Techniques for Tumor Classification (Using Microarray Gene Expression Data). *International Journal of Lateral Computing*, *2*(1), 38–45.

MyHealth.Alberta.ca. (2018). *Colon Polyps: Care Instructions*. Retrieved from: https://myhealth.alberta.ca/Health/aftercareinformation/pages/conditions.aspx?hwid=ut2896

Nair, A. S. (2007). Computational Biology & Bioinformatics: A Gentle Overview. *Communications of the Computer Society of India*, *2*, 1–12.

Ng, S.-K., & Wong, L. (2004). Accomplishments and Challenges in Bioinformatics. *IT Professional*, *6*(1), 44–50. doi:10.1109/MITP.2004.1265543

Ni, B., & Liu, J. (2004). A Hybrid Filter/Wrapper Gene Selection Method for Microarray Classification. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, 2537-2542. 10.1109/ICMLC.2004.1382231

Nurminen, J. K. (2003). Using Software Complexity Measures to Analyze Algorithms—An Experiment with the Shortest-Paths Algorithms. *Computers & Operations Research*, *30*(8), 1121–1134. doi:10.1016/S0305-0548(02)00060-6

Ong, H. F., Mustapha, N., & Sulaiman, M. N. (2011). Integrative Gene Selection for Classification of Microarray Data. *Computer and Information Science*, *4*(2), 55–63. doi:10.5539/cis.v4n2p55

Osareh, A., & Shadgar, B. (2010). *Microarray Data Analysis for Cancer Classification*. Paper presented at the 5th International Symposium on Health Informatics and Bioinformatics (HIBIT), Antalya, Turkey. 10.1109/HIBIT.2010.5478893

Park, C., & Cho, S.-B. (2003). *Evolutionary Ensemble Classifier for Lymphoma and Colon Cancer Classification*. Paper presented at the 2003 Congress on Evolutionary Computation (CEC'03), Canberra, Australia. 10.1109/CEC.2003.1299385

Patil, M. D., & Sane, S. S. (2014). Effective Classification after Dimension Reduction: A Comparative Study. *International Journal of Scientific and Research Publications*, *4*(7), 1–4.

Pattanateepapon, A., Suwansantisuk, W., & Kumhom, P. (2016). *Methods to Transform Microarray Data for Cancer Prediction*. Paper presented at the IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Chiang Mai, Thailand. 10.1109/CIBCB.2016.7758104

Paul, T. K., & Iba, H. (2004). *Selection of the Most Useful Subset of Genes for Gene Expression-Based Classification*. Paper presented at the Congress on Evolutionary Computation (CEC2004), Portland, OR. 10.1109/CEC.2004.1331152

Peng, Y., Wang, G., Kou, G., & Shi, Y. (2011). An Empirical Study of Classification Algorithm Evaluation for Financial Risk Prediction. *Applied Soft Computing*, *11*(2), 2906–2915. doi:10.1016/j.asoc.2010.11.028

Porkodi & Suganya. (2015). A Comparative Study on Classification Algorithms in Data Mining Using Microarray Dataset of Colon Cancer. *International Journal of Advanced Research in Computer Science and Software Engineering, 5*(5), 1768-1777.

Pradhananga, N. (2007). *Effective Linear-Time Feature Selection* (MSc Dissertation). The University of Waikato, Hamilton, New Zealand.

Rathore, S., Hussain, M., Ali, A., & Khan, A. (2013). A Recent Survey on Colon Cancer Detection Techniques. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *10*(3), 545–563. doi:10.1109/TCBB.2013.84 PMID:24091390

Rathore, S., Hussain, M., & Khan, A. (2014). Gecc: Gene Expression Based Ensemble Classification of Colon Samples. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *11*(6), 1131–1145. doi:10.1109/TCBB.2014.2344655 PMID:26357050

Raut, S., Sathe, S. R., & Raut, A. (2010). *Bioinformatics: Trends in Gene Expression Analysis*. Paper presented at the International Conference on Bioinformatics and Biomedical Technology. 10.1109/ICBBT.2010.5479003

Raza, K. (2012). Application of Data Mining in Bioinformatics. *Indian Journal of Computer Science and Engineering*, *1*(2), 114–118.

Rhodes, D., Yu, J., Shanker, K., Deshpande, N., Varambally, R., Ghosh, D., Barrette, T., Pandey, A., & Chinnaiyan, A. (2004). Oncomine: A Cancer Microarray Database and Integrated Data-Mining Platform. *Neoplasia (New York, N.Y.)*, *2004*(6), 1–6. doi:10.1016/S1476-5586(04)80047-2 PMID:15068665

Rodriguez-Lujan, I., Huerta, R., Elkan, C., & Cruz, C. S. (2010). Quadratic Programming Feature Selection. *Journal of Machine Learning Research*, *11*, 1491–1516.

Ruiz, R., Riquelme, J. C., & Aguilar-Ruiz, J. S. (2006). Incremental Wrapper-Based Gene Selection from Microarray Data for Cancer Classification. *Pattern Recognition*, *39*(12), 2383–2392. doi:10.1016/j.patcog.2005.11.001

Saeys, Y., Abeel, T., & Van de Peer, Y. (2008). *Robust Feature Selection Using Ensemble Feature Selection Techniques*. Paper presented at the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Antwerp, Belgium. 10.1007/978-3-540-87481-2_21

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A Review of Feature Selection Techniques in Bioinformatics. *Bioinformatics (Oxford, England)*, *23*(19), 2507–2517. doi:10.1093/bioinformatics/btm344 PMID:17720704

Salem, H., Attiya, G., & El-Fishawy, N. (2015). *Gene Expression Profiles Based Human Cancer Diseases Classification*. Paper presented at the 11th International Computer Engineering Conference (ICENCO), Cairo, Egypt. 10.1109/ICENCO.2015.7416345

Salem, H., Attiya, G., & El-Fishawy, N. (2017). Classification of Human Cancer Diseases by Gene Expression Profiles. *Applied Soft Computing*, *50*, 124–134. doi:10.1016/j.asoc.2016.11.026

Shah, S., & Kusiak, A. (2007). Cancer Gene Search with Data-Mining and Genetic Algorithms. *Computers in Biology and Medicine*, *37*(2), 251–261. doi:10.1016/j.compbiomed.2006.01.007 PMID:16616736

Shah, Z. A., Saad, P., & Othman, R. M. (2009). *Feature Selection for Classification of Gene Expression Data: The 5th Postgraduate Annual Research*. Academic Press.

Sharma, A., Imoto, S., & Miyano, S. (2012). A Top-R Feature Selection Algorithm for Microarray Gene Expression Data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9*(3), 754–764. doi:10.1109/TCBB.2011.151 PMID:22084149

Shukla, A. (2020). Feature selection inspired by human intelligence for improving classification accuracy of cancer types. *Computational Intelligence*, coin.12341. Advance online publication. doi:10.1111/coin.12341

Simon, R. (2005). Bioinformatics in Cancer Therapeutics-Hype or Hope? *Nature Clinical Practice. Oncology*, *2*(5), 223–224. doi:10.1038/ncponc0176 PMID:16264939

Spruyt. (2014). *About the Curse of Dimensionality*. Retrieved from: https://www.datasciencecentral.com/profiles/blogs/about-the-curse-of-dimensionality

Sreepada, R. S., Vipsita, S., & Mohapatra, P. (2015). *An Efficient Approach for Microarray Data Classification Using Filter Wrapper Hybrid Approach*. Paper presented at the IEEE International Advance Computing Conference (IACC), Bangalore, India. 10.1109/IADCC.2015.7154710

Stewart, B. W., & Wild, C. P. (2014). *World Cancer Report*. International Agency for Research on Cancer.

Su, J., & Zhang, H. (2006). *A Fast Decision Tree Learning Algorithm.* Paper presented at the In Proceedings of the 21st national conference on Artificial intelligence (AAAI'06), Boston, MA.

Sujatha, G., & Rani, K. U. (2013). Evaluation of Decision Tree Classifiers on Tumor Datasets. *International Journal of Emerging Trends & Technology in Computer Science*, *2*(4), 418–423.

Sumam, M. I., Sudheep, E. M., & Joseph, A. (2013). A Novel Decision Tree Algorithm for Numeric Datasets-C 4.5* Stat. *International Journal of Advanced Computing*, *36*(1), 1120–1123.

Sun, X., Liu, Y., Xu, M., Chen, H., Han, J., & Wang, K. (2013). Feature Selection Using Dynamic Weights for Classification. *Knowledge-Based Systems*, *37*, 541–549. doi:10.1016/j.knosys.2012.10.001

Szaboj. (2015). *Codedreaming*. Retrieved from http://codedreaming.com/algorithm-time-complexity/

Tan, A. C., & Gilbert, D. (2003). Ensemble Machine Learning on Gene Expression Data for Cancer Classification. *Applied Bioinformatics*, *2*(3), 75–83. PMID:15130820

Tan, A. C., Naiman, D. Q., Xu, L., Winslow, R. L., & Geman, D. (2005). Simple Decision Rules for Classifying Human Cancers from Gene Expression Profiles. *Bioinformatics (Oxford, England)*, *21*(20), 3896–3904. doi:10.1093/bioinformatics/bti631 PMID:16105897

Tan, F. (2007). *Improving Feature Selection Techniques for Machine Learning* (PhD Thesis). Georgia State University.

Tan, F., Fu, X., Zhang, Y., & Bourgeois, A. G. (2006). *Improving Feature Subset Selection Using a Genetic Algorithm for Microarray Gene Expression Data*. Paper presented at the IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, Canada. 10.1109/CEC.2006.1688623

Tan, P., Dowe, D., & Dix, T. (2007). Building Classification Models from Microarray Data with Tree-Based Classification Algorithms. *Advances in Artificial Intelligence*, *4830*, 589–598. doi:10.1007/978-3-540-76928-6_60

Tarek, S., Elwahab, R. A., & Shoman, M. (2017). Gene Expression Based Cancer Classification. *Egyptian Informatics Journal*, *18*(3), 151–159. doi:10.1016/j.eij.2016.12.001

Tsai, C.-W., Tseng, S.-P., Chiang, M.-C., Yang, C.-S., & Hong, T.-P. (2014). A High-Performance Genetic Algorithm: Using Traveling Salesman Problem as a Case. *TheScientificWorldJournal*, *2014*, 1–14. doi:10.1155/2014/178621 PMID:24892038

Tsang, I. W., Kwok, J. T., & Cheung, P.-M. (2005). Core Vector Machines: Fast Svm Training on Very Large Data Sets. *Journal of Machine Learning Research*, *6*, 363–392.

Tseng, L. Y., & Yang, S. B. (2001). A Genetic Approach to the Automatic Clustering Problem. *Pattern Recognition*, *34*(2), 415–424. doi:10.1016/S0031-3203(00)00005-4

Urli, T., Wagner, M., & Neumann, F. (2012). Experimental Supplements to the Computational Complexity Analysis of Genetic Programming for Problems Modelling Isolated Program Semantics. *Parallel Problem Solving from Nature-PPSN XII*, *7491*, 102–112. doi:10.1007/978-3-642-32937-1_11

Vanitha, C. D. A., Devaraj, D., & Venkatesulu, M. (2015). Gene Expression Data Classification Using Support Vector Machine and Mutual Information-Based Gene Selection. *Procedia Computer Science*, *47*, 13–21. doi:10.1016/j.procs.2015.03.178

Venkatesh, E., Tangaraj, P., & Chitra, S. (2010). *Classification of Cancer Gene Expressions from Micro-Array Analysis*. Paper presented at the International Conference on Innovative Computing Technologies (ICICT), Tamil Nadu, India. 10.1109/ICINNOVCT.2010.5440095

Vinaya, V., Bulsara, N., Gadgil, C. J., & Gadgil, M. (2009). Comparison of Feature Selection and Classification Combinations for Cancer Classification Using Microarray Data. *International Journal of Bioinformatics Research and Applications*, *5*(4), 417–431. doi:10.1504/IJBRA.2009.027515 PMID:19640829

Wang, J., Wu, L., Kong, J., Li, Y., & Zhang, B. (2013). Maximum Weight and Minimum Redundancy: A Novel Framework for Feature Subset Selection. *Pattern Recognition*, *46*(6), 1616–1627. doi:10.1016/j.patcog.2012.11.025

Wang, L., Chu, F., & Xie, W. (2007). Accurate Cancer Classification Using Expressions of Very Few Genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *4*(1), 40–53. doi:10.1109/TCBB.2007.1006 PMID:17277412

Wang, L., Zhu, J., & Zou, H. (2008). Hybrid Huberized Support Vector Machines for Microarray Classification and Gene Selection. *Bioinformatics (Oxford, England)*, *24*(3), 412–419. doi:10.1093/bioinformatics/btm579 PMID:18175770

Wang, S., Zhang, Y., Zhan, T., Phillips, P., Zhang, Y., Liu, G., Lu, S., & Wu, X. (2016). Pathological Brain Detection by Artificial Intelligence in Magnetic Resonance Imaging Scanning (Invited Review). *Progress in Electromagnetics Research*, *156*, 105–133. doi:10.2528/PIER16070801

Wang, X., & Gotoh, O. (2009a). Accurate Molecular Classification of Cancer Using Simple Rules. *BMC Medical Genomics*, *2*(1), 64–87. doi:10.1186/1755-8794-2-64 PMID:19874631

Wang, X., & Gotoh, O. (2009b). Microarray-Based Cancer Prediction Using Soft Computing Approach. *Cancer Informatics*, *7*, 123–139. doi:10.4137/CIN.S2655 PMID:19718448

Wang, X., & Gotoh, O. (2010). A Robust Gene Selection Method for Microarray-Based Cancer Classification. *Cancer Informatics*, *9*, 15–30. doi:10.4137/CIN.S3794 PMID:20234770

Wang, Y., Makedon, F. S., Ford, J. C., & Pearlman, J. (2004). Hykgene: A Hybrid Approach for Selecting Marker Genes for Phenotype Classification Using Microarray Gene Expression Data. *Bioinformatics (Oxford, England)*, *21*(8), 1530–1537. doi:10.1093/bioinformatics/bti192 PMID:15585531

Webb, G. I., Boughton, J. R., & Wang, Z. (2005). Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, *58*(1), 5–24. doi:10.100710994-005-4258-6

WHO. (2008). *The Global Burden of Disease: 2004 Update, Health Statistics and Information Systems.* Retrieved from: https://www.who.int/healthinfo/global_burden_disease/GBD_report_2004update_full.pdf?ua=1

Wilkin, D. (2017). *Gene Regulation and Cancer*. Retrieved 2018, from: https://www.ck12.org/biology/gene-regulation-and-cancer/lesson/Gene-Regulation-and-Cancer-Advanced-BIO-ADV/

Xiong, M., Li, W., Zhao, J., Jin, L., & Boerwinkle, E. (2001). Feature (Gene) Selection in Gene Expression-Based Tumor Classification. *Molecular Genetics and Metabolism*, *73*(3), 239–247. doi:10.1006/mgme.2001.3193 PMID:11461191

Yang, C.-H., Chuang, L.-Y., & Yang, C. H. (2010). Ig-Ga: A Hybrid Filter/Wrapper Method for Feature Selection of Microarray Data. *Journal of Medical and Biological Engineering*, *30*(1), 23–28.

Yang, P., & Zhang, Z. (2007). Hybrid Methods to Select Informative Gene Sets in Microarray Data Classification. *AI 2007. Advances in Artificial Intelligence*, *4830*, 810–814. doi:10.1007/978-3-540-76928-6_97

Yang, P., Zhou, B. B., Zhang, Z., & Zomaya, A. Y. (2010). A Multi-Filter Enhanced Genetic Ensemble System for Gene Selection and Sample Classification of Microarray Data. *BMC Bioinformatics*, *11*(1), 1471–2105. doi:10.1186/1471-2105-11-S1-S5 PMID:20122224

Yang, T., Kecman, V., Cao, L., & Zhang, C. (2010). *Combining Support Vector Machines and the T-Statistic for Gene Selection in DNA Microarray Data Analysis*. Paper presented at the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hyderabad, India. 10.1007/978-3-642-13672-6_6

Yeh, Wu, & Chang. (2007). *Applying Data Mining Techniques for Cancer Classification from Gene Expression Data*. Paper presented at the International Conference on Convergence Information Technology (ICCIT 2007), Gyeongju, South Korea. 10.1109/ICCIT.2007.153

Yeung, K. Y., & Bumgarner, R. E. (2005). Multiclass Classification of Microarray Data with Repeated Measurements: Application to Cancer. *Genome Biology*, *6*(13), 405–424. doi:10.1186/gb-2005-6-13-405 PMID:14659020

Yourgenome. (2016). *What Is the 'Central Dogma'?* Retrieved from: https://www.yourgenome.org/facts/what-is-the-central-dogma

Zadeh, J. (2006). An Undergraduate Program in Bioinformatics. *IEEE Potentials*, *25*(3), 43–46. doi:10.1109/MP.2006.1657762

Zeng, Z., Zhang, H., Zhang, R., & Yin, C. (2015). A Novel Feature Selection Method Considering Feature Interaction. *Pattern Recognition*, *48*(8), 2656–2666. doi:10.1016/j.patcog.2015.02.025

Zhang, B., & Cao, P. (2019). Classification of high dimensional biomedical data based on feature selection using redundant removal. *PLoS One*, *14*(4), e0214406. doi:10.1371/journal.pone.0214406 PMID:30964868

Zhang, Z., Yang, P., Wu, X., & Zhang, C. (2009). An Agent-Based Hybrid System for Microarray Data Analysis. *IEEE Intelligent Systems*, *24*(5), 53–63. doi:10.1109/MIS.2009.92

Zhu, Z., Ong, Y.-S., & Dash, M. (2007). Markov Blanket-Embedded Genetic Algorithm for Gene Selection. *Pattern Recognition*, *40*(11), 3236–3248. doi:10.1016/j.patcog.2007.02.007

APPENDIX

Colon Cancer Dataset References

In this appendix, a list is presented of current state-of-the-art studies, scientific research papers and theses using the colon cancer dataset retrieved from (Alon et al., 1999).

Table 14. A list of colon cancer dataset state of art studies that used similar dataset

No.	Reference	Research/ Study/ Thesis Title
1.	(Ali & Gupta, 2006)	Classification and rule generation for colon tumour gene expression data.
2.	(Alladi et al., 2008)	Colon cancer prediction with genetic profiles using intelligent techniques.
3.	(Ammu, Siva Kumar, & Mundayoor, 2014)	A BBO Based Feature Selection Method for DNA Microarray.
4.	(Archetti et al., 2008)	Classification of colon tumour tissues using genetic programming.
5.	(Babu & Sarkar, 2016)	A comparative study of gene selection methods for cancer classification using microarray data.
6.	(Banerjee, Mitra, & Banka, 2007)	Evolutionary rough feature selection in gene expression data.
7.	(Ben-Dor et al., 2000)	Tissue classification with gene expression profiles.
8.	(Chen, 2003)	Gene selection for cancer classification using bootstrapped genetic algorithms and support vector machines.
9.	(Chitode & Nagori, 2013)	A comparative study of microarray data analysis for cancer classification.
10.	(Cho & Won, 2003)	ML in DNA microarray analysis for cancer classification.
11.	(Dang, 2014)	Evolutionary approaches for feature selection in biological data.
12.	(Dash & Patra, 2012)	BIOCOMP Study of Classification Accuracy of Microarray Data for Cancer Classification using Hybrid, Wrapper and Filter Feature Selection Method.
13.	(El Akadi et al., 2009a)	Feature selection for genomic data by combining filter and wrapper approaches.
14.	(El Akadi et al., 2009b)	A new gene selection approach based on Minimum Redundancy-Maximum Relevance (MRMR) and Genetic Algorithm (GA).
15.	(Elsheikh et al., 2011)	Effectiveness of Feature Selection and Classification Techniques for Gene Expression Data Analysis.
16.	(Elyasigomari et al., 2017)	Development of a two-stage gene selection method that incorporates a novel hybrid approach using the cuckoo optimisation algorithm and harmony search for cancer classification.
17.	(Fang, Mustapha, & Sulaiman, 2010)	Integrating biological information for feature selection in microarray data classification.
18.	(Fujarewicz & Wiench, 2003)	Selecting differentially expressed genes for colon tumour classification.
19.	(Gao et al., 2012)	Robust integrated framework for effective feature selection and sample classification and its application to gene expression data analysis.
20.	(García-Nieto et al., 2009)	Sensitivity and specificity based multi objective approach for feature selection: Application to cancer diagnosis.
21.	(Huang & Kecman, 2005)	Gene extraction for cancer diagnosis by support vector machines.
22.	(Huerta, Duval, & Hao, 2006)	A hybrid GA/SVM approach for gene selection and classification of microarray data.
23.	(Kim & Cho, 2004)	Prediction of colon cancer using an evolutionary neural network.

continues on following page

Table 14. Continued

No.	Reference	Research/ Study/ Thesis Title
24.	(Kim & Cho, 2008)	An evolutionary algorithm approach to optimal ensemble classifiers for DNA microarray data analysis.
25.	(Kulkarni et al., 2011)	Colon cancer prediction with genetics profiles using evolutionary techniques.
26.	(Lee & Leu, 2011)	A novel hybrid feature selection method for microarray data analysis.
27.	(Lee et al., 2011)	Gene selection and sample classification on microarray data based on adaptive genetic algorithm/k-nearest neighbour method.
28.	(Lee et al., 2005)	An extensive comparison of recent classification tools applied to microarray data.
29.	(Li, Wu, & Hu, 2008)	Gene selection using genetic algorithm and support vectors machines.
30.	(Liu & Iba, 2002)	Selecting informative genes using a multi objective evolutionary algorithm.
31.	(Lu & Han, 2003)	Cancer classification using gene expression data.
32.	(Mohamad, Deris, & Illias, 2005)	A hybrid of genetic algorithm and support vector machine for features selection and classification of gene expression microarray.
33.	(Mohamad, Omatu, Deris, & Yoshioka, 2009)	Particle swarm optimisation for gene selection in classifying cancer classes.
34.	(Mukkamala et al., 2005)	Computational intelligent techniques for tumour classification (using microarray gene expression data).
35.	(Ni & Liu, 2004)	A hybrid filter/wrapper gene selection method for microarray classification.
36.	(Osareh & Shadgar, 2010)	Microarray data analysis for cancer classification.
37.	(Paul & Iba, 2004)	Selection of the most useful subset of genes for gene expression-based classification.
38.	(Porkodi & Suganya, 2015)	A Comparative Study on Classification Algorithms in Data Mining Using Microarray Dataset of Colon Cancer.
39.	(Rathore, Hussain, & Khan, 2014)	ECC: Gene expression-based ensemble classification of colon samples.
40.	(Ruiz, Riquelme, & Aguilar-Ruiz, 2006)	Incremental wrapper-based gene selection from microarray data for cancer classification.
41.	(Saeys, Abeel, & Van de Peer, 2008)	Robust feature selection using ensemble feature selection techniques.
42.	(Salem, Attiya, & El-Fishawy, 2017)	Classification of human cancer diseases by gene expression profiles.
43.	(Shah, Saad, & Othman, 2009)	Feature Selection for Classification of Gene Expression Data.
44.	(Shen et al., 2007)	A combination of modified particle swarm optimisation algorithm and support vector machine for gene selection and tumour classification.
45.	(Sreepada, Vipsita, & Mohapatra, 2015)	An efficient approach for microarray data classification using filter wrapper hybrid approach.
46.	(Sujatha & Rani, 2013)	Evaluation of decision tree classifiers on tumour datasets.
47.	(Tan & Gilbert, 2003)	Ensemble ML on gene expression data for cancer classification.
48.	(Tan et al., 2005)	Simple decision rules for classifying human cancers from gene expression profiles.
49.	(Tan et al., 2006)	Improving feature subset selection using a genetic algorithm for microarray gene expression data.
50.	(Tarek, Elwahab, & Shoman, 2017)	Gene expression-based cancer classification.
51.	(Vanitha, Devaraj, & Venkatesulu, 2015)	Gene expression data classification using support vector machine and mutual information-based gene selection.
52.	(Vinaya et al., 2009)	Comparison of feature selection and classification combinations for cancer classification using microarray data.

continues on following page

Table 14. Continued

No.	Reference	Research/ Study/ Thesis Title
53.	(Wang, Zhu, & Zou, 2008)	Hybrid huberised support vector machines for microarray classification and gene selection.
54.	(Wang & Gotoh, 2009b)	Microarray-based cancer prediction using soft computing approach.
55.	(Xiong et al., 2001)	Feature (gene) selection in gene expression-based tumour classification.
56.	(Yang & Zhang, 2007)	Hybrid methods to select informative gene sets in microarray data classification.
57.	(Yang, Kecman, et al., 2010)	Combining support vector machines and the t-statistic for gene selection in DNA microarray data analysis.
58.	(Yeh et al., 2007)	Applying Data Mining Techniques for Cancer Classification from Gene Expression Data.
59.	(Hu, 2010)	Personalised Modelling Framework and Systems for Gene Data Analysis and Biomedical Applications

Chapter 3
Research Approach With Machine Learning Underpinned

ABSTRACT

This chapter describes several methodologies and proposed models used to examine the accuracy and efficiency of high-performance colon-cancer feature selection and classification algorithms to solve the problems identified in Chapter 2. An elaboration of the diverse methods of gene/feature selection algorithms and the related classification algorithms implemented throughout this study are presented. A prototypical methodology blueprint for each experiment is developed to answer the research questions in Chapter 1. Each system model is also presented, and the measures used to validate the performance of the model's outcome are discussed.

1. INTRODUCTION

This chapter describes several methodologies and proposed models used to examine the accuracy and efficiency of high-performance colon-cancer feature selection and classification algorithms to solve the problems identified in Chapter 2. An elaboration of the diverse methods of gene/feature selection algorithms and the related classification algorithms implemented throughout this study are presented. A prototypical methodology blueprint for each experiment is developed to answer the research questions in Chapter 1. Each system model is also presented, and the measures used to validate the performance of the model's outcome are discussed.

2. FEATURE SELECTION ALGORITHMS AND TECHNIQUES APPLIED IN THE RESEARCH

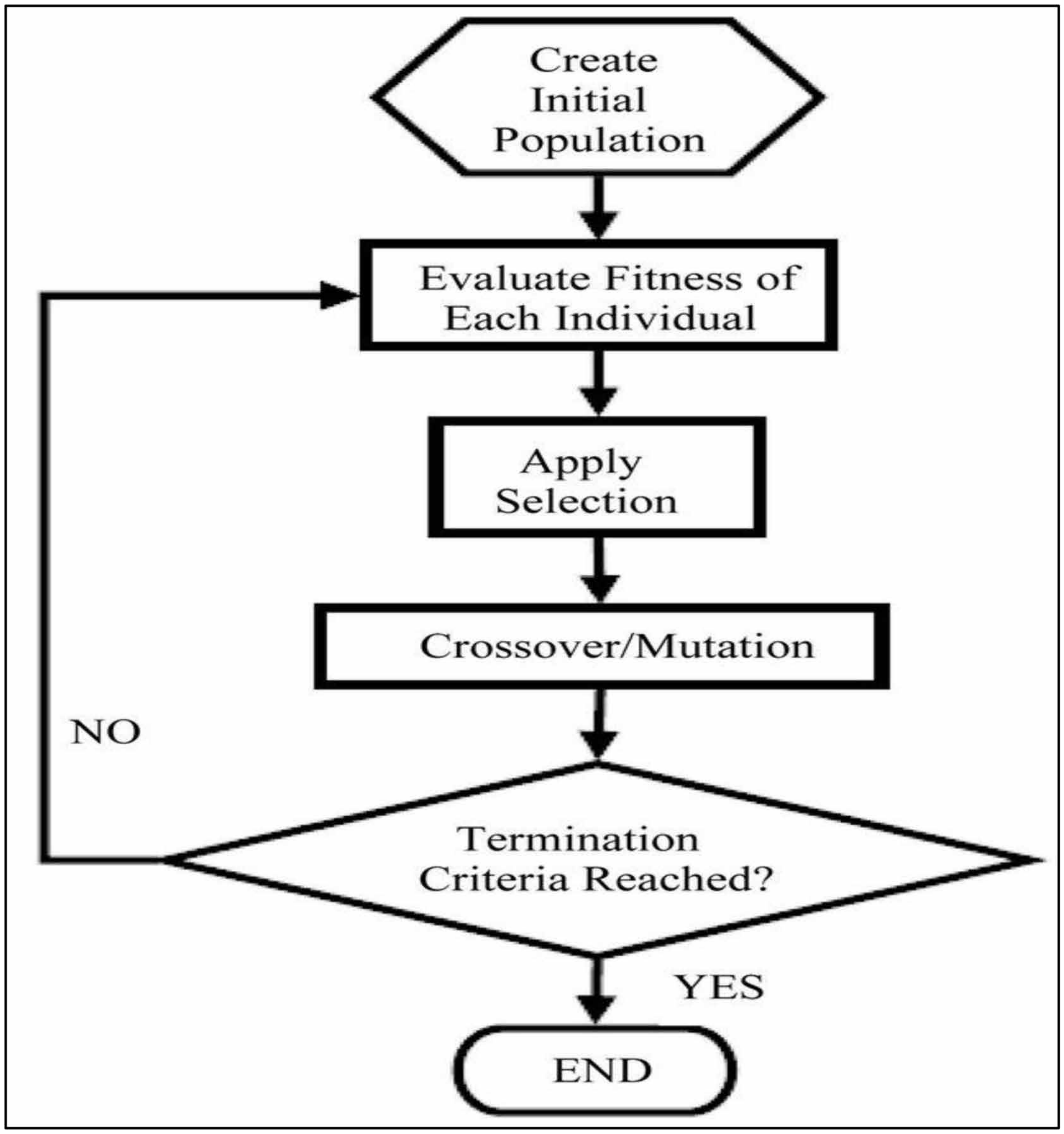
This section spotlights the gene selection algorithms applied in this research. Gene selection processes are applied to provide subsets of genes as candidates to enhance the prediction and classification accuracy (Chapter 2). Before digging into the experiments and case studies, a review of feature selection algorithms and methods is described below, providing mathematical and scientific background on each.

DOI: 10.4018/978-1-7998-7316-7.ch003

2.1 Genetic Algorithm (GA)

A GA is an evolutionary searching algorithm that relies on the idea survival of the fittest, derived from evolutionary biology. It was first proposed by John Holland in 1975, developed from Darwin's theory (Dang, 2014). With a GA, each generated potential chromosome competes with others to create a better generation (i.e., solution) of the target. There is need to know much information about the function used to optimise the generation, because it is a by-product of the GA. Consequently, GAs work well with large search spaces. A GA comprises a set of units including *initial population*, *fitness function*, *population evaluation*, *selection*, *crossover* and *mutation*. Figure 1 presents these sets.

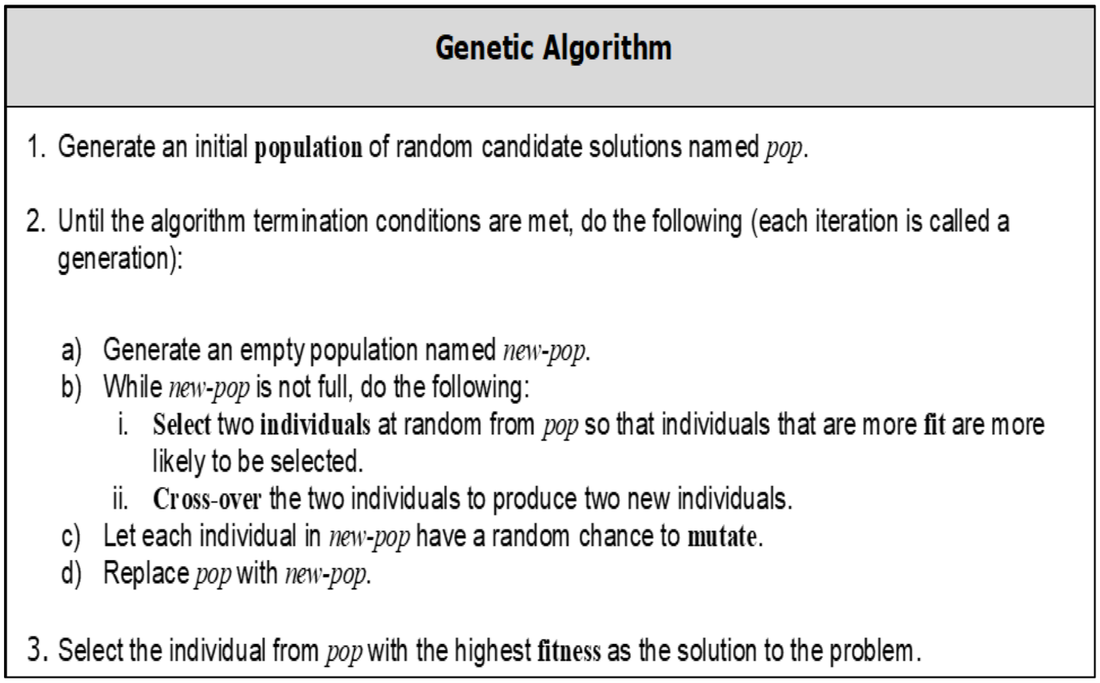
Figure 1. GA general steps


Create Initial Population
Evaluate Fitness of Each Individual
Apply Selection
Crossover/Mutation
NO
Termination Criteria Reached?
YES
END

GA begins with a potential set of solutions (i.e., chromosomes) known as the population (Al-Rajab & Lu, 2016; Goldberg, 1989; Shah & Kusiak, 2007). Solutions from the same population are employed to produce a new population set. This cycle is inspired by the expectation that the newly generated population will be superior than the previous set. Solutions are nominated to generate a new set and are selected based on their suitability (Yeh et al., 2007). GA steps are shown in Figure 2.

- **Initial Population** The initial population is generated from random individual candidates.
- **Fitness Function** The core idea of GA is improving the predictive classification accuracy of the ML algorithm by using the training dataset to extract a feature subset. The fitness function of each candidate is defined as $F(i)=C(i)$, in which $C(i)$ is the accuracy of classification, and the individual, i, in the trained data, is used in the feature subset.

Figure 2. GA pseudocode



```
Genetic Algorithm

1. Generate an initial population of random candidate solutions named pop.

2. Until the algorithm termination conditions are met, do the following (each iteration is called a
   generation):

    a) Generate an empty population named new-pop.
    b) While new-pop is not full, do the following:
        i.  Select two individuals at random from pop so that individuals that are more fit are more
            likely to be selected.
        ii. Cross-over the two individuals to produce two new individuals.
    c) Let each individual in new-pop have a random chance to mutate.
    d) Replace pop with new-pop.

3. Select the individual from pop with the highest fitness as the solution to the problem.
```

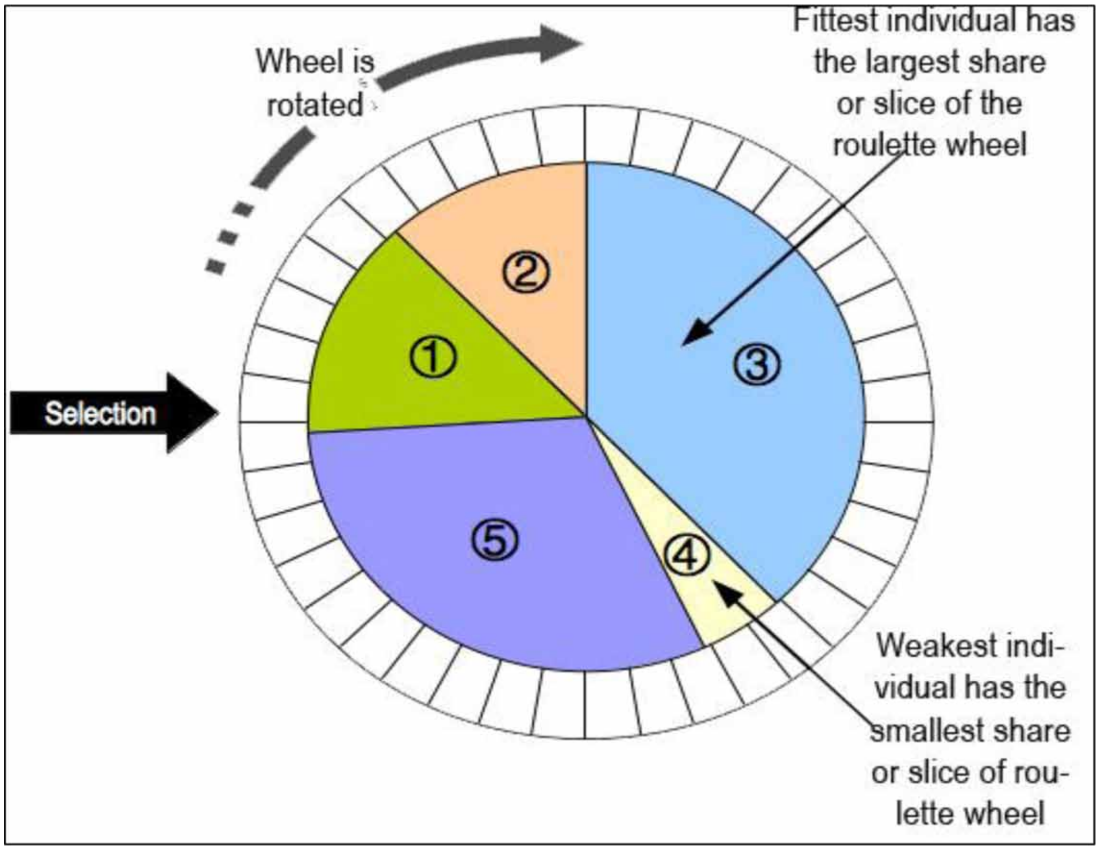
- **Fitness Evaluation** The fitness function is used by the algorithm to evaluate and estimate the fitness and rank each candidate in the initial population.
- **Selection** After fitness ranking, individual selection takes place to locate the fittest individuals to combine for the next generation. There are many selection techniques applied with a GA, including tournament election, ranking selection and roulette-wheel selection. The type of GA selection used in these experiments is the *roulette wheel,* proposed by Goldberg (1989). It is a very popular method in this field of study and is considered a prevalent selection operation, because it is based on the concept of proportionality (Abd Rahman et al., 2016). The probability of selecting any individual, *i,* is given by

$$P(i) = \frac{F(i)}{\sum_{i=1}^{N} F(i)} \tag{1}$$

where $F(i)$ is the fitness value function computed for an individual, i.

The concept of roulette-wheel selection is a linear search mechanism where each candidate is allocated a slot on the 'wheel' based on its weighted fitness value. Thus, all individuals are placed on the wheel in accordance to their fitness value with segment sizes proportional to individual fitness values. The fittest individual will have the biggest slice, and the weakest individual will have the smallest slice. Then, the roulette wheel is spun. When it stops, the candidate linked to the sector upon which the wheel stops is selected. This process is looped until the anticipated number of individuals for the next generation are selected. This procedure is demonstrated in Figure 3.

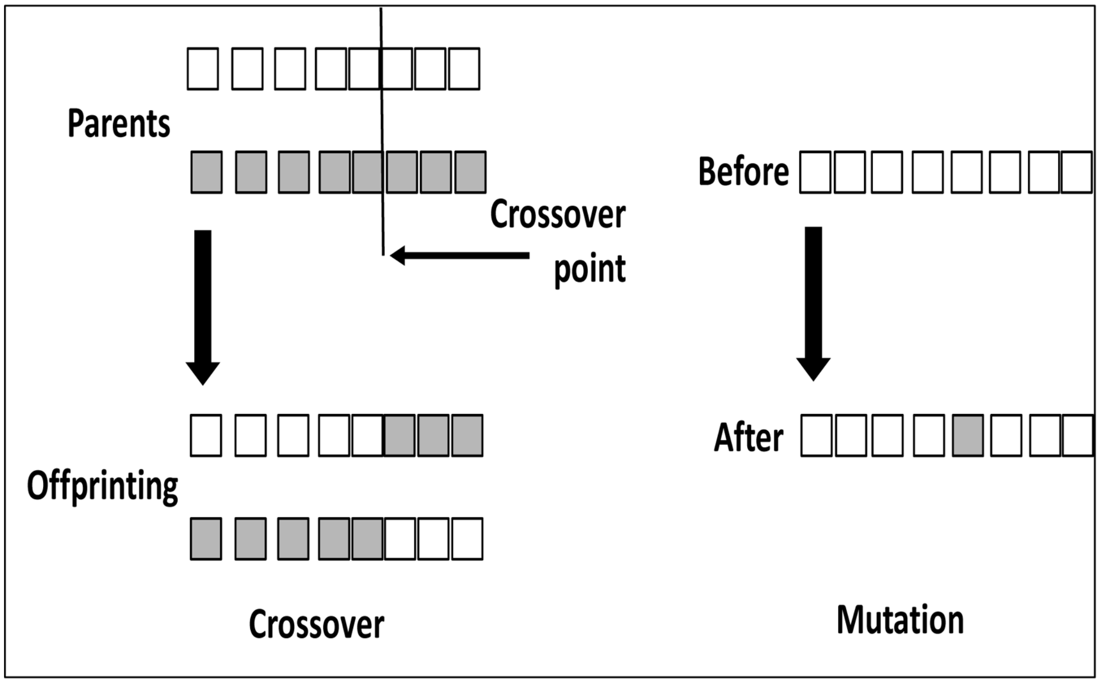
Figure 3. Roulette-wheel fitness distribution of individuals


Wheel is rotated
Fittest individual has the largest share or slice of the roulette wheel
Selection
①
②
③
④
⑤
Weakest individual has the smallest share or slice of roulette wheel

- **Crossover** A crossover or recombination step brings together two or more nominated parents to produce offspring. Generally, the crossover method relies on the type of chromosomal representation.

- **Mutation** After crossover, the offspring can be mutated from a single parent. This generates new individuals with different features not present in their parents. Mutation prevents the new generation from falling into a local optimum. Everyone has a probability of mutation. Figure 4 shows an illustration of crossover and mutation methods.
- **Termination** The evaluation, selection, crossover and mutation processes are looped until a new population is generated, completed and termination condition met. This results in the best set of individuals for the next generation.

Figure 4. CROSSOVER and mutation illustration
Source [Adapted from]: (El-Garhy et al., 2011)



2.2. Particle Swarm Optimization (PSO)

PSO is an evolutionary search algorithm dependent upon population and is stimulated by the social behaviour of fish schooling or bird flocking. It was first proposed in 1995 by Kennedy and Eberhart (Chuang, Ke, & Yang, 2008). It is an optimisation algorithm identical to GA, wherein the schema of each algorithm is initiated with random potential solutions. It is dissimilar from GA, however, in that each potential solution is appointed a random velocity. This velocity is utilised by each particle to 'fly' through the problem space (Shi, 2001).

In PSO, the potential solution can be treated as a distinct fish or bird from the 'swarm' (Chuang et al., 2009). Thus, the set is assembled from a specific number of particles. Each particle possesses a memory and knowledge experience that is obtained from the set to explore the best optimum solution. Each particle is associated with a velocity attribute that directs their movement through the search space.

Additionally, each gains a fitness value, which is assessed by a fitness function similar to that of the GA. Particles with dynamically adjusted velocities fly through the search space according to their historical behaviours, their own experience and the neighbour experience. This process assists the particle in discovering the optimum position using its experience in a *d*-dimensional search space while benefitting from the neighbour experience. The global optima of each repetition can be discovered when all particles move inside the problem space (Alba et al., 2007). Each particle is represented as $x_i = (x_i1, x_i2, x_i3, \ldots, x_id)$, where *d* stands for the dimension number. Its best or optimal former position of the *i*th particle is logged and presented as $p_i = (p_i1, p_i2, p_i3, \ldots, p_id)$. The *i*th particle velocity is signified by $v_i = \left(v_i1, v_i2, v_i3, \ldots, v_id\right)$. The best particle found in the swarm is called $pBest_i$ (best global), and its index is represented by *g*. The particle at each iteration update its velocity and position according to following equations.

$$vNew_i d = w \times vOld_i d + C_1 \times rand\left(\ \right) \times \left(pBest_i d - xOld_i\right) + C_2 \times rand\left(\ \right) \times gBest_d - xOld_d) \tag{2}$$

$$xNew_i d = xOld_i d + vNew_i d \tag{3}$$

For a specific particle, *i*, at a specific iteration, *k*, the updated velocity and position can be computed with Equations 2 and 3. In these equations, *w* stands for the inertial weight, whereas C_1 and C_2 are two acceleration positive constants (i.e., factors) applied in the set's simulations. They are known as cognitive and social acceleration coefficients, respectively (Dara & Tiwari, 2017). *C*1 and *C*2 represent the stochastic acceleration weight, which is used to pull each particle toward *pBest* and *gBest*. *rand*() is random function that generates random numbers. $vNew_i d$ and $vOld_i d$ describe the new and old particle velocities, respectively. Additionally, $xNew_i d$ and $xOld_i d$ represent the updated and current particle positions, respectively. Figure 5 explains the particle swarm optimisation algorithm.

2.2.1. Binary PSO

Many modifications are carried out, improving PSO's performance. A common variant of PSO is the binary version of the algorithm (i.e., Binary PSO (BPSO)), developed to deal with discrete values of the search space instead of real values to enhance performance. In the BPSO, the particle positions are defined as binary vectors. The same equation that manages the velocity (Equation 2) is used. The only difference with BPSO is that the velocity vector is mapped into discrete values between intervals of 0 and 1 for each component using the sigmoid function,

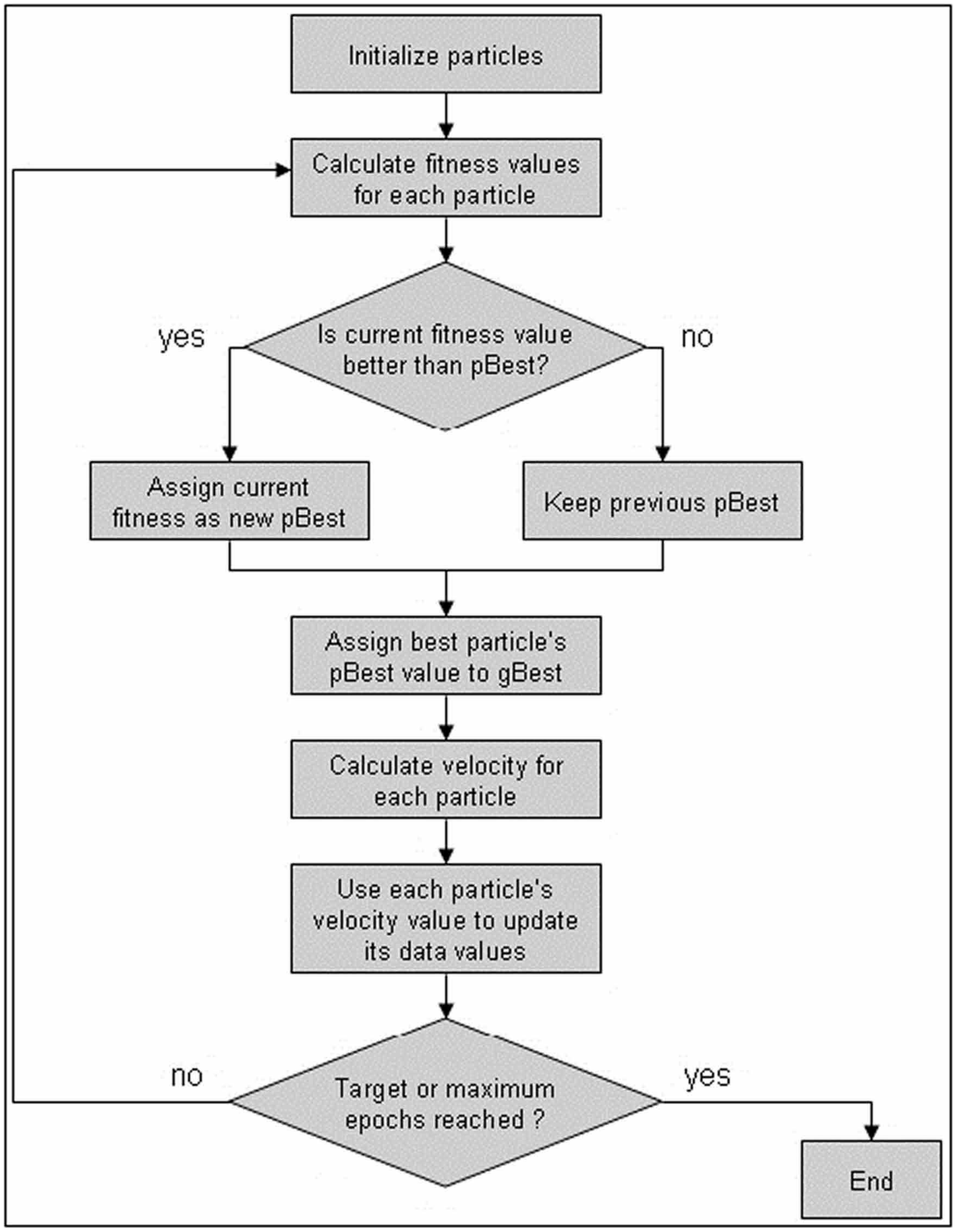
$$S\left(V_{id}\right) = \frac{1}{1 + e^{-v_{id}}} \tag{4}$$

where V_{id} is the component located in dimension *d* of the *i*th particle's velocity. The velocity is updated based on Equation 3-5, where *p* signifies a random number between 0 and 1.

$$if\left(p < S\left(V_{id}\right)\right),\ then\ X_{id} = 1;\ else\ X_{id} = 0 \tag{5}$$

BPSO is an appropriate and convenient algorithm, because binary encoding is natural for feature selection (Chantar, 2013). Therefore, BPSO is implemented throughout this research as a selection algorithm when appropriate via the case-study experiments.

Figure 5. PSO Algorithm


Initialize particles
Calculate fitness values for each particle
Is current fitness value better than pBest?
yes
no
Assign current fitness as new pBest
Keep previous pBest
Assign best particle's pBest value to gBest
Calculate velocity for each particle
Use each particle's velocity value to update its data values
Target or maximum epochs reached ?
no
yes
End

2.3. Minimum Redundancy Maximum Relevance (mRMR)

mRMR was first suggested by Peng et al. (Peng, Long, & Ding, 2005) as a method of searching for subsets from the entire population sustaining the mRMR criteria. It provides a better-balanced space coverage. This method captures wider phenotypes borders. It is a filter method that selects genes containing the information most related to the objective class of the disease, maintaining minimal redundancy (Alshamlan, Badr, & Alohali, 2015; El Akadi et al., 2009b). mRMR selects genes that are extremely different from each other to enhance the selections of genes and to filter redundant and unwanted genes (Abdi, Hosseini, & Rezghi, 2012). This method depends on mutual or common information between the genes to specify their relevance compared to the target class. Thus, mRMR selects genes with the maximum relevance to the target class with minimal redundancy. The mutual information defined for the frequencies of gene and class appearance in terms of their probabilities, ($p(g_i)$, $p(c)$ and $p(g_i,c)$), are given by Equation 6 as follows (El Akadi et al., 2009a, 2009b; Elyasigomari et al., 2017).

$$I\left(g_i,c\right)=\iint p\left(g_i,c\right)\log\frac{p\left(g_i,c\right)}{p\left(g_i\right)p\left(c\right)}dg_i dc \tag{6}$$

where g_i signifies gene i, and c is the class label. The method then selects the top genes with high relevance using the maximum-relevance equation in descending order of $I(g_i,c)$, using Equation 7. This method selects the top individual features correlated with the class labels.

$$V\left(S\right)=\frac{1}{\left|S\right|}\sum_{g_i\in S}I\left(g_i,c\right) \tag{7}$$

where S represents the total number of significant genes extracted, and $I(g_i,c)$ is the mutual information value between gene g_i and its corresponding class, c. After selecting the maximum relevant genes, there is no guarantee that they are not redundant. Hence, to eliminate redundancy, we apply the minimum-redundancy criteria expressed in Equation 8.

$$W\left(S\right)=\frac{1}{\left|S\right|^2}\sum_{g_i,g_j\in S}I\left(g_i,g_j\right) \tag{8}$$

where $I(g_i,g_j)$ is the mutual information value between genes g_i and g_j. Because both Equations (7) and (8) are important (Elyasigomari et al., 2017), they must be combined to formulate the mRMR method. The simplest combination requires either the additive or multiplicative combinations:

$$\boldsymbol{max}\left(\boldsymbol{V}\left(\boldsymbol{S}\right)-\boldsymbol{W}\left(\boldsymbol{S}\right)\right) \tag{9}$$

$$\max\left(\boldsymbol{V}\left(\boldsymbol{S}\right)/\boldsymbol{W}\left(\boldsymbol{S}\right)\right) \tag{10}$$

2.4. Information Gain (IG)

IG is a prevailing algorithm for gene selection based on information theory (Jeyachidra & Punithavalli, 2013; Patil, Naik, & Pai, 2014; Salem, Attiya, & El-Fishawy, 2017). It is a public-filter method for estimating attributes (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2015), and it measures how much information a feature can provide about a class label. It is an efficient feature-entropy-based selection algorithm, widely applied in the ML domain. It can select related features and eliminate redundant ones. It can be described by the existence or absence of features inside the population when the number of information bits is provided by the feature items. Thus, the optimal subset will be selected. A gene with a high computed information gain will be more appropriate and relevant to classification (Lei, 2012; Liu, Li, & Wong, 2002; Zhang, Ren, & Yang, 2013). It measures the class having the best probability of being gained by dividing a certain full-range gene expression into two intervals corresponding to gene regulation (Al-Rajab & Lu, 2014). It detects samples of one interval as ‘normal’ and the other as ‘cancer’ (Al-Rajab & Lu, 2016; Yeh et al., 2007). To compute IG, a threshold value for selecting features must be determined first. If the feature’s IG value is greater than the purposed threshold value, it will be selected. Otherwise, it will be deleted. The formula used for calculating IG is given as

$$Entropy\left(S\right) = -\sum_{i=1}^{k} P\left(C_i, S\right) \times \log\left(P\left(C_i, S\right)\right) \tag{11}$$

where *S* represents a set of *n* instances (i.e., training samples), and *C* represents a set of *K* classes. $P(C_i,S)$ is the fraction of examples in *S* having the class C_i. Next, let *Sv* be the set remaining from *S*, containing training instances and limited to those from which attribute *A* has the value, *v*. IG can be computed as the variance between *Entropy*(*S*) and the expected information sum of the of the subset, *A*, according to the separate values,

$$infoGain\left(S, A\right) = Entropy\left(S\right) - \sum \left(\frac{\left|S_v\right|}{\left|S\right|}\right) \times Entropy\left(S_v\right) \tag{12}$$

However, if the calculated IG is greater than the threshold value, the chance of having optimum classes is also high (Chuang et al., 2009). The pseudocode of the Information Gain method is illustrated by Figure 6.

2.5. Correlation-Based Feature Selection (CFS)

CFS is a significant and frequently applied data pre-processing filter selection method used for data mining. CFS attempts to optimally decrease the search space by choosing a subset of original features based on a specific evaluation criterion (Yu & Liu, 2003). It decreases the number of features, eliminates unrelated, redundant or noisy data and carries instant effects for applications, such as speeding up a data-mining algorithm or enhancing mining performance (Al-Rajab & Lu, 2016; Tiwari & Singh, 2010). The feature subsets are ranked by an experimental assessment function (i.e., evaluation function), which groups the correlated and uncorrelated features together. Thus, unrelated and redundant features can be removed. The following equation is the evaluation function.

$$CFS_S = \frac{k\overline{r}_{cf}}{\sqrt{k + k \times \left(k-1\right) \times \overline{r}_{ff}}} \tag{13}$$

where CFS_s represents the score of an attribute subset, S, including k attributes, whereas $\overline{r}_{cf}$ represents the class correlation average attribute, where ($f \in S$) and $\overline{r}_{ff}$ is the inter-correlation average attribute.

Figure 6. Information Gain Pseudocode
Source [Adapted from]: (Ahmed et al., 2016)

```
Information Gain Pseudocode
Input:
    $T \leftarrow 0$;
    $C \leftarrow$ domain of class labels;
    $A \leftarrow$ domain of attribute values;
For every $c_i \in C$ do:
    compute $E(c[i])$;
    $H(C) = T + E(c[i]) \times log_2(E(c[i]))$;
    $T \leftarrow H(C)$;
End For
For every $a_j \in A$ do:
    compute $E(a[j])$;
    $sum \leftarrow T + E(a[j]) \times log_2(E(a[j]))$;
    $T \leftarrow sum$;
End For
For every $c_i$ do:
    For every $a_j$ do:
        compute $E(c[i] \mid a[j])$;
        $M \leftarrow T + E(c[i] \mid a[j]) \times log_2(E(c[i] \mid a[j]))$;
        $T \leftarrow M$;
    End For
End For
$H(C|A) \leftarrow (-1) \times sum \times (-1) \times M$;
$InfoGain(C|A) \leftarrow H(C) - H(C|A)$;
return $InfoGain(C|A)$;
```

3. ML CLASSIFICATION ALGORITHMS APPLIED IN THE RESEARCH

This section highlights the ML classification algorithms and methods applied in this research. There are many ML classification algorithms, but as argued in the literature (Cho & Ryu, 2002; Fiori, 2010; Lu & Han, 2003; Patil & Sane, 2014), there exists no single perfect classifier, because the classification accuracy of each algorithm depends mainly on the experimental design. A distinguishing point about past studies is that, for most of the proposed gene classification algorithms, those works were concerned about classification accuracy and did not dedicate much effort on temporal analysis.

An overview of the most popular and successful ML algorithms utilised for colon-cancer classification includes SVM, NB, DTs, GP and KNN. These algorithms are selected because they show highest classification contribution to colon cancer research, as discussed in Chapter 2. A summary of each algorithm is presented below.

3.1. Support Vector Machines (SVM)

SVM is a common and powerful gene expression algorithm for classification, is applied to binary cancer classification and depends on statistical learning theory (Bennet, Ganaprakasam, & Kumar, 2015; Lorena, Costa, & de Souto, 2008; Vapnik, 1999; Wu et al., 2008). It is a supervised learning method that has many advantages. It reflects good performance and data robustness with high dimensionality, which is helpful for microarray gene expression data. It is also flexible for modelling various data types (Alshamlan, Badr, & Alohali, 2015; Dash, Patra, & Tripathy, 2012; El Akadi et al., 2009a; George & Raj, 2011; Salem, Abul Seoud, & Ali, 2011). The original training data are split by the SVM classifier into two distinct linear classes using non-linear mapping. SVM searches for an ideal hyperplane that splits the labelled training sets into positive and negative labelled samples with the best isolation margin. The distance between the hyperplane and the nearest data point of any class label forms the margin. The data points that fall in the margin are support vectors. Generally, the bigger the margin, the lower the classifier error will be, the better the generality of the classifier (Bennet, Ganaprakasam, & Kumar, 2015; Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2015; Wang et al., 2004). The main goal of SVM is choosing an ideal hyperplane, because there may exist several having the ability to divide the training set, as shown in Figure 7. The optimal hyperplane is found by the SVM kernel function (Kwon & Sim, 2013), which is effected by the kernel function type (e.g., linear, polynomial, sigmoid or radial base function) (Kwon & Sim, 2013). The SVM discovers the hyperplane to which it can increase the margin from the class-label boundaries when the data are linearly separated. However, if the data are not separated linearly, the samples will be mapped by a kernel function to construct a hyperplane, and the data become separated linearly (Babu & Sarkar, 2016).

The two main parameters of the hyperplane formula are w and b. The hyperplane formula is defined as (Chitode & Nagori, 2013)

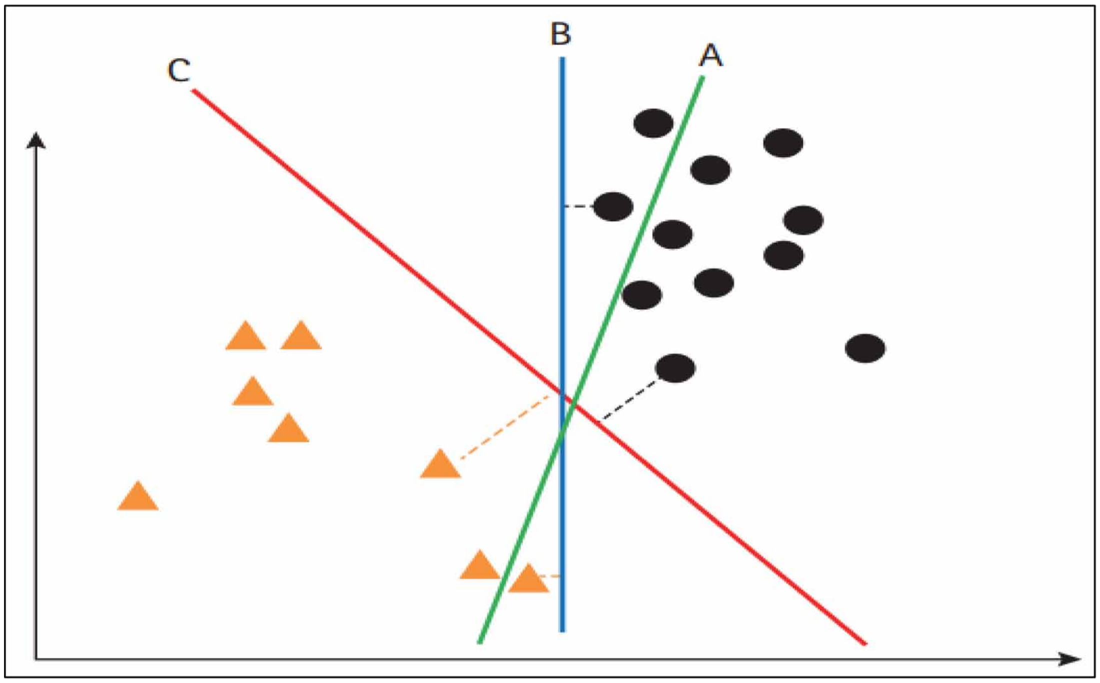
$$wx + b = 0 \tag{14}$$

Where x represents a data point vector, w represents a weight coefficient vector perpendicular to the hyperplane and b represents a scalar, frequently denoted as a bias (i.e., offset term). Nearby hyperplane samples are measured as support vectors and gain a significant role in classifying test samples. Generally, the SVM receives as input a training data set with negative and positive classes (i.e., binary training data). Then, it searches or constructs a decision boundary (i.e., maximum margin) between the two classes, aiming to select the most related data to participate in the decision process via the support vectors. In a given instance, training set label pairs, (x_i,y_i), i=1,2,…,l, where x_i is a training sample belonging to $\mathcal{R}^m$ with y_i class labels, belong to {-1.1}. l is the number of training set samples. The following optimisation problem can be solved by the SVM classifier.

$$min_{w,b,\varepsilon}\ \frac{1}{2}\,||w||^2 + c\sum_{i=1}^{l}\varepsilon_i \qquad (15)$$

Subject to $y_i(wx_i+b)\leq 1-\varepsilon_i$, $\varepsilon_i\leq 0$, $i=1,\ldots,l$, where C *r*epresents the parameter of SVM sensitivity, and ε represents a slack variable (i.e., penalty parameter). Figure 8 reflects the SVM classifiers that find the optimal hyperplane with maximum margin.

Figure 7. Hyperplane of the SVM: (A) where the hyperplane fails to distinguish classes; (B) where the hyperplane distinguishes classes with a small margin; (C) where the hyperplane distinguishes classes with the maximum margin
Source [Adapted from]: (Lee et al., 2017)

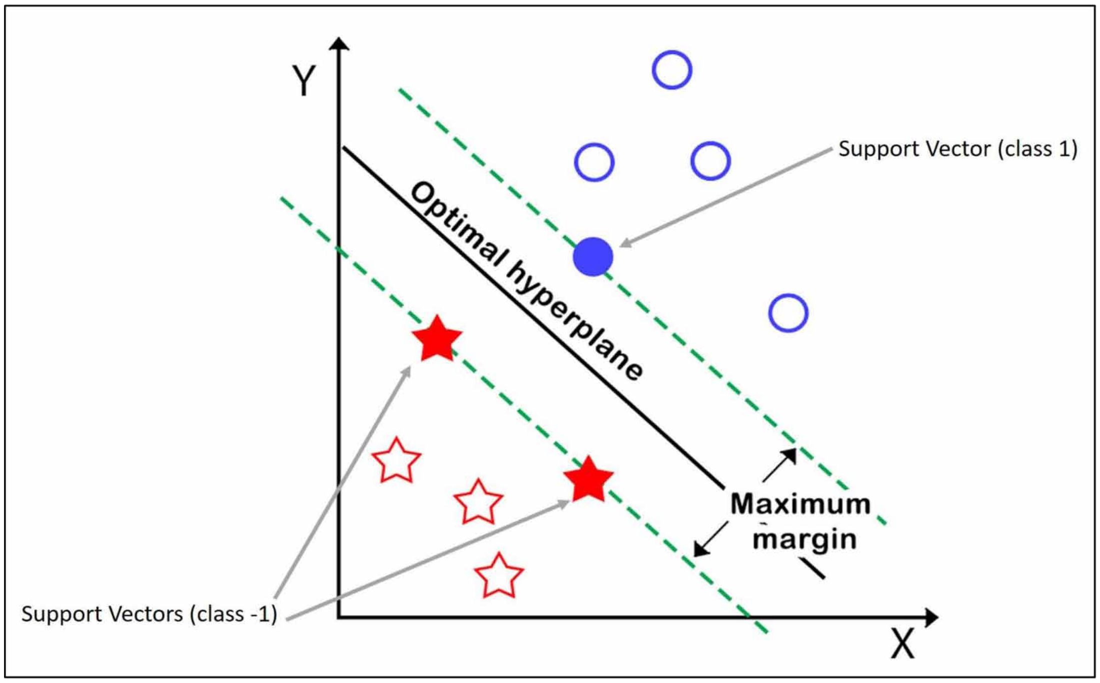


In this research, the John–Platt implementation of the sequential minimal optimisation (SMO) algorithm, available in Weka ML tool (Chapter 4), is used as the representation and induction of the SVM using the polynomial kernel function $K(x,y)=(x.y+1)^{d}$ where x and y are two data points, and the d is a user defined parameter (it has the value of 1 (as default) for the experiments) (Kwon & Sim, 2013; Platt, 1999).

3.2. Naïve Bayes (NB)

NB is a probabilistic classification algorithm that applies Bayes' theory to specify the probability of each class for a given attribute (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2010, 2015). The term 'naïve' indicates that the function enforces strong independent assumptions on the attributes (i.e.,

Figure 8. SVM scheme to discover the optimal hyperplane with the maximum margin



samples) for a given class (Karthikeyan & Thangaraju, 2013; Lorena, Costa, & de Souto, 2008). The NB classifier expects that the absence or presence of a specified feature has no relation to the absence or presence of any feature if the class variable is given. It assumes that each feature independently contributes to the belonging probability of a sample within a specific class, considering the existence or absence of other features (Verónica Bolón, Amparo, & Sánchez, 2017). This classifier has shown many advantages, because it contributed to many complex problems in the real world. This classifier is simple, efficient and robust to noise and unrelated features (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2010, 2015; El Akadi et al., 2009a; Karthikeyan & Thangaraju, 2013).

If a classification problem exists with the *m* class, let *x* be a test example with a probability density observation, $p(x)$. If the class previous probability, $p(m)$, and class conditional probability, $p(x|m)$, are recognised, then based on Bayes' theory, the posterior probability, $p(m|x)$, is

$$p(m \mid x) = \frac{p(x|m).p\left(m\right)}{p\left(x\right)} \tag{16}$$

Afterward, the Bayesian allocates an observation to the class known as a discriminant function with the maximum subsequent probability:

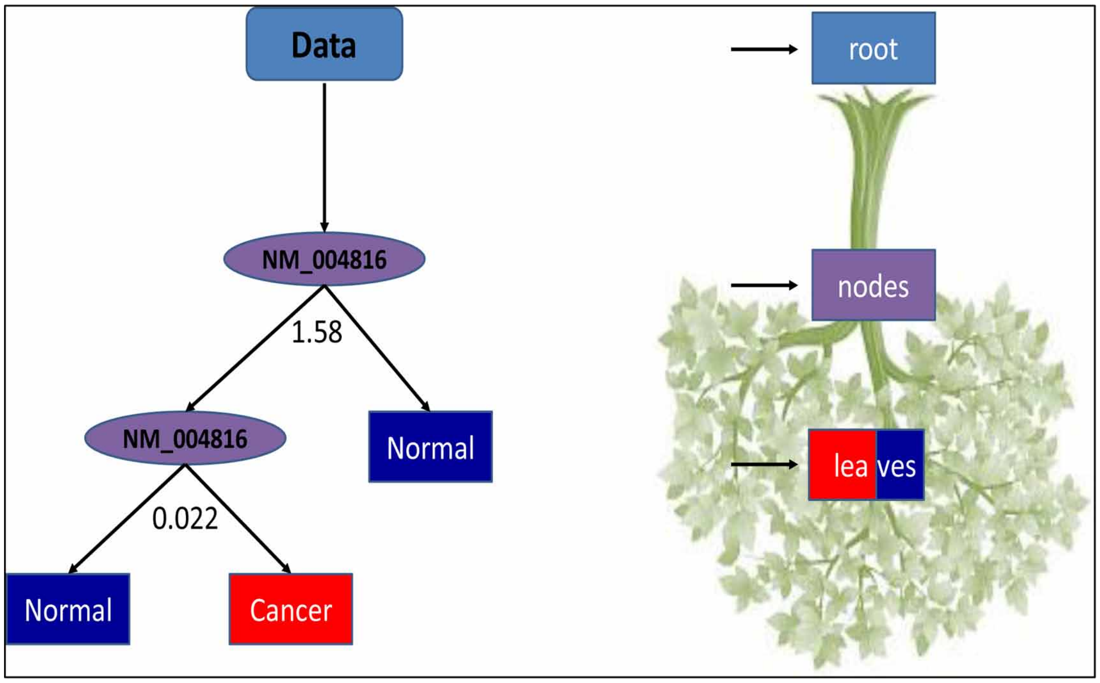
$$f(x) = argmax_{m} p(m|x) \tag{17}$$

3.3. Decision Tree (DT C4.5)

DT is a popular and a common supervised ML algorithm developed by Quinlan Ross (Quinlan, 1993), based on a hierarchal flowchart (i.e., tree) of iterative training samples. It recursively divides the dataset using depth-first search or breadth-first search until all data are mapped to a certain class. DTs are composed of *IF–THEN* rules for classification. The main advantage of the DT is its capability to produce understandable structures. It is a fast, accurate and robust classification algorithm (Hall, 1999).

The most recognised DT algorithms are C4.5, ID3 and classification and regression trees (Sujatha & Rani, 2013). In this research, the J4.8 algorithm of the Weka ML tool (Chapter 4) is the C4.5 Java implementation (Release 8). It is applied throughout the experiments. This structure comprises three components: root, nodes and leaves. See Figure 9.

Figure 9. C4.5 DT structure for gene classification
Source [Adapted from]: (Horng et al., 2009)



In Figure 9, the node represents the classified rules (i.e., tests) for one or more genes/features of gene expression, whereas the leaf nodes represent the classification results. The DT classification algorithm has two phases (Al Snousy et al., 2011; Karthikeyan & Thangaraju, 2013; Sujatha & Rani, 2013):

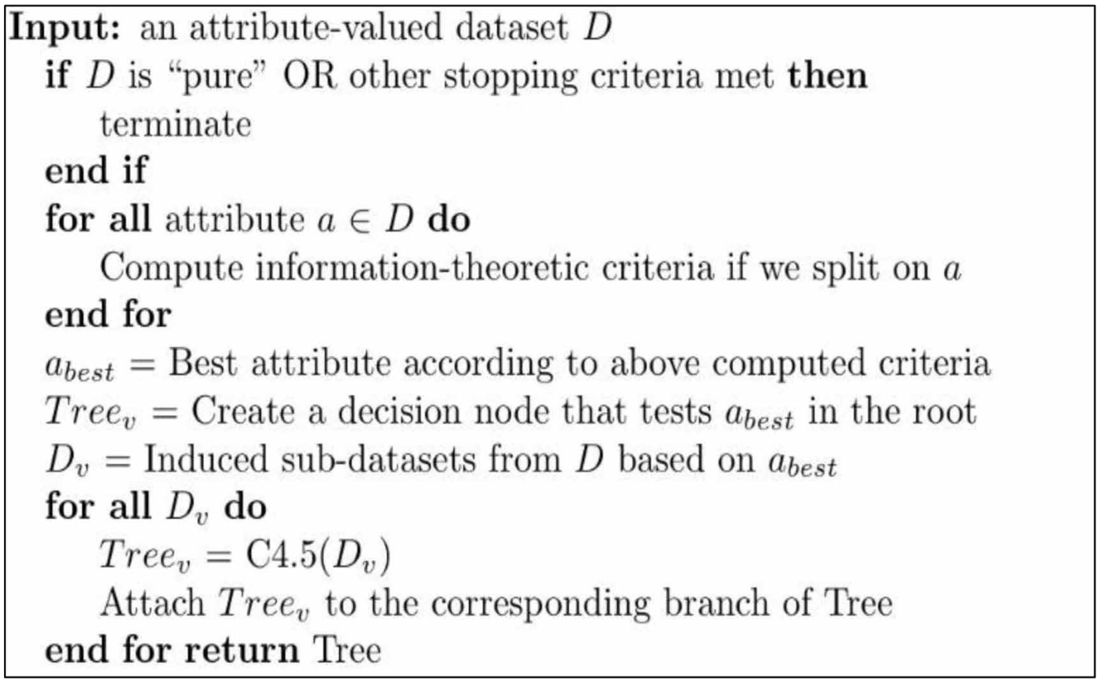
1. **Tree building** is a top-down approach in which the tree is recursively split until all the training data fits an equivalent class label. The tree-building process starts from the root node, and it tests its definite attribute. Then, it moves down the branch per the attribute value. This process is recursively repeated until the leaf node is reached and a given subset of the tree contains instances

from only one class. The C4.5 model uses a gain ratio as a measurement of attribute selection for building the tree. The data are sorted at every node to specify the best-splitting attribute. Thus, the attribute having the highest gain ratio is selected to be the best-splitting attribute.

2. **Tree pruning** is a bottom-up approach used to minimise noise in the training data (i.e., overfitting), which causes misclassification errors. This enhances the classification accuracy of the algorithm and decreases algorithm complexity when datasets contain many attributes. Tree pruning utilises statistical methods to prune the least significant branches of the tree. Thus, the best leaf will replace a node when the leaf estimated error is within one standard deviation of that node's error (Hall, 1999).

The DT pseudocode is presented in Figure 10,

Figure 10. C4.5 DT pseudocode
Source [Adapted from]: (Wu et al., 2008)



```
Input: an attribute-valued dataset $D$
  if $D$ is "pure" OR other stopping criteria met then
      terminate
  end if
  for all attribute $a \in D$ do
      Compute information-theoretic criteria if we split on $a$
  end for
  $a_{best}$ = Best attribute according to above computed criteria
  $Tree_v$ = Create a decision node that tests $a_{best}$ in the root
  $D_v$ = Induced sub-datasets from $D$ based on $a_{best}$
  for all $D_v$ do
      $Tree_v$ = C4.5($D_v$)
      Attach $Tree_v$ to the corresponding branch of Tree
  end for return Tree
```

3.4. K-Nearest Neighbour (K-NN)

K-NN is a classification algorithm with no parametric (Li, Zhang, & Ogihara, 2004). It uses an integer parameter known as *k* (the number of neighbours). The *K*-NN algorithm is considered an unpretentious classification algorithm, easy to implement and works fine with large training datasets (Ammu & Preeja, 2013; Lorena, Costa, & de Souto, 2008; Salem, Abul Seoud, & Ali, 2011). It is one of the most common ML algorithms of the instance-based learners (e.g., lazy learners). K-NN has no building-process model (Janecek, 2009). For a given input, *x*, the algorithm searches for the *k*-nearest training data points to *x*. Then, it forecasts the class label of *x*, depending on the class label of *k* points. Thus, the data are classified by their neighbours' majority voting.

A measurement to compute the distance between different pairs of data items is applied by *K*-NN to determine the nearest neighbours for each datum. Thus, the classifier measures the distance between test data and all training data. Three main elements are utilised to calculate the nearest neighbour (i.e., training data, class labels for each data and the value of *K* to determine the nearest neighbour number) (Salem, Abul Seoud, & Ali, 2011). The main idea of *K*-NN is to determine a sample (i.e., new data item) to the class containing most of the selected numbers of neighbours. These neighbours can be specified by calculating the distance for the *K*-NN and can be computed in a variety of ways, such as the Euclidean distance (the most common) as follows.

$$d_e\left(x_i, x_j\right) = \sqrt{\sum_{q=1}^{d}\left(x_{iq} - x_{jq}\right)^2} \tag{18}$$

where x_i and x_j represent the two points from the training domain, $\Re_d$, and x_{iq} and x_{jq} are their attributes values (i.e. coordinates). A clarification of the *K*-NN algorithm process is presented in Figure 11, and a flowchart of the same algorithm is presented in Figure 12.

Figure 11. K-NN algorithm process

Source [Adapted from]: Latysheva (2016). 'Implementing Your Own k-Nearest Neighbour Algorithm Using Python' [Image]. Retrieved from: https://www.kdnuggets.com/2016/01/implementing-your-own-knn-using-python.html

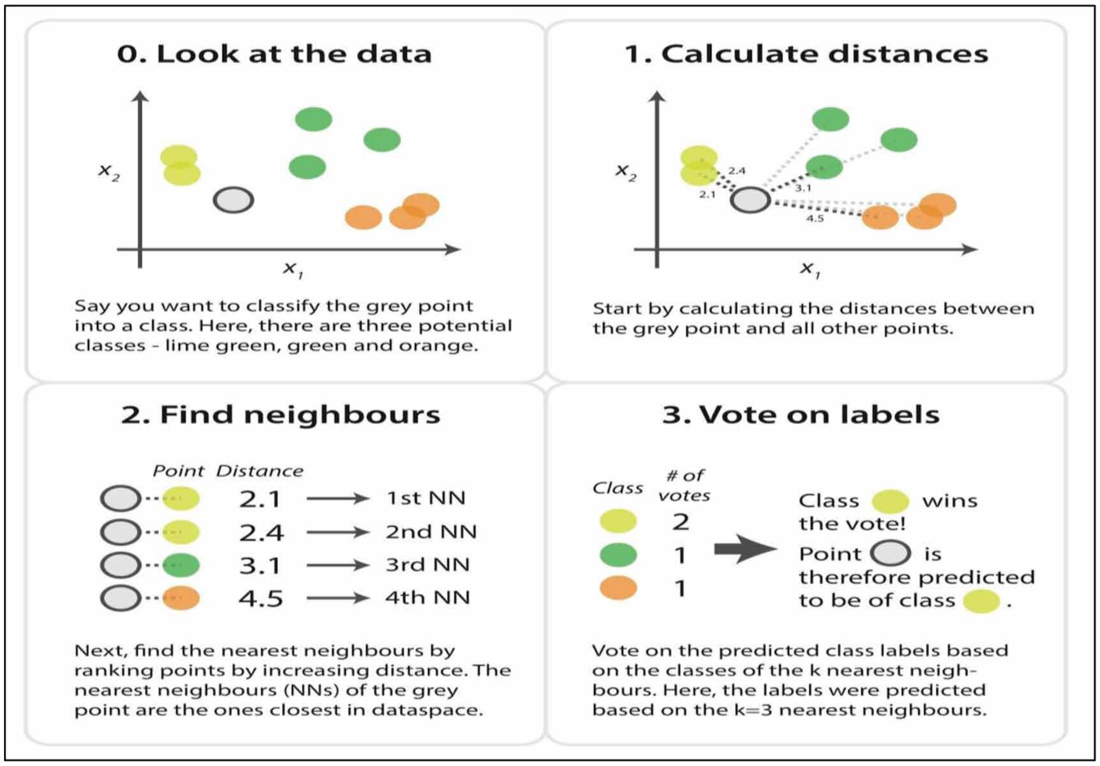
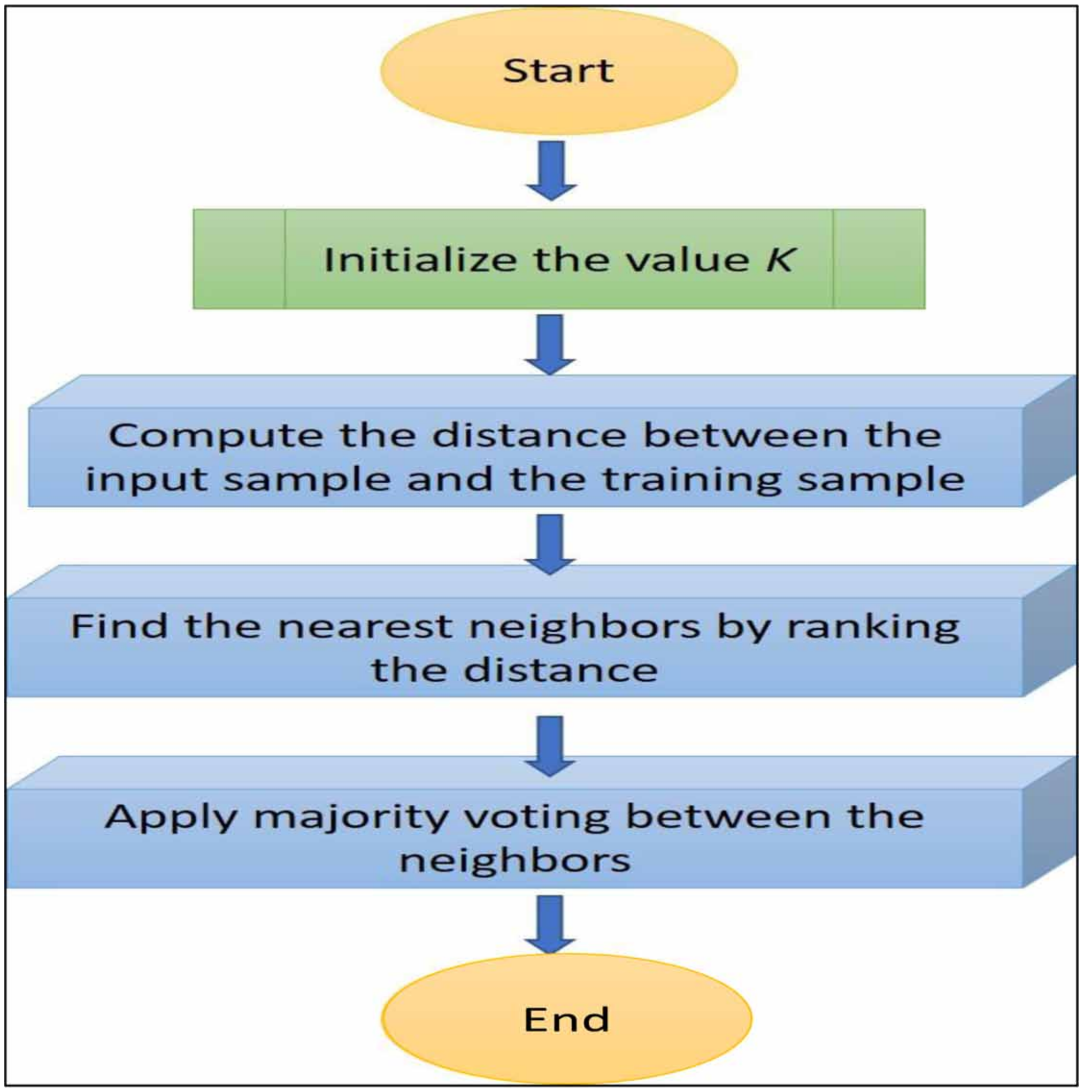

0. Look at the data
x_2
x_1
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.
1. Calculate distances
x_2
2.4
2.1
3.1
4.5
x_1
Start by calculating the distances between the grey point and all other points.
2. Find neighbours
Point Distance
2.1 → 1st NN
2.4 → 2nd NN
3.1 → 3rd NN
4.5 → 4th NN
Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.
3. Vote on labels
Class # of votes
2
1
1
Class wins the vote!
Point is therefore predicted to be of class .
Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

Figure 12. K-NN algorithm flowchart

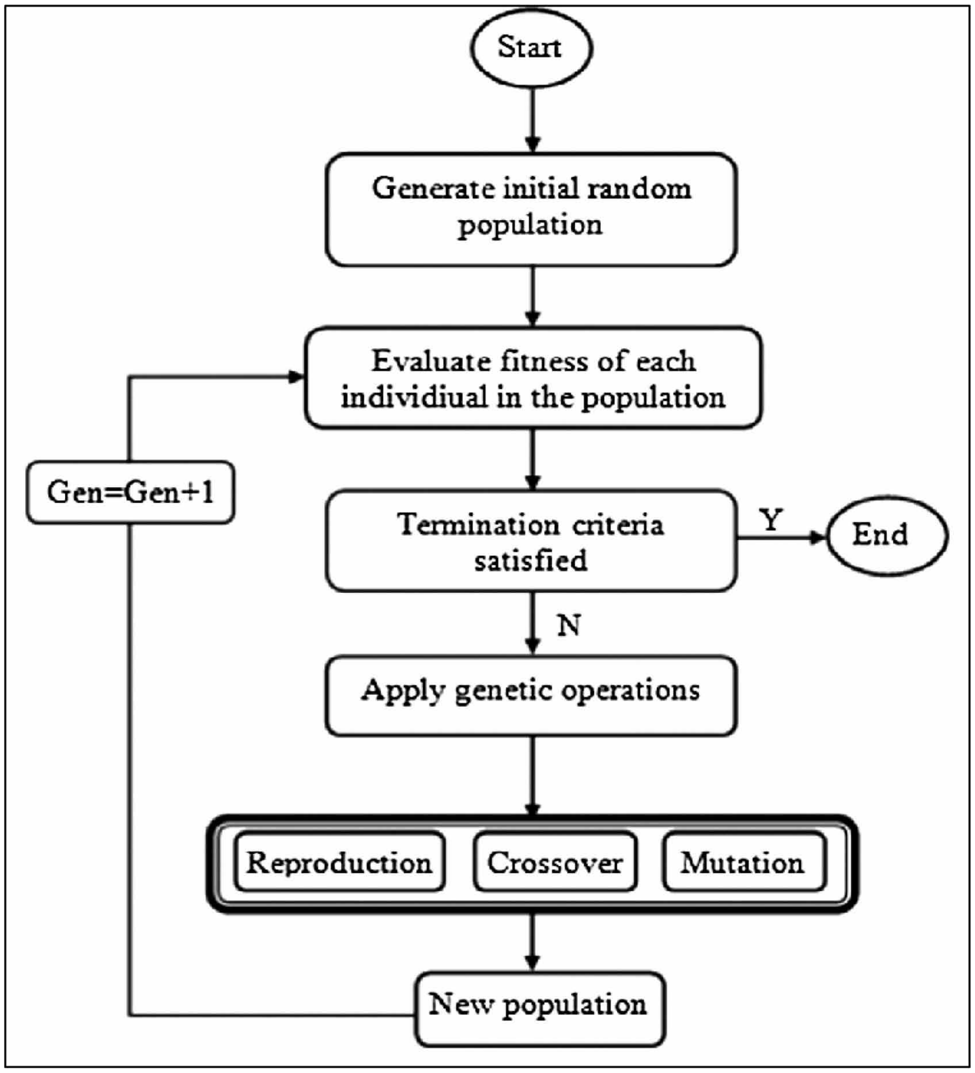

Start
Initialize the value *K*
Compute the distance between the input sample and the training sample
Find the nearest neighbors by ranking the distance
Apply majority voting between the neighbors
End

3.5. Genetic Programming (GP)

GP is an evolutionary algorithm extended from the GA, which simulates natural selection and population dynamics. The key advantage of GP is its simplicity of understanding the structure of the classifier. GP and GA are distinguished by the structure of their population candidates. In GP, the individual structure is a tree, whereas GA is a string (Salem, Attiya, & El-Fishawy, 2017). GP works toward determining quality fitness as a goal and consuming this fitness to develop a set of candidates (i.e., individuals) (Archetti, Giordani, & Vanneschi, 2010). The tree structure of the GP is recursively built using a set of function symbols, $F= \{f_1,f_2,\ldots,f_n\}$, used to label the nodes of the tree and a set of terminal symbols, $T= \{t_1,t_2,\ldots,t_m\}$, used to label the leaves.

GP works by randomly selecting an input (e.g., identifier genes and constant values), which are used to signify the expressions of equivalent genes. These inputs are then combined with function operators (arithmetic or Boolean) to build GP-tree-based classifiers. These tree-based classifiers are accumulated to establish a preliminary population, where a small classifier subgroup is extracted to establish a mating group. Each classifier in a subgroup is assessed by a fitness function. Later, the two fittest classifiers are selected via a tournament selection process as mating pairs for their selective genetic operators (i.e., crossover or mutation) to produce offspring. The generated offspring replaces the smallest fit classifier in the population. Afterwards, a new generation with better classifiers is produced. The entire process of mating selection, fitness evaluation, mating and swapping is recursively repeated until a termination condition is met (Yu et al., 2007). A description of the above GP working flow is demonstrated by the flowchart in Figure 13.

Figure 13. GP flowchart
Source [Adapted from]: (Lim, Karakus, & Ozbakkaloglu, 2016)


Start
Generate initial random population
Evaluate fitness of each individiual in the population
Gen=Gen+1
Termination criteria satisfied
Y
End
N
Apply genetic operations
Reproduction
Crossover
Mutation
New population

4. SYSTEM DEVELOPMENT

The problems presented in Chapter 2 compelled to investigate a better solution for identifying the performance of the common gene selection and cancer classification algorithms relevant to the colon cancer dataset and to implement new methodologies to enhance feature selection and classification accuracy. The case studies (experiments) defined in this research compare the efficiency and measure the ML algorithms with and without gene selection with the colon cancer dataset. The detection results are measured via splitting the input samples into training dataset samples and testing dataset samples. Thus, while running the experiments, the algorithms are trained using the training data samples then utilise the test dataset to evaluate the proposed method.

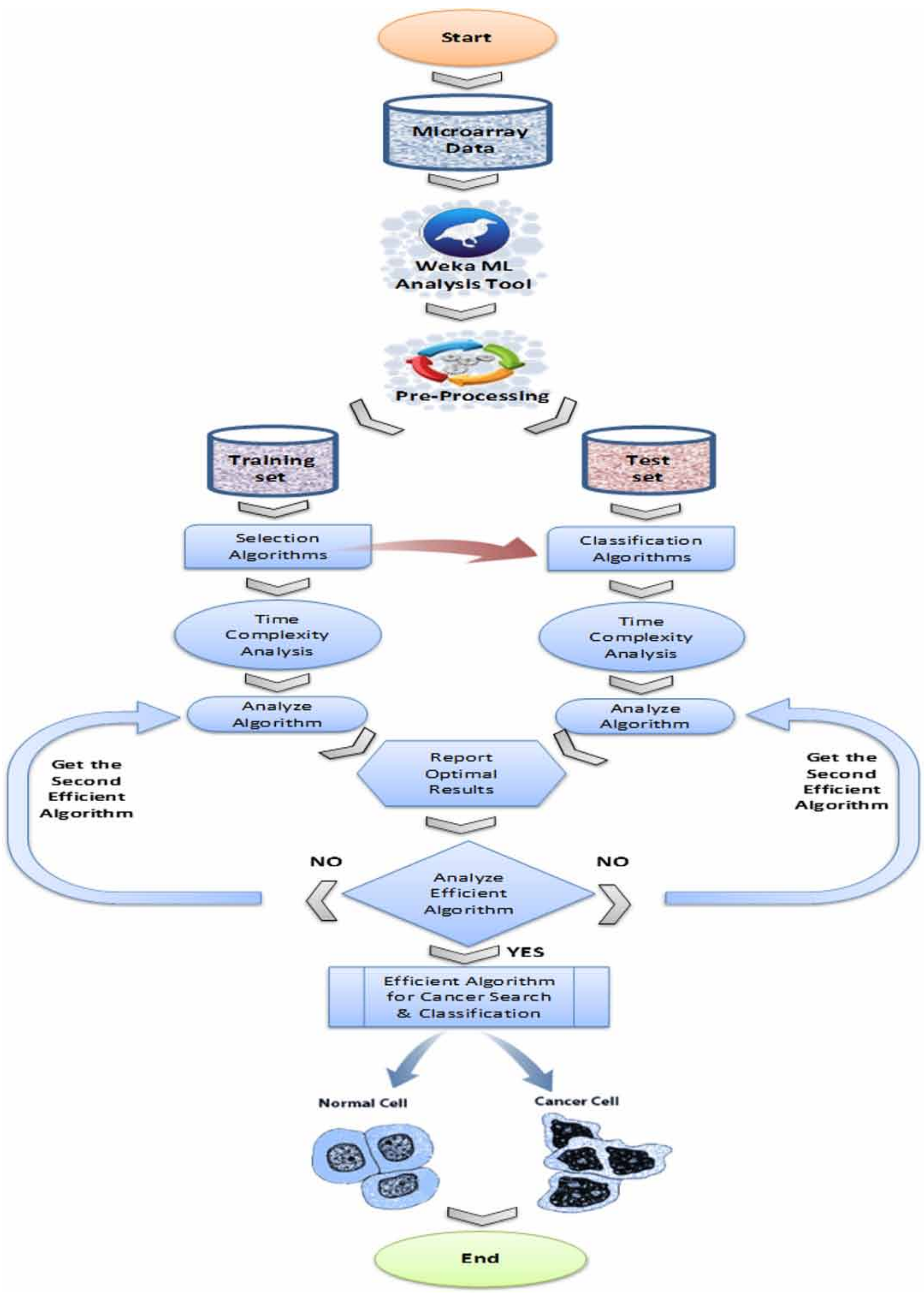
An overview of the general roadmap prototype for a typical gene/feature selection and classification implemented in this research is shown in Figure 14. The system begins by receiving the colon cancer microarray data as input. Then, the data are pre-processed using the Weka ML analysis tool (Witten et al., 2016), discussed in Chapter 4. Data pre-processing includes filtering by threshold values, normalisation and data discretisation. Thereafter, the data is divided into sets of training and testing data. The training data are evaluated first using feature selection algorithms. The system, at this stage, analyses the time complexity for the feature selection algorithm and searches for the optimal solution. If the algorithm shows high time complexity, the same process will be repeated using the same algorithm after adjusting some parameters or by using another selection algorithm until all algorithms are analysed, recording a better efficiency time. However, the same procedure is applied to the classification algorithms. The general process of the classification algorithms is training the classifier by applying the training samples and thereafter classifying the test samples using the classifier that is already trained. The testing dataset is applied to classification algorithms, and the system analyses the time complexity of the ML (i.e., classification) algorithm and searches for the optimal solution. If the algorithm shows high time complexity, the same process will be repeated using a different classification algorithm until all algorithms are analysed, recording a better efficiency time. Finally, efficient algorithms recorded from each selection and classification category are integrated to give the most efficient and effective optimum method for colon-gene selection and cancer classification.

This research proposes two system methodologies (i.e., case studies). One examines the accuracy and efficiency (i.e., time complexity) of high-performance gene selection and cancer classification algorithms for colon cancer, whereas the second implements the proposed two-stage hybrid multifilter gene/feature selection algorithm for the same dataset. Both methodologies are discussed in the following sections.

5. METHODS AND TECHNIQUES FOR GENE SELECTION AND CANCER CLASSIFICATION

Several algorithms have shown success in solving cancer classification problems in the field of ML studies. Some of these are implemented in this research. Chitode and Nagori (2013) defined classification as the procedure of determining a model that discriminates classes of data. Most proposed cancer classification paradigms depend on ML studies (Lu & Han, 2003). In summary, the following points are investigated (Al-Rajab, Lu, & Xu, 2017).

Figure 14. Overview of the general prototype for gene (feature) selection and cancer classification methodology



- DNA microarray technology suggests a system of digitally analysing data. However, it requires regularisation and normalisation for steering experiments.
- Searching and selection algorithms commence the procedure of identifying informative genes. Feature selection can be formally described for a dataset, *D*, of dimension, *d*, with a feature set, $F = (f_1, f_2, \ldots, f_d)$. The problem is to select an optimum subset of related features, F', where $F' \subseteq F$ and F' results in a very good classification rate.
- Classification algorithms categorise the potential data into two states of either 'normal' or 'cancer'. That is, $F'\left(f_d\right) = C\left(f_d\right)$, in which $C(f_d)$ is the accuracy of the classification when the individual, f_d, in the trained data, is used in the feature subset.

6. EXAMINING THE ACCURACY AND EFFICIENCY OF HIGH-PERFORMANCE GENE SELECTION AND CANCER CLASSIFICATION ALGORITHMS

The system methodology, and the flowchart of the proposed prototype are discussed in this section.

This proposed system contains three steps (Al-Rajab, Lu, & Xu, 2017): examine and analyse the performance of gene/feature selection algorithms employed to the colon dataset; analyse the classification algorithm's performance using the same dataset; and analyse the performance of combining or incorporating both gene/feature selection and cancer classification algorithms.

The performance and accuracy of the most common gene selection and cancer classification algorithms applied to colon cancer research is evaluated by the proposed system. The system proposes and distinguishes the relationship between the time required to classify colon tumours and the performance of the algorithm (i.e., accuracy). It proposes an alternative way to compare diverse algorithms using the relationship of accuracy and time efficiency. The author published this proposed system in (Al-Rajab, Lu, & Xu, 2017).

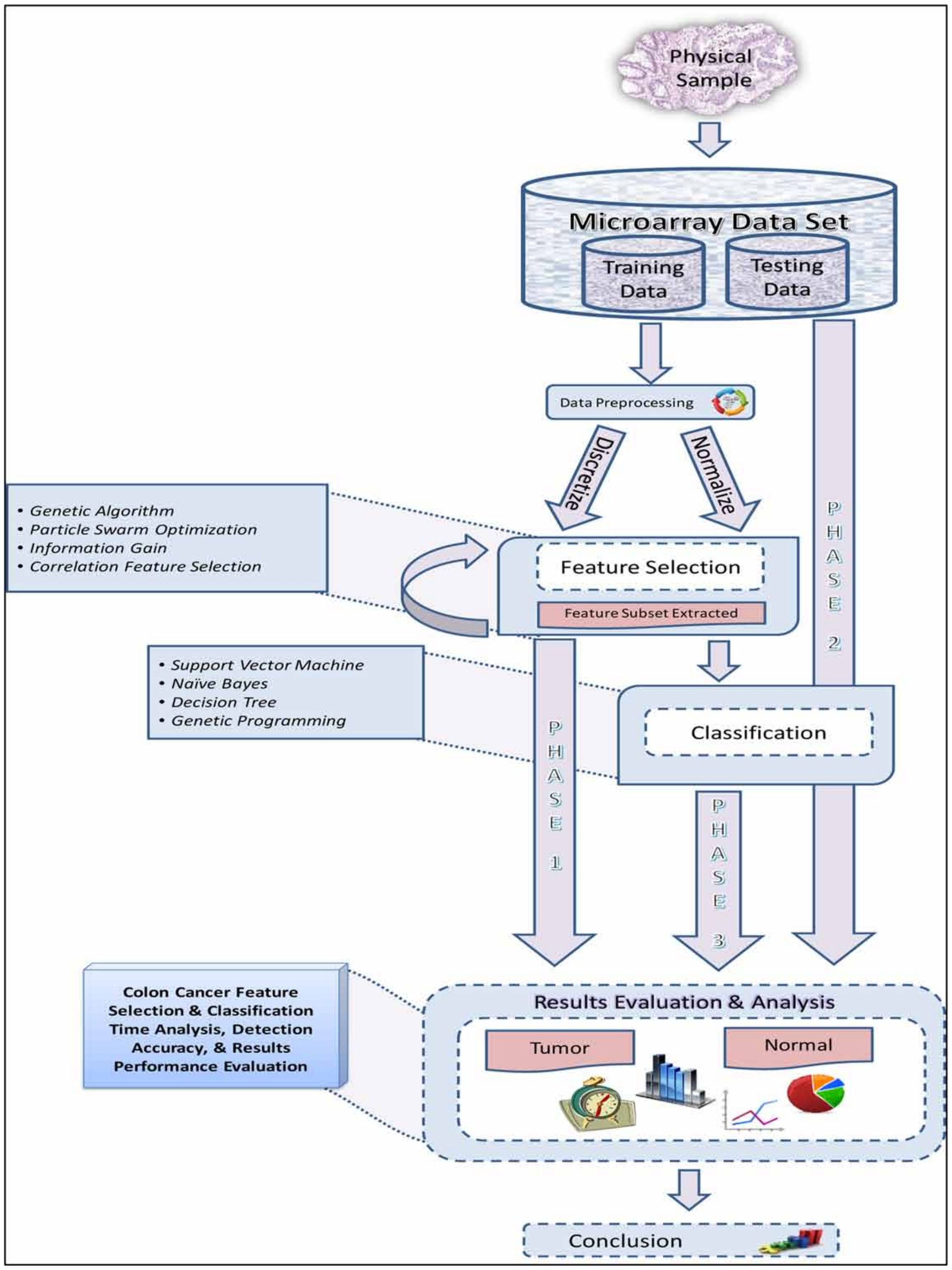
6.1. Proposed System for Case Study One

The flowchart presented in Figure 15 demonstrates the suggested experimental environment conducted to examine accuracy and efficiency (i.e., time complexity) of high-performance gene selection and cancer classification algorithms for colon cancer across three distinct phases. The proposed system is described as follows.

- A physical sample is the system input. It is collected from the colon tissues of patients via physical means inside a medical lab. Data are prepared for insertion into the microarray.
- The microarray gene expression dataset contains a comprehensive view and a measure of all gene expressions generated from the data collected earlier. It is ready to be computerised and analysed.
- After receiving the colon cancer dataset from the DNA microarray technology, the data are preprocessed to reduce data noise and to categorise them to enhance ML classifier outcomes. The system is then implemented in three phases.
    - Phase 1 implements and studies the feature selection algorithms across the dataset to evaluate the best and efficient feature selection. This process is repeated multiple times by adjusting the algorithmic parameters to reach the optimal results.

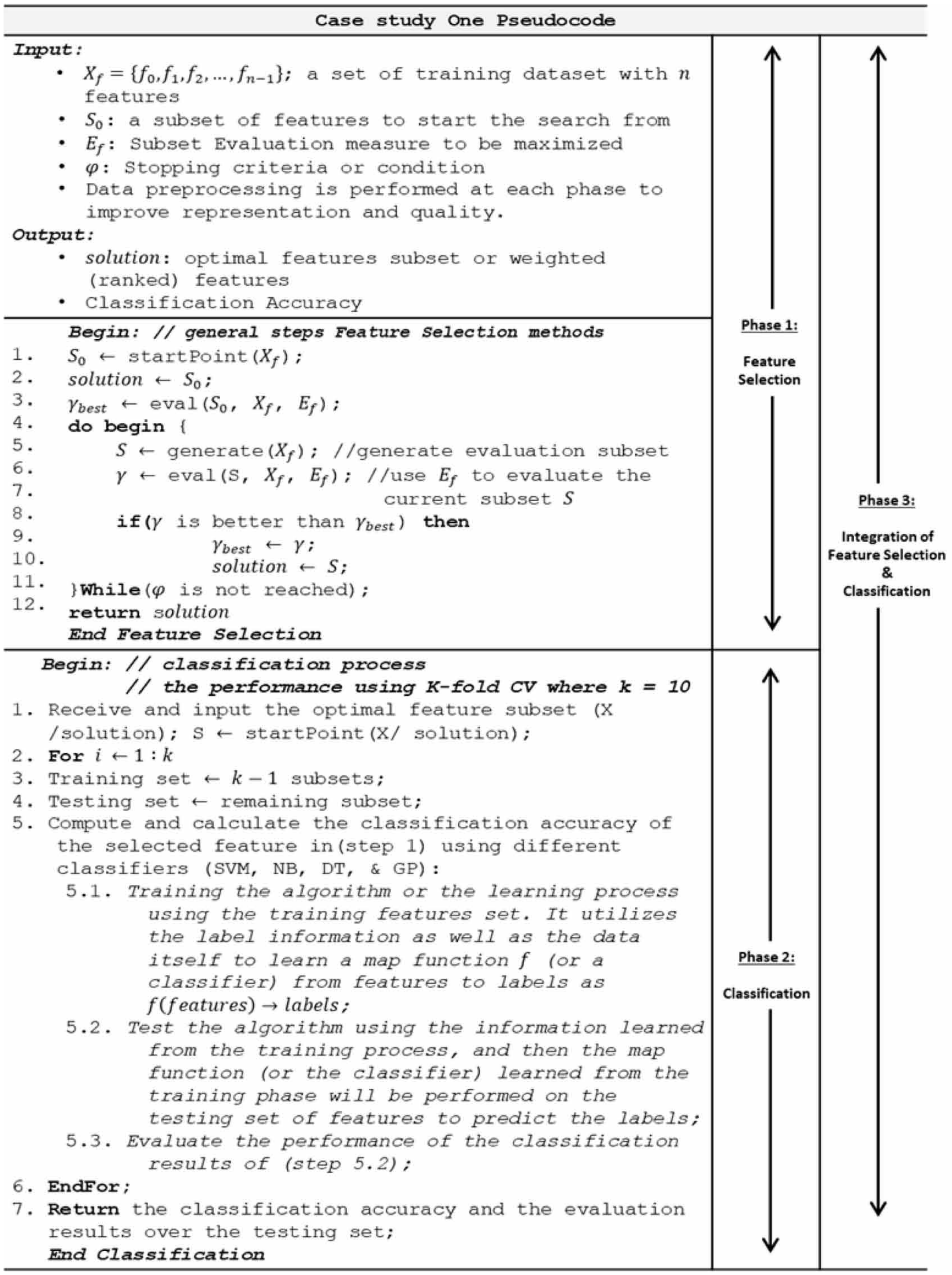
- ◦ Phase 2 implements the classification algorithms using only the original dataset without prior employment of feature selection algorithms to analyse the results.
- ◦ Phase 3 implements the combination technique or incorporates feature selection and classification algorithms to determine the best and most efficient combination of algorithms for detecting colon cancer.

Figure 15. First proposed system workflow for examining high-performance gene selection and cancer classification algorithms



The pseudocode of the proposed model is displayed in Figure 16.

Figure 16. Case study one pseudocode



Case study One Pseudocode

```
Input:
    • $X_f = \{f_0, f_1, f_2, \ldots, f_{n-1}\}$; a set of training dataset with $n$
      features
    • $S_0$: a subset of features to start the search from
    • $E_f$: Subset Evaluation measure to be maximized
    • $\varphi$: Stopping criteria or condition
    • Data preprocessing is performed at each phase to
      improve representation and quality.
Output:
    • $solution$: optimal features subset or weighted
      (ranked) features
    • Classification Accuracy

     Begin: // general steps Feature Selection methods
1.   $S_0 \leftarrow$ startPoint($X_f$);
2.   $solution \leftarrow S_0$;
3.   $\gamma_{best} \leftarrow$ eval($S_0$, $X_f$, $E_f$);
4.   do begin {
5.       $S \leftarrow$ generate($X_f$); //generate evaluation subset
6.       $\gamma \leftarrow$ eval(S, $X_f$, $E_f$); //use $E_f$ to evaluate the
7.                                     current subset $S$
8.       if($\gamma$ is better than $\gamma_{best}$) then
9.               $\gamma_{best} \leftarrow \gamma$;
10.              $solution \leftarrow S$;
11.  }While($\varphi$ is not reached);
12.  return $solution$
     End Feature Selection

   Begin: // classification process
          // the performance using K-fold CV where k = 10
1. Receive and input the optimal feature subset (X
   /solution); S ← startPoint(X/ solution);
2. For $i \leftarrow 1 : k$
3. Training set ← $k - 1$ subsets;
4. Testing set ← remaining subset;
5. Compute and calculate the classification accuracy of
    the selected feature in(step 1) using different
    classifiers (SVM, NB, DT, & GP):
     5.1. Training the algorithm or the learning process
            using the training features set. It utilizes
            the label information as well as the data
            itself to learn a map function $f$ (or a
            classifier) from features to labels as
            $f(features) \rightarrow labels$;
     5.2. Test the algorithm using the information learned
            from the training process, and then the map
            function (or the classifier) learned from the
            training phase will be performed on the
            testing set of features to predict the labels;
     5.3. Evaluate the performance of the classification
            results of (step 5.2);
6. EndFor;
7. Return the classification accuracy and the evaluation
   results over the testing set;
   End Classification
```

7. A 2-STAGE HYBRID MULTIFILTER FEATURE SELECTION METHOD FOR HIGH COLON-CANCER CLASSIFICATION

Hybridisation of feature selection has been explored in the literature. However, the idea proposed in this research is a hybrid 2-stage multifilter incorporated for gene selection, comprising filtering and ranking methods that enhance the classification accuracy using previous classification results and decrease the number of genes selected. This method achieves the goal of classification by enhancing the classification accuracy and explores the biological information between relevant genes as well. This method is less computationally complex compared to the wrapper method. The problem with existing gene ranking and selection methods is the huge number of genes that are not related and are redundant. Thus, a better selection strategy is needed to reduce the noise and omit only related and informative genes from the selected subset. The proposed system differs from previous studies, which considered only 1-stage hybrid selection approaches as presented in (Chapter 2). The proposed system is structured along three sections: data input, gene selection and data classification. Gene search and selection is the most significant part of the proposed system.

7.1. Case Study Two Proposed System

The 2-stage selection system is proposed with the following components, as shown in Figure 17.

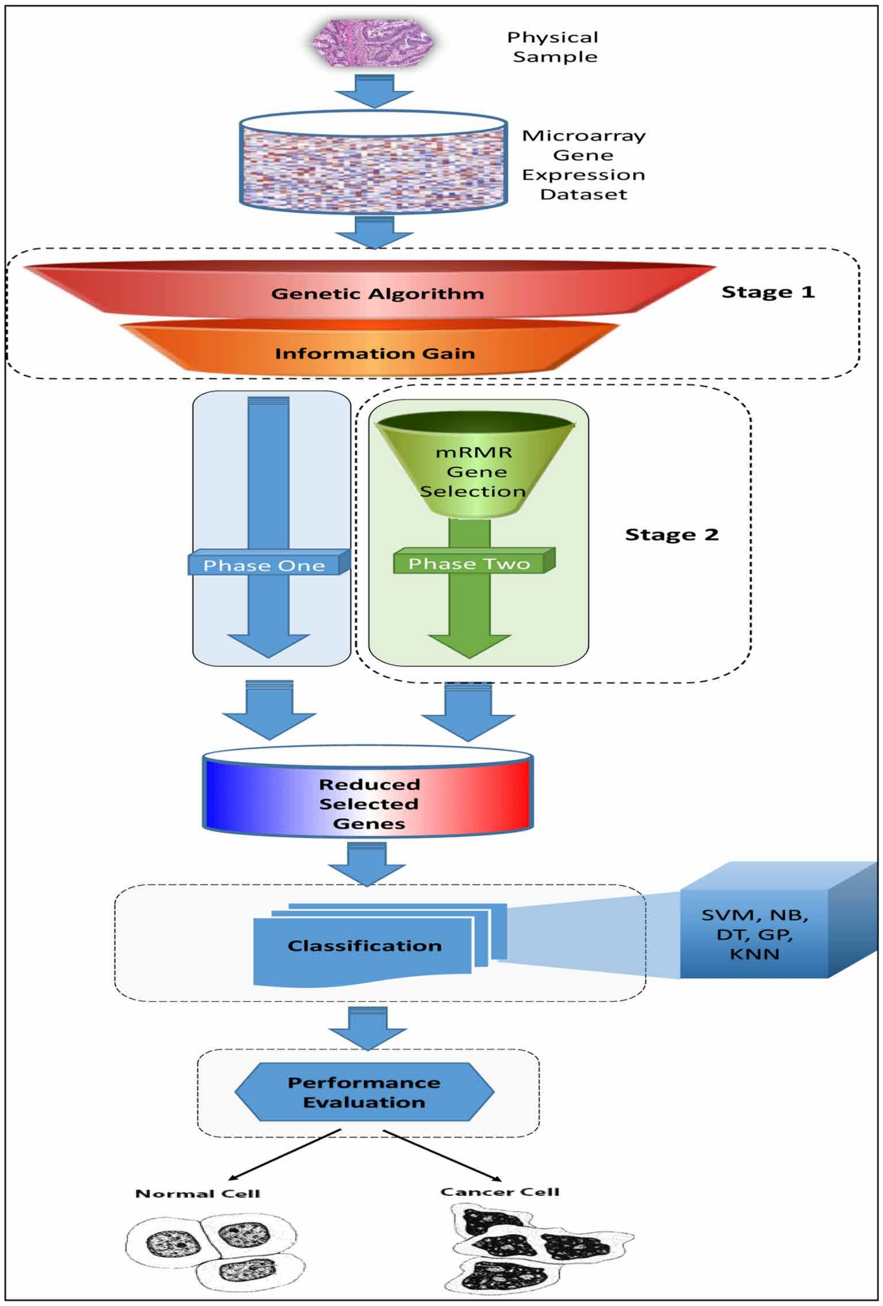
- A physical sample is the system input. It is collected from the colon tissues of patients via physical means inside a medical lab. Data is prepared for insertion into the microarray.
- The microarray gene expression dataset contains a comprehensive view and a measure of all gene expressions generated from the data collected earlier. It is ready to be computerised and analysed.
- After receiving the colon cancer dataset from the DNA microarray technology, the data are preprocessed to reduce data noise and to categorise them to enhance ML classifier outcomes.
    - Phase 1 implements a hybrid selection technique (i.e., first selection stage) using the GA and the information gain algorithms.
    - Phase 2 implements the selection procedure, comprising the hybridisation of phase 1 and the mRMR method.
- A variety of classification algorithms using SVM, NB, GP, DT and K-NN are applied to assess the performance of the subset generated from both phases and determine the prediction accuracy. Finally, the output of the model assists detection and distinction of colon cells as either normal or cancer.

The proposed model description is presented below and the pseudocode is elaborated in Figure 18. The goal is to enhance the feature selection process to select a subset of more efficient and related genes that can distinguish the samples of different classes into tumour or normal cells. Given a set of features, $F = \{(i) \mid i = 1,2,3,\ldots,n\}$, the feature selection process finds a feature subset in which $F' \subseteq F$, maximising a scoring target function, Θ: Γ®Z, where Γ represents the space for all probable subsets of features of *F*, and *Z* is a subset of Γ.

Assuming the m-dimensional dataset is input into the feature selection method of stage one (i.e., GA), the data values are represented by a 2-D matrix, $(Data_{n \times m})$, where *n* is the total number of data samples (i.e., patients), and *m* is the total number of features (i.e., genes) in the dataset. The aim of the

2-stage multifilter feature selection is to select an effective and optimal feature subset from the pool of available features. Let us suppose that the original feature set of m-dimensions is $X=\{x(i)|\ i=1,2,3,\ldots,m\}$. The goal of the first stage of feature selection using GA is finding a new reduced feature vector, $Y=\{y(i)|\ i=1,2,3,\ldots,p\}$. Y should be an effective and optimal subset of the original dataset, X. Thus, $Y \subset X$, and p£m.

Figure 17. Proposed computational framework model for the 2-stage hybrid feature selection model



Now, the IG attribute evaluator is computed over the new subset, *Y*, for each feature, $y(i) \in Y$, according to Equation 3-12. Then, the features are sorted from highest to lowest information-gain value. The features with the highest gain, greater than the threshold value (0 in this model), is selected, and the *Y* subset is refined to form a new feature vector, $Z = \{Z(i) | i=1,2,3,\ldots,q\}$. Normally, an ML feature with higher information gain is ranked higher than other features, because it has a stronger effective power in classifying data. However, $Z \subseteq Y$, and *q£p*. Testing each feature individually is the main drawback of IG filtering. Thus, the correlation between features may be discarded. The mRMR method takes advantage of the search for high relevant and correlated features showing minimum redundancy. Given the last dataset, *Z*, the mRMR selects features with minimum redundancy having maximum relevance to the class problem, producing a new feature vector set, $A = \{A(i) | i=1,2,3,\ldots,s\}$. Thus, $A \subseteq Z$, and *s£q*. Next, the optimal selected feature subset, *A*, is input to the classification algorithm, which classifies the data into two labels, {-1,1} (i.e., tumour or normal). After gene selection, the dataset is defined as $\left\{\left(\ell\left(A_i\right), C_i\right)\right\}_{i=1}^{l} = \ell\left(\mathcal{D}\right)$, with $\ell\left(A_i\right) \in \mathfrak{R}^{m'}$, where the function, ℓ, selects $m' < m$ gene features across all *n* genes from the gene expression microarray dataset, $\mathcal{D}$.

In summary, before feature selection, the classification algorithm is considered a function *F*: *X®C*. After the 2-stage hybrid multifilter selection, the classifier function is *F*: *A®C*.

There are common evaluation performance measures applied to classification results to determine the effectiveness of the classification algorithms that are employed, besides the 10-cross validation technique (Chapter 4) for testing the classification results. These evaluation metrics were designed to evaluate a binary classification test performance. A list of different measures employed to validate the results of the research case studies are discussed below (Fawcett, 2006; Gu, Zhu, & Cai, 2009; James et al., 2013; Tharwat, 2018).

(1) Confusion Matrix is a very simple way of presenting ML classifier performance. A 2-class problem can use the confusion matrix to evaluate and predict the test set instances of a classifier using the matrix in Figure 19. True positives (TP) are the number of positive samples diagnosed properly, true negatives (TN) are the number of negative samples diagnosed properly, false negatives (FN) are the number of positive samples identified as negative, and false positives (FP) are the number of negative samples recognised as positive.

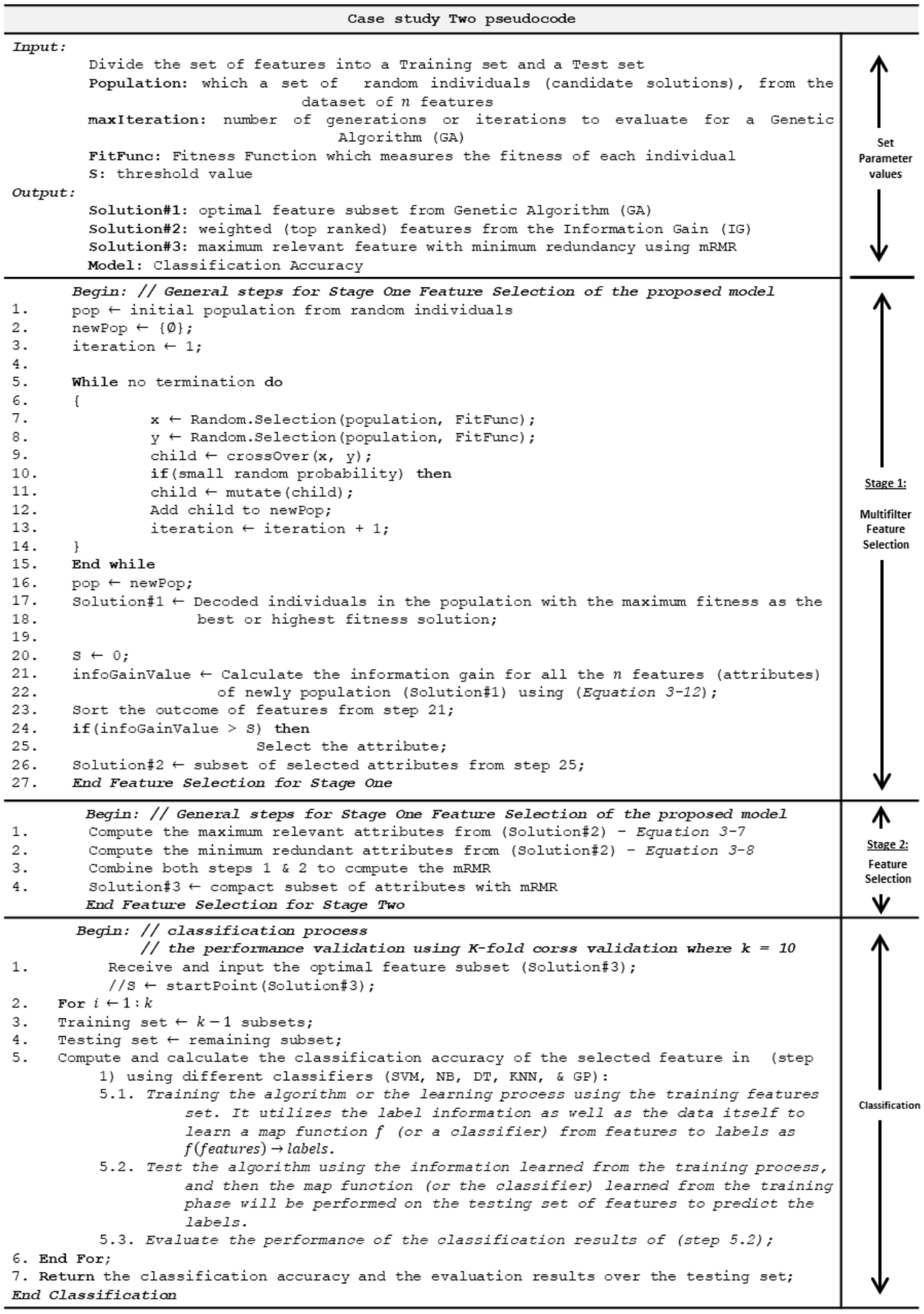
(2) Accuracy measures the total effectiveness of the classifier. It is the percentage calculated by dividing the instances (i.e., samples) that are correctly classified by the overall number of instances, and it can be calculated from the confusion matrix using the following formula (Sokolova, Japkowicz, & Szpakowicz, 2006).

$$\boldsymbol{Accuracy} = \frac{\textbf{\textit{Total number of correctly predicted record for all the runs}}}{\textbf{\textit{Total number of pedicted records}}} \tag{19}$$

$$\boldsymbol{Classification\ Accuracy}\left(\boldsymbol{CA}\right) = \frac{\boldsymbol{TN + TP}}{\boldsymbol{TN + TP + FN + FP}} \times 100 \tag{20}$$

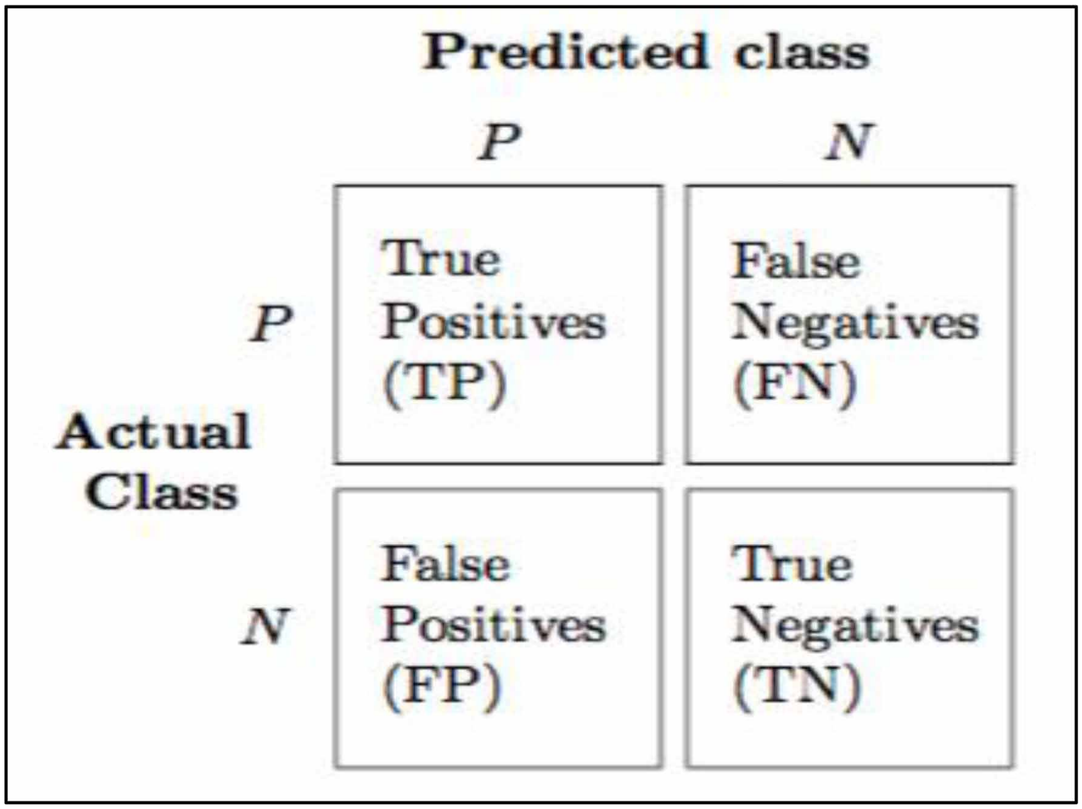
$$\boldsymbol{Error} = \frac{\boldsymbol{FP + FN}}{\boldsymbol{TP + FP + TN + FN}} \tag{21}$$

Figure 18. Case study 2 pseudocode



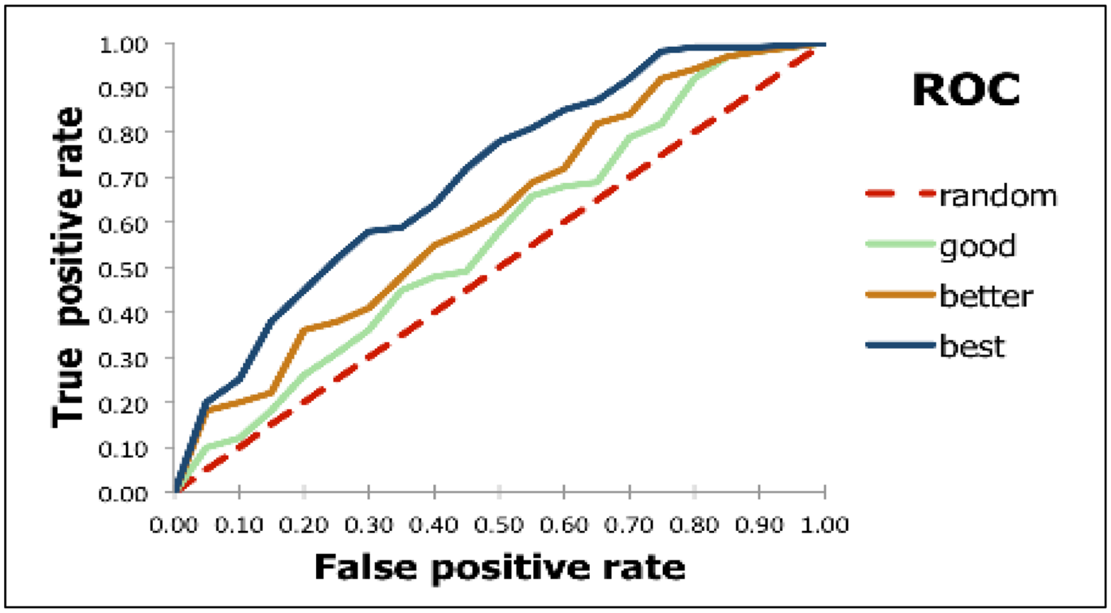
```
Case study Two pseudocode

Input:
        Divide the set of features into a Training set and a Test set
        Population: which a set of random individuals (candidate solutions), from the
                    dataset of n features
        maxIteration: number of generations or iterations to evaluate for a Genetic
                    Algorithm (GA)
        FitFunc: Fitness Function which measures the fitness of each individual
        S: threshold value
Output:
        Solution#1: optimal feature subset from Genetic Algorithm (GA)
        Solution#2: weighted (top ranked) features from the Information Gain (IG)
        Solution#3: maximum relevant feature with minimum redundancy using mRMR
        Model: Classification Accuracy

      Begin: // General steps for Stage One Feature Selection of the proposed model
1.    pop ← initial population from random individuals
2.    newPop ← {∅};
3.    iteration ← 1;
4.
5.    While no termination do
6.    {
7.            x ← Random.Selection(population, FitFunc);
8.            y ← Random.Selection(population, FitFunc);
9.            child ← crossOver(x, y);
10.           if(small random probability) then
11.           child ← mutate(child);
12.           Add child to newPop;
13.           iteration ← iteration + 1;
14.   }
15.   End while
16.   pop ← newPop;
17.   Solution#1 ← Decoded individuals in the population with the maximum fitness as the
18.                best or highest fitness solution;
19.
20.   S ← 0;
21.   infoGainValue ← Calculate the information gain for all the n features (attributes)
22.                of newly population (Solution#1) using (Equation 3-12);
23.   Sort the outcome of features from step 21;
24.   if(infoGainValue > S) then
25.                    Select the attribute;
26.   Solution#2 ← subset of selected attributes from step 25;
27.   End Feature Selection for Stage One

      Begin: // General steps for Stage One Feature Selection of the proposed model
1.     Compute the maximum relevant attributes from (Solution#2) - Equation 3-7
2.     Compute the minimum redundant attributes from (Solution#2) - Equation 3-8
3.     Combine both steps 1 & 2 to compute the mRMR
4.     Solution#3 ← compact subset of attributes with mRMR
      End Feature Selection for Stage Two

      Begin: // classification process
            // the performance validation using K-fold corss validation where k = 10
1.       Receive and input the optimal feature subset (Solution#3);
         //S ← startPoint(Solution#3);
2.   For i ← 1 : k
3.   Training set ← k − 1 subsets;
4.   Testing set ← remaining subset;
5.   Compute and calculate the classification accuracy of the selected feature in (step
        1) using different classifiers (SVM, NB, DT, KNN, & GP):
        5.1. Training the algorithm or the learning process using the training features
             set. It utilizes the label information as well as the data itself to
             learn a map function f (or a classifier) from features to labels as
             f(features) → labels.
        5.2. Test the algorithm using the information learned from the training process,
             and then the map function (or the classifier) learned from the training
             phase will be performed on the testing set of features to predict the
             labels.
        5.3. Evaluate the performance of the classification results of (step 5.2);
6. End For;
7. Return the classification accuracy and the evaluation results over the testing set;
End Classification
```

Figure 19. A 2-class confusion matrix



		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Figure 20. ROC graph pattern example
Source [Adapted from]: (OpenEye, 2018) 'Example of ROC curves' [Image]. Retrieved from https://docs.eyesopen.com/toolkits/cookbook/python/plotting/roc.html



(3) Sensitivity or Recall or TP measures the classifier ratio of real positive instances (i.e., samples) that are predicted correctly as positive. Sensitivity specifies how well the testing data predict the actual positives (e.g., the patients' percentage who are properly recognised as having the cancer disease), as illustrated by (Bolón-Canedo et al., 2014). It can be calculated from the confusion matrix using the following formula.

$$\textbf{\textit{Sensitivity}} = \frac{\textbf{\textit{TP}}}{\textbf{\textit{TP}} + \textbf{\textit{FN}}} \tag{22}$$

(4) Specificity or TN measures the ratio of real negative instances that are correctly predicted to be negative for a classifier. It computes how well the test determines the negatives (e.g., the patients' percentages that are properly recognised without cancer). It can be calculated from the confusion matrix using the following formula.

$$\textbf{\textit{Specificity}} = \frac{\textbf{\textit{TN}}}{\textbf{\textit{TN}} + \textbf{\textit{FP}}} \tag{23}$$

Thus, both sensitivity and specificity approximate the percentage of positive and negative instances or samples that are correctly classified by the ML classifier (Rathore, Iftikhar, & Hassan, 2016). An ideal predictor would be identified as 100% sensitive (e.g., estimating all patients who have cancer) or 100% specific (e.g., not estimating any healthy patient as having cancer). Accuracy measures how well the test detects both predictors. Sensitivity, specificity and accuracy can be defined using the confusion matrix in recognition of TP, TN, FN and FP.

(5) Matthews's Correlation Coefficient (MCC) was first expressed in 1975 (Matthews, 1975; Tharwat, 2018) and is used to measure the classification of binary class problems. The value of MCC ranges between -1 and +1. A value of +1 suggests that the classifier always predicts the correct label, whereas a value of -1 suggests that the classifier always commits a mistake. However, a value of 0 identifies a random prediction. The MCC can be computed form the confusion matrix as follows.

$$\textbf{\textit{MCC}} = \frac{\textbf{\textit{TP}} \times \textbf{\textit{TN}} - \textbf{\textit{FP}} \times \textbf{\textit{FN}}}{\sqrt{\left(\left(\textbf{\textit{TP}} + \textbf{\textit{FN}}\right)\left(\textbf{\textit{TP}} + \textbf{\textit{FP}}\right)\left(\textbf{\textit{TN}} + \textbf{\textit{FN}}\right)\left(\textbf{\textit{TN}} + \textbf{\textit{FP}}\right)\right)}} \tag{24}$$

(6) Receiver Operating Characteristic (ROC) Area measures the performance of a binary classifier with a graph that discriminates between the TP and the FP for each classification threshold. It is a graphical representation that visualises and selects the best classifier based on the performance (Fawcett, 2006). The graph is plotted on a 2-D axis. The *x-axis* contains the FP rate (FPR), which is known as the sensitivity (i.e., recall), and the *y-axis* contains the TP rate (TPR), which can be calculated as (1-*Specificity*). It is considered a vital parameter to measure the best classifier, depending on performance. The best prediction is the point having the possibility to be at the coordinate of (0, 1), reflecting a sensitivity of 100%. This implies no false negatives and 100% specificity. This coordinate is known as the perfect or best classification. The ROC curve has a diagonal line that divides the curve from the left-bottom of the coordinate to the top-right corner, called the line of non-discrimination. Thus, points located above the diagonal line are good classifiers, whereas those below the line are bad classifiers. Figure 20 presents an example of an ROC graph.

REFERENCES

Abd Rahman, R., Ramli, R., Jamari, Z., & Ku-Mahamud, K. R. (2016). Evolutionary Algorithm with Roulette-Tournament Selection for Solving Aquaculture Diet Formulation. *Mathematical Problems in Engineering*, *5*, 1–10. doi:10.1155/2016/3672758

Abdi, M. J., Hosseini, S. M., & Rezghi, M. (2012). A Novel Weighted Support Vector Machine Based on Particle Swarm Optimization for Gene Selection and Tumor Classification. *Computational and Mathematical Methods in Medicine*, *2012*, 1–7. doi:10.1155/2012/320698 PMID:22924059

Ahmed, T., Md Siraj, M., Zainal, A., Elshoush, H., & Elhaj, F. (2016). Feature Selection Using Information Gain for Improved Structural-Based Alert Correlation. *PLoS One*, *11*(11), 1–18. doi:10.1371/journal.pone.0166017 PMID:27893821

Al-Rajab, M., Lu, J., & Xu, Q. i. (2017). Examining Applying High Performance Genetic Data Feature Selection and Classification Algorithms for Colon Cancer Diagnosis. *Computer Methods and Programs in Biomedicine*, *146*, 11–24. doi:10.1016/j.cmpb.2017.05.001 PMID:28688481

Al-Rajab, M. M., & Lu, J. (2014). *Algorithms Implemented for Cancer Gene Searching and Classifications*. Paper presented at the International Symposium on Bioinformatics Research and Applications, Zhangjiajie, China. 10.1007/978-3-319-08171-7_6

Al-Rajab, M. M., & Lu, J. (2016). A Study on the Most Common Algorithms Implemented for Cancer Gene Search and Classifications. *International Journal of Data Mining and Bioinformatics*, *14*(2), 159–176. doi:10.1504/IJDMB.2016.074685

Al Snousy, M. B., El-Deeb, H. M., Badran, K., & Al Khlil, I. A. (2011). Suite of Decision Tree-Based Classification Algorithms on Cancer Gene Expression Data. *Egyptian Informatics Journal*, *12*(2), 73–82. doi:10.1016/j.eij.2011.04.003

Alba, E., Garcia-Nieto, J., Jourdan, L., & Talbi, E.-G. (2007). *Gene Selection in Cancer Classification Using Pso/Svm and Ga/Svm Hybrid Algorithms*. Paper presented at the IEEE Congress on Evolutionary Computation (CEC), Singapore, Singapore. 10.1109/CEC.2007.4424483

Alshamlan, H., Badr, G., & Alohali, Y. (2015). Mrmr-Abc: A Hybrid Gene Selection Algorithm for Cancer Classification Using Microarray Gene Expression Profiling. *BioMed Research International*, *2015*, 1–15. doi:10.1155/2015/604910 PMID:25961028

Ammu, P., & Preeja, V. (2013). Review on Feature Selection Techniques of DNA Microarray Data. *International Journal of Computers and Applications*, *61*(12), 39–44. doi:10.5120/9983-4814

Archetti, F., Giordani, I., & Vanneschi, L. (2010). Genetic Programming for Anticancer Therapeutic Response Prediction Using the Nci-60 Dataset. *Computers & Operations Research*, *37*(8), 1395–1405. doi:10.1016/j.cor.2009.02.015

Babu, M., & Sarkar, K. (2016). *A Comparative Study of Gene Selection Methods for Cancer Classification Using Microarray Data*. Paper presented at the Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India. 10.1109/ICRCICN.2016.7813657

Bennet, J., Ganaprakasam, C., & Kumar, N. (2015). A Hybrid Approach for Gene Selection and Classification Using Support Vector Machine. *The International Arab Journal of Information Technology*, *12*(6A), 695–700.

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2010). *On the Effectiveness of Discretization on Gene Selection of Microarray Data*. Paper presented at the 2010 International Joint Conference on Neural networks (IGCNN), Barcelona, Spain. 10.1109/IJCNN.2010.5596825

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2015). Distributed Feature Selection: An Application to Microarray Data Classification. *Applied Soft Computing*, *30*, 136–150. doi:10.1016/j.asoc.2015.01.035

Bolón-Canedo, V., Sánchez-Marono, N., Alonso-Betanzos, A., Benítez, J. M., & Herrera, F. (2014). A Review of Microarray Datasets and Applied Feature Selection Methods. *Information Sciences*, *282*, 111–135. doi:10.1016/j.ins.2014.05.042

Chantar, H. K. H. (2013). *New Techniques for Arabic Document Classification* (PhD Thesis). Heriot-Watt University.

Chitode, K., & Nagori, M. (2013). A Comparative Study of Microarray Data Analysis for Cancer Classification. *International Journal of Computers and Applications*, *81*(15), 14–18. doi:10.5120/14198-2392

Cho, S.-B., & Ryu, J. (2002). Classifying Gene Expression Data of Cancer Using Classifier Ensemble with Mutually Exclusive Features. *Proceedings of the IEEE*, *90*(11), 1744–1753. doi:10.1109/JPROC.2002.804682

Chuang, L.-Y., Ke, C.-H., Chang, H.-W., & Yang, C.-H. (2009). A Two-Stage Feature Selection Method for Gene Expression Data. *OMICS: A Journal of Integrative Biology*, *13*(2), 127–137. doi:10.1089/omi.2008.0083 PMID:19182978

Chuang, L.-Y., Ke, C.-H., & Yang, C.-H. (2008). A Hybrid Both Filter and Wrapper Feature Selection Method for Microarray Classification. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 146-150.

Dang, V. Q. (2014). *Evolutionary Approaches for Feature Selection in Biological Data* (PhD Thesis). Edith Cowan University, Perth, Australia.

Dara, S., & Tiwari, A. K. (2017). Application of Optimization Techniques for Gene Expression Data Analysis. In P. Kumar & A. Tiwari (Eds.), *Ubiquitous Machine Learning and Its Applications* (pp. 168–180). IGI Global. doi:10.4018/978-1-5225-2545-5.ch008

Dash, S., Patra, B., & Tripathy, B. (2012). Study of Classification Accuracy of Microarray Data for Cancer Classification Using Multivariate and Hybrid Feature Selection Method. *IOSR Journal of Engineering*, *2*(8), 112–119. doi:10.9790/3021-0281112119

El Akadi, A., Amine, A., El Ouardighi, A., & Aboutajdine, D. (2009a). Feature Selection for Genomic Data by Combining Filter and Wrapper Approaches. *INFOCOMP Journal of Computer Science*, *8*(4), 28–36.

El Akadi, A., Amine, A., El Ouardighi, A., & Aboutajdine, D. (2009b). *A New Gene Selection Approach Based on Minimum Redundancy-Maximum Relevance (Mrmr) and Genetic Algorithm (Ga)*. Paper presented at the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2009), Rabat, Morocco. 10.1109/AICCSA.2009.5069306

El-Garhy, A., Amer, F., Awadalla, M., Rashad, S., & Abdien, A. (2011). A Real-Valued Genetic Algorithm to Optimize the Parameters of Support Vector Machine for Classification of Multiple Faults in Npp. *Nukleonika*, *56*, 323–332.

Elyasigomari, V., Lee, D., Screen, H. R., & Shaheed, M. H. (2017). Development of a Two-Stage Gene Selection Method That Incorporates a Novel Hybrid Approach Using the Cuckoo Optimization Algorithm and Harmony Search for Cancer Classification. *Journal of Biomedical Informatics*, *67*, 11–20. doi:10.1016/j.jbi.2017.01.016 PMID:28163197

Fawcett, T. (2006). An Introduction to Roc Analysis. *Pattern Recognition Letters*, *27*(8), 861–874. doi:10.1016/j.patrec.2005.10.010

Fiori, A. (2010). *Extraction of Biological Knowledge by Means of Data Mining Techniques* (PhD Thesis). Polytechnic of Turin.

George, G., & Raj, V. C. (2011). Review on Feature Selection Techniques and the Impact of Svm for Cancer Classification Using Gene Expression Profile. *International Journal of Computer Science & Engineering Survey*, *2*(3), 16–26. doi:10.5121/ijcses.2011.2302

Goldberg. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley.

Gu, Q., Zhu, L., & Cai, Z. (2009). *Evaluation Measures of the Classification Performance of Imbalanced Data Sets*. Paper presented at the International Symposium on Intelligence Computation and Applications (ISICA 2009), Huangshi, China. 10.1007/978-3-642-04962-0_53

Hall, M. A. (1999). *Correlation-Based Feature Selection for Machine Learning* (PhD Thesis). The University of Waikato, Hamilton, New Zealand.

Horng, J.-T., Wu, L.-C., Liu, B.-J., Kuo, J.-L., Kuo, W.-H., & Zhang, J.-J. (2009). An Expert System to Classify Microarray Gene Expression Data Using Gene Selection by Decision Tree. *Expert Systems with Applications*, *36*(5), 9072–9081. doi:10.1016/j.eswa.2008.12.037

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning* (Vol. 112). Springer. doi:10.1007/978-1-4614-7138-7

Janecek, A. (2009). *Efficient Feature Reduction and Classification Methods, Applications in Drug Discovery and Email Categorization* (PhD Thesis). University of Vienna. (A 786 880)

Jeyachidra, J., & Punithavalli, M. (2013). *A Comparative Analysis of Feature Selection Algorithms on Classification of Gene Microarray Dataset*. Paper presented at the International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India. 10.1109/ICICES.2013.6508165

Karthikeyan, T., & Thangaraju, P. (2013). Analysis of Classification Algorithms Applied to Hepatitis Patients. *International Journal of Computers and Applications*, *62*(15), 25–30. doi:10.5120/10157-5032

Kwon, O., & Sim, J. M. (2013). Effects of Data Set Features on the Performances of Classification Algorithms. *Expert Systems with Applications*, *40*(5), 1847–1857. doi:10.1016/j.eswa.2012.09.017

Latysheva, N. (2016). *Implementing Your Own K-Nearest Neighbour Algorithm Using Python*. Retrieved from: https://www.kdnuggets.com/2016/01/implementing-your-own-knn-using-python.html

Lee, E.-J., Kim, Y.-H., Kim, N., & Kang, D.-W. (2017). Deep into the Brain: Artificial Intelligence in Stroke Imaging. *Journal of Stroke*, *19*(3), 277–285. doi:10.5853/jos.2017.02054 PMID:29037014

Lei, S. (2012). *A Feature Selection Method Based on Information Gain and Genetic Algorithm*. Paper presented at the International Conference on Computer Science and Electronics Engineering (ICCSEE), Hangzhou, China. 10.1109/ICCSEE.2012.97

Li, T., Zhang, C., & Ogihara, M. (2004). A Comparative Study of Feature Selection and Multiclass Classification Methods for Tissue Classification Based on Gene Expression. *Bioinformatics (Oxford, England)*, *20*(15), 2429–2437. doi:10.1093/bioinformatics/bth267 PMID:15087314

Lim, J. C., Karakus, M., & Ozbakkaloglu, T. (2016). Evaluation of Ultimate Conditions of Frp-Confined Concrete Columns Using Genetic Programming. *Computers & Structures*, *162*, 28–37. doi:10.1016/j.compstruc.2015.09.005

Liu, H., Li, J., & Wong, L. (2002). A Comparative Study on Feature Selection and Classification Methods Using Gene Expression Profiles and Proteomic Patterns. *Genome Informatics*, *13*, 51–60. doi:10.11234/gi1990.13.51 PMID:14571374

Lorena, A. C., Costa, I. G., & de Souto, M. C. (2008). *On the Complexity of Gene Expression Classification Data Sets*. Paper presented at the Eighth International Conference on Hybrid Intelligent Systems (HIS'080), Barcelona, Spain. 10.1109/HIS.2008.163

Lu, Y., & Han, J. (2003). Cancer Classification Using Gene Expression Data. *Information Systems*, *28*(4), 243–268. doi:10.1016/S0306-4379(02)00072-8

Matthews, B. W. (1975). Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405*(2), 442–451. doi:10.1016/0005-2795(75)90109-9 PMID:1180967

OpenEye. (2018). *Example of Roc Curves*. Retrieved from: https://docs.eyesopen.com/toolkits/cookbook/python/plotting/roc.html

Patil, S., Naik, G., & Pai, K. (2014). Survey of Microarray Data Processing for Cancer Sub-Classification. *International Journal of Emerging Technology and Advanced Engineering*, *4*(2), 110–113.

Peng, H., Long, F., & Ding, C. (2005). Feature Selection Based on Mutual Information Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(8), 1226–1238. doi:10.1109/TPAMI.2005.159 PMID:16119262

Platt, J. C. (1999). Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In S. Bernhard, lkopf, J. C. B. Christopher & J. S. Alexander (Eds.), Advances in Kernel Methods (pp. 185-208). MIT Press.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

Rathore, S., Iftikhar, M. A., & Hassan, M. (2016). *Ensemble Sparse Classification of Colon Cancer*. Paper presented at the International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan. 10.1109/FIT.2016.050

Salem, D., Abul Seoud, R., & Ali, H. (2011). A New Gene Selection Technique Based on Hybrid Methods for Cancer Classification Using Microarrays. *International Journal of Bioscience, Biochemistry, Bioinformatics*, *1*(4), 261–266. doi:10.7763/IJBBB.2011.V1.49

Salem, H., Attiya, G., & El-Fishawy, N. (2017). Classification of Human Cancer Diseases by Gene Expression Profiles. *Applied Soft Computing*, *50*, 124–134. doi:10.1016/j.asoc.2016.11.026

Shah, S., & Kusiak, A. (2007). Cancer Gene Search with Data-Mining and Genetic Algorithms. *Computers in Biology and Medicine*, *37*(2), 251–261. doi:10.1016/j.compbiomed.2006.01.007 PMID:16616736

Shi, Y. (2001). Particle Swarm Optimization: Developments, Applications and Resources. *Proceedings of the 2001 Congress on Evolutionary Computation*, 81-86. 10.1109/CEC.2001.934374

Sokolova, Japkowicz, & Szpakowicz. (2006). *Beyond Accuracy, F-Score and Roc: A Family of Discriminant Measures for Performance Evaluation*. Paper presented at the Australasian Joint Conference on Artificial Intelligence, Hobart, Australia. doi:10.1007/11941439_114

Sujatha, G., & Rani, K. U. (2013). Evaluation of Decision Tree Classifiers on Tumor Datasets. *International Journal of Emerging Trends & Technology in Computer Science*, *2*(4), 418–423.

Tharwat, A. (2018). Classification Assessment Methods. *Applied Computing and Informatics*. Advance online publication. doi:10.1016/j.aci.2018.08.003

Tiwari, R., & Singh, M. P. (2010). Correlation-Based Attribute Selection Using Genetic Algorithm. *International Journal of Computers and Applications*, *4*(8), 28–34. doi:10.5120/847-1182

Vapnik, V. N. (1999). An Overview of Statistical Learning Theory. *IEEE Transactions on Neural Networks*, *10*(5), 988–999. doi:10.1109/72.788640 PMID:18252602

Verónica Bolón, A.-B., Amparo, M., & Sánchez, C. N. (2017). *Artificial Intelligence: Foundations, Theory, and Algorithms Feature Selection for High-Dimensional Data*. Springer.

Wang, Y., Makedon, F. S., Ford, J. C., & Pearlman, J. (2004). Hykgene: A Hybrid Approach for Selecting Marker Genes for Phenotype Classification Using Microarray Gene Expression Data. *Bioinformatics (Oxford, England)*, *21*(8), 1530–1537. doi:10.1093/bioinformatics/bti192 PMID:15585531

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., & Steinberg, D. (2008). Top 10 Algorithms in Data Mining. *Knowledge and Information Systems*, *14*(1), 1–37. doi:10.100710115-007-0114-2

Yeh, Wu, & Chang. (2007). *Applying Data Mining Techniques for Cancer Classification from Gene Expression Data*. Paper presented at the International Conference on Convergence Information Technology (ICCIT 2007), Gyeongju, South Korea. 10.1109/ICCIT.2007.153

Yu, J., Yu, J., Almal, A. A., Dhanasekaran, S. M., Ghosh, D., Worzel, W. P., & Chinnaiyan, A. M. (2007). Feature Selection and Molecular Classification of Cancer Using Genetic Programming. *Neoplasia (New York, N.Y.)*, *9*(4), 292–303. doi:10.1593/neo.07121 PMID:17460773

Yu, L., & Liu, H. (2003). Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 856-863.

Zhang, H., Ren, Y.-g., & Yang, X. (2013). *Research on Text Feature Selection Algorithm Based on Information Gain and Feature Relation Tree*. Paper presented at the 10th Web Information System and Application Conference (WISA), Yangzhou, Jiangsu, China. 10.1109/WISA.2013.90

Chapter 4
Design and Procedures for the Investigation Conducted

ABSTRACT

In this chapter, the design of each proposed case study model mentioned in Chapter 3 is presented with their different experimental procedures. The chapter includes the data preparation, suitable parameters and data pre-processing, and detailed design of two case studies. Case 1: examining the accuracy and efficiency (time complexity) of high-performance gene selection and cancer classification algorithms; Case 2: A two-stage hybrid multi-filter feature selection method for high colon-cancer classification. It shows the experimental setup and environment and the description of the hardware and software components used.

1. INTRODUCTION

In this chapter, the design of each proposed case study model mentioned in Chapter 3 is presented with their different experimental procedures. The chapter includes the data preparation, suitable parameters and data pre-processing. It shows the experimental setup and environment and the description of the hardware and software components used.

2. CASE STUDY ONE: 'EXAMINING THE ACCURACY AND EFFICIENCY (TIME COMPLEXITY) OF HIGH-PERFORMANCE GENE SELECTION AND CANCER CLASSIFICATION ALGORITHMS'

2.1 Data Preparation and Pre-Processing

One way of addressing the key challenges posed by the massive number of genes associated with the small number of samples is to apply gene reduction (Scrucca, 2007), where the raw data are converted into a format more appropriate for analysis. As discussed in Chapter 2, the colon dataset is a group of

DOI: 10.4018/978-1-7998-7316-7.ch004

diverse expression records containing 62 samples extracted from various patients. 2000 gene expressions are selected, depending on their levels of expression confidence. 40 biopsies are counted (tumour) and 22 biopsies are counted (normal). The two pre-processing approaches discussed in Chapter 2 are employed and compared for this experiment.

1. The min–max normalisation is evaluated using Equation 2-2 to normalise the gene expression data to a scale between 0 and 1. The minimum value is scaled to 0, and the maximum value is scaled to 1.
2. The entropy-based discretisation method discretises the data categorically to reduce the noise of the original data using Equation 2-3 and Equation 2-4.

Both normalisation techniques prevent attributes and samples with large numerical ranges from dominating those with small range values. Thus, they enhance the similarity of gene expression data across the dataset.

2.2. Parameters Setting

The setting of parameters for GA and PSO algorithms implemented in this investigation are shown in Table 1. The parameter values are altered one-by-one for each algorithm using several test evaluations until the target values of best configurations are reached with high-performance results and better solution quality. Thus, the parameter values are chosen following multiple evaluation tests of the GA and PSO algorithms until the best configuration of performance results and computational outputs are met. Default original parameters are employed for the remaining algorithms, because they verify very good experimental results. Appendix 1 involves some investigation trials over several tests and evaluations that are randomly conducted to improve the results and assist in the selection of potential parameters. These trials reflect the reason behind the choice of parameters values throughout this research. Appendix 2 presents the default parameters for the ML algorithms conducted for classification throughout this research.

Table 1. PSO and GA parameter values for gene-subset selection and classification

PSO		**GA**	
Parameter	Value	Parameter	Value
Population Size (set Size)	10	Population Size	20
Iterations (No. of Generations)	20	Number of Generations	10
C1	1.0	Probability of Crossover Rate	0.6
C2	2.0	Probability of Mutation Rate	0.033

3. CASE STUDY TWO: 'A TWO-STAGE HYBRID MULTIFILTER FEATURE SELECTION METHOD FOR HIGH COLON-CANCER CLASSIFICATION'

3.1. Data Preparation and Pre-Processing

The colon-cancer dataset used in this case study is similar to the one discussed in Chapter 2. The data utilised in this system are pre-processed using the entropy-based discretisation method. This method is common as a global discretisation procedure and has shown interesting results. The default parameters are applied to the GA, IG, mRMR and SVM algorithms when conducting the experiments of this case study. Thus, the preliminary population size for GA is 20, and the generation number (i.e., termination criterion) is also 20. The values of 0.6 and 0.033 are set to crossover and mutation probabilities, respectively. The IG threshold value is set to 0.

3.2. Experimental Instrumentation and Environment

All case study experiments are implemented employing the Weka toolkit (Waikato Environment for Knowledge Analysis, Release 3.8). It is am ML toolkit developed in New Zealand at the University of Waikato (Witten et al., 2016). It is considered the most well-known open-source tool to perform ML and decision-making tasks (Alcalá-Fdez et al., 2009). It is a prevailing open-source Java-based ML and data pre-processing software package that includes a rich library of functions (Witten et al., 2016). It is freely available online at (https://www.cs.waikato.ac.nz/ml/weka/). The reason for using the Weka software package in this investigation is that it integrates data visualisation, data preparation, pre-processing, feature selection and ML algorithms in one attractive, user friendly and publicly available tool related to the scope of the study.

The computing environment for this research is a personal computer (PC) running an operating system of Windows 10 on a 1.8-GHz Intel Core i5 microprocessor with 8-GB RAM. The same experiments were tested on other machines with different specifications (Windows 7 and 8, Intel Core i7, and 16-GB RAM), achieving identical accuracy results.

The input file was an attribute-relation file format (ARFF) ASCII text file defining instance lists that share a set of attributes (Shraddha, Anuradha, & Swapnil, 2014). This file is developed to be easily input into Weka ML software.

3.3. Experiment Design

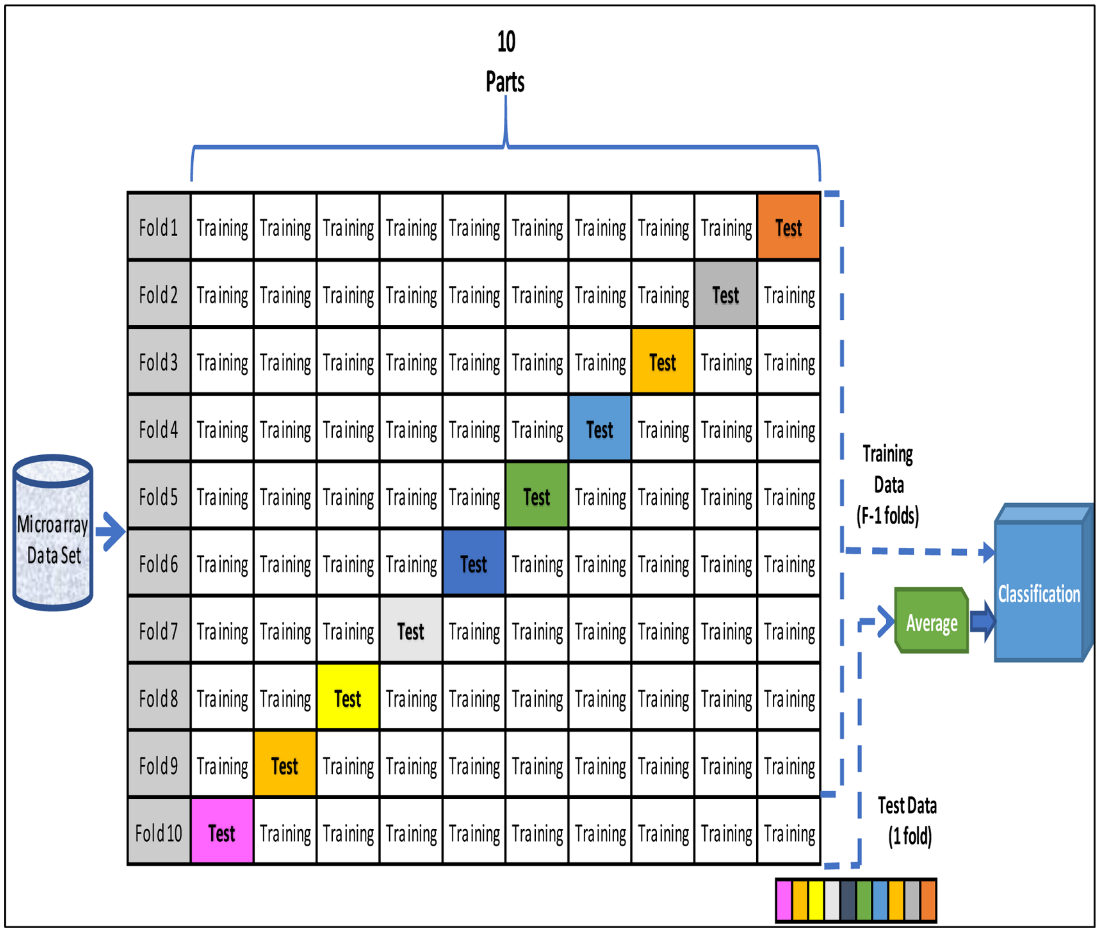
Validating ML accuracy is a primary step of data mining. To validate selected gene quality and to guarantee valid performance results for detecting fresh data, it is important to apply an independent test set (i.e., an unseen dataset sample) for the feature selection. To obtain valid and valuable results for higher classification accuracy rates, the selected gene dataset must have an independent test set. Thus, the data must be divided into two sets for training and testing. The training set validates the gene selection procedure, whereas the test set evaluates the classification methods. In our case, the results provide an unbiased evaluation for the final model.

In addition to the performance evaluation techniques discussed in Chapter 3, there exist many popular validation approaches to validate the data inside the Weka ML analysis tool, including leave-one-out cross-validation (LOOCV) and *k*-fold cross-validation. In both techniques, the dataset contents are randomly partitioned into training and testing sets for the validation process. These approaches are applied when an actual dataset is absent or unavailable. The dataset contents are distributed randomly into two sets: one for training and the other for testing. Then, diverse predictors are measured. LOOCV is a technique implemented on n–1 testers and later verified on those remaining (Al-Rajab, Lu, & Xu, 2017; Jäger, Sengupta, & Ruzzo, 2002; Kuncheva, 2004). The technique is repeated n times, leaving out one sample at the end. In the k-fold cross validation approach, data are randomly assigned to k non-overlapping groups (i.e., default folds division) of nearly equivalent size (Al-Rajab, Lu, & Xu, 2017; Dang, 2014; Kuncheva, 2004; Zeng, Li, & Chen, 2010). Thus, in most cases, the data are split into 10 parts using the K-fold cross validation approach. ($Data= D_1,D_2,D_3,\ldots,D_{10}$), and both training and testing data sets are repeated 10 times. When any part, D_n, where n=1,2,3,…,10 is employed as a test set, the remaining 9 parts are utilised as training sets. This process is repeated 10 times, and, at the end, the average of these 10 performance accuracy results are used to detect the classification accuracy for the dataset (Dash & Patra, 2012).

Because the number of samples generated from the medical research of the colon data set is considered small (<1000), cross validation is well-adopted as a measure of performance (Tan et al., 2006). Thus, all experiments conducted in this research apply the k-fold cross validation approach. Thus, the data are divided into 10 parts; both training and testing are implemented 10 times. The value of 10 is a good choice, because it is very popular in the ML field and has been proven to result in lower bias and modest variance (James et al., 2013; Kuhn & Johnson, 2013). As stated by (Kohavi, 1995), if choosing high values for K (e.g., leave-one-out), it typically results in lower bias. However, it provides estimators of high variance. Furthermore, if using low K values, it will result in higher bias and lower variance estimators. For the 10-fold cross validation, when k=1 is processed as a test set, the remaining nine parts are processed as a training set. This process is repeated 10 times, and the average of these 10 repetitions (k-folds) are considered for the classification accuracy for the dataset. Figure -1 presents the description of the 10-fold cross validation in which data is portioned into 10 sets. In each, one is for testing. Generally, the advantage of choosing cross validation over LOOCV is that it enables all the test data set samples to be independent, and it enables the reliability of the outcome results to be improved (Hu, 2010; Huang & Wang, 2006). Moreover, all the samples in the dataset are utilised for training and testing when applying the k-fold cross validation. However, applying the LOOCV results in high variance results, because this approach makes the training dataset sample identical to the entire dataset.

Moreover, the advantage of applying the feature selection before the classification algorithms is to get rid of the unrelated or redundant (i.e., uninformative) genes. This enhances the detection accuracy and reduces the computational complexity and time requirements. Another objective applying a feature selection process former to classification, expanding overfitting because the lack of the generalisation caused by overfitting results in high classification accuracy on the training data. Overfitting causes good performance and accuracy results to be obtained using a training sample. However, when fresh data samples are applied, satisfactory results cannot be obtained using the training model. Thus, it leads to very poor accuracy performance on the test data samples, affecting the outcome.

Figure 1. Design of the 10-fold cross validation



Fold1	Training	Training	Training	Training	Training	Training	Training	Training	Training	Test
Fold2	Training	Training	Training	Training	Training	Training	Training	Training	Test	Training
Fold3	Training	Training	Training	Training	Training	Training	Training	Test	Training	Training
Fold4	Training	Training	Training	Training	Training	Training	Test	Training	Training	Training
Fold5	Training	Training	Training	Training	Training	Test	Training	Training	Training	Training
Fold6	Training	Training	Training	Training	Test	Training	Training	Training	Training	Training
Fold7	Training	Training	Training	Test	Training	Training	Training	Training	Training	Training
Fold8	Training	Training	Test	Training	Training	Training	Training	Training	Training	Training
Fold9	Training	Test	Training	Training	Training	Training	Training	Training	Training	Training
Fold10	Test	Training	Training	Training	Training	Training	Training	Training	Training	Training

4. EXPERIMENT PROCEDURE

4.1 Case Study One: 'Examining the Accuracy and Efficiency (Time Complexity) of High-Performance Gene Selection and Cancer Classification Algorithms'

As discussed in Chapter 3, the first proposed method is conducted via several phases. The first phase implements the feature selection algorithms (i.e., PSO, GA and IG) across the entire dataset. The second phase utilises the original dataset to apply only the cancer classification algorithms without former feature selection algorithms. The third phase implements a combination of approaches, including both the selection and classification algorithms.

4.1.1. Phase One

In this phase, we examine the variance among multiple selection methods (i.e., PSO, GA and IG) compared to the number of genes selected across the normalised and discretised standard colon dataset, utilising the list of parameters given in Table 1.

Feature selection highlights genes that are relevant across the colon-cancer dataset. The best attributes of the dataset are selected during this step. Using the Weka data-mining tool, feature selection includes searching through all possible features to trigger the subset that works best for prediction. In Weka, there are two objects set: the attribute evaluator and the search method. The attribute evaluator governs which technique is employed to assign a value to each subset of attributes, whereas the search method specifies which search techniques are to be implemented. The most common evaluators used in these experiments are presented in Table 2. Attribute selection evaluation techniques. However, IG is considered a ranking method for feature and attribute selection. It is compulsory to utilise a threshold value for this algorithm. Thus, 0 is designated a threshold value. If the feature's weight is larger than 0, it is selected. Otherwise, it is neglected. Therefore, genes with discriminatory values equivalent to zero are also neglected.

Table 2. Attribute selection evaluation techniques

Attribute subset evaluator technique	Name	Function
	CfsSubsetEval	A simple filter algorithm which evaluates subsets of gene/feature by the means of predicting each individual gene/feature in relation to the redundancy degree between them
	WrapperSubsetEval	Uses a learning algorithm to measure attributes
	FilteredSubsetEval	A methodology which runs a random subset evaluator across the randomly filtered data
	InfoGainAttributeEval	Estimates the attribute value by computing the information gain in correspondence to the class.

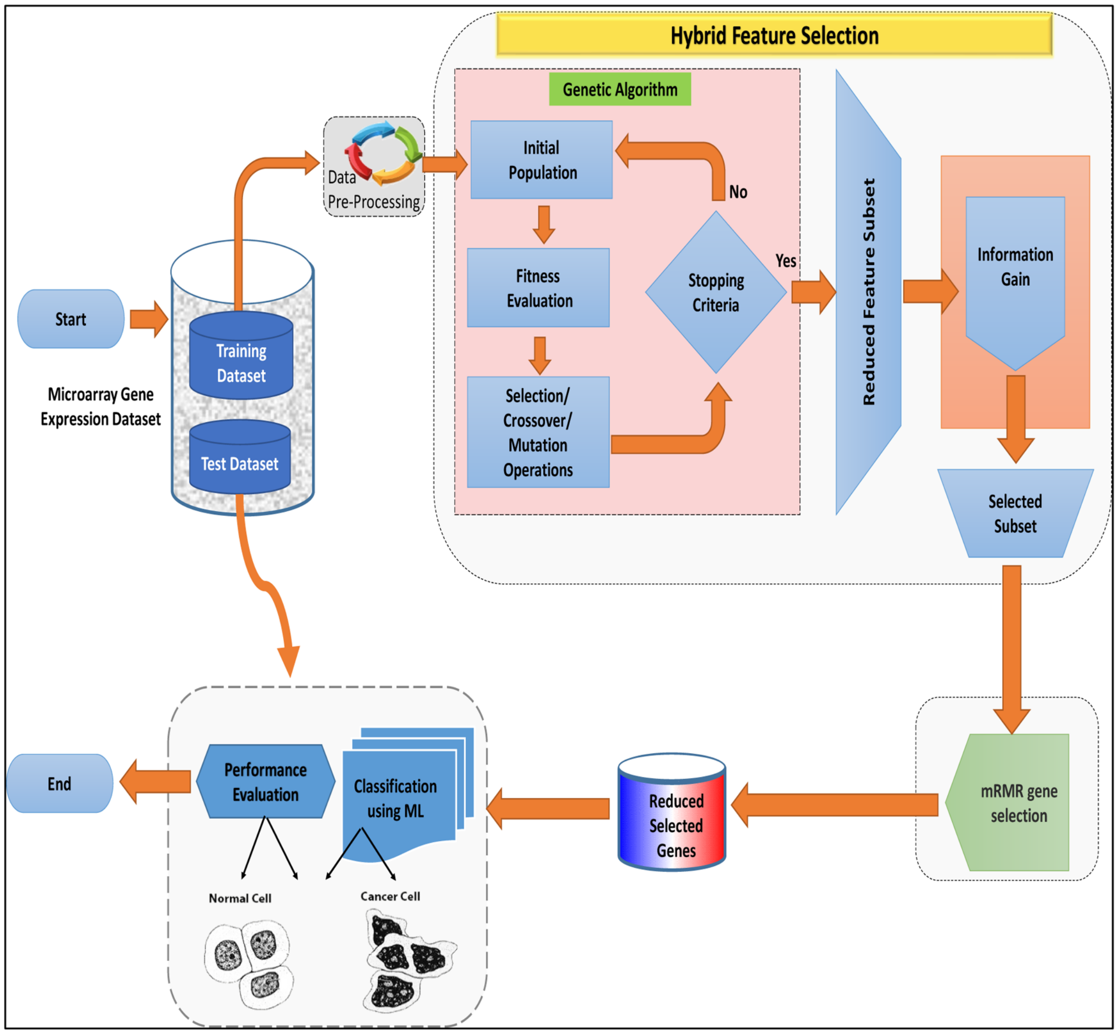
4.1.2. Phase Two

This phase implements classification algorithms (i.e., DT, SVM, NB and GP) without prior incorporation of gene selection algorithms. Thus, no filtering is applied. Only ML algorithms are executed without the involvement of gene selection algorithms. For the dataset, the experiments are steered by the first 10, 50, 100, 500, 1000, 1500 and 2000 attributes. The last value is the full number of features available in the dataset. This phase's experiment is applied using cross validation of k=10 and by employing default parameters (Appendix 2) to record coherence and valid results.

4.1.3. Phase Three

This phase implements classification algorithms after the implementation of gene selection algorithms (those implemented earlier in Phase One), resulting in informative selected data subsets, because these selected subsets should contain the most related data to the classification procedure. Default parameters are applied for the classification algorithms (Appendix 2), and the same experimental tools and conditions are applied as for phases one and two to assure accurate and coherent results.

Figure 2. 2-stage hybrid multifilter feature selection method



4.2. Case Study Two: 'A Two-Stage Hybrid Multifilter Feature Selection Method for High Colon-Cancer Classification'

In this study, data is pre-processed by applying the entropy-based discretisation method, as discussed in. Then, gene selection is implemented in two stages as follows.

- First, hybrid selection is employed. It comprises a GA as the selection method and a ranking algorithm, IG, as the first stage to select from all genes in the dataset. IG is used to compute the discriminative ranks for all genes selected by the GA. It computes the rank of each feature related to the class and sorts the features per their information values. Genes with discriminative scores equal to 0 are eliminated. This step optimises the outcome and reduces the gene set having more informative and related genes. The important part of this hybridisation process is the effectiveness

of the GA, because it is a well-known algorithm that discovers optimal solutions and is easy to implement. IG, however, ranks genes according to their importance and prepares them for stage two.
- Second, the mRMR filter method is applied across the reduced data from phase one, removing redundant genes, reducing noise and leaving only maximum relevant and related genes in the new subset. It enhances gene selection accuracy and speeds up the process. In this step, the mRMR is used to reduce the number of features and eliminate noise. Each gene is evaluated using the mRMR method. Then, the top *M* genes are formulated to select a newly reduced subset.
- Finally, a set of ML algorithms (e.g., SVM, NB, K-NN, GP and DT) are evaluated for both phases to assess the classification rate and to decide which is more efficient with greater accuracy.

An illustration of the pseudocode in Chapter 3 of this experiment is presented in Figure 2. GA is applied first to search the pool of features (i.e., population). The fitness function assists in the evaluation of each gene inside the population. Then, operations are applied to produce a new enhanced population (i.e., selection, crossover and mutation). This procedure is repeated until a stopping criterion is met. Later, the new feature subset is filtered using the IG, which evaluates the individuals of the subset and sorts them according to their information gain value. Next, the reduced subset is filtered by the mRMR method, keeping only the most related and relevant genes. Then, a diversity of ML algorithms are tested to evaluate performance and accuracy.

REFERENCES

Al-Rajab, M., Lu, J., & Xu, Q. i. (2017). Examining Applying High Performance Genetic Data Feature Selection and Classification Algorithms for Colon Cancer Diagnosis. *Computer Methods and Programs in Biomedicine*, *146*, 11–24. doi:10.1016/j.cmpb.2017.05.001 PMID:28688481

Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M. J., Ventura, S., Garrell, J. M., Otero, J., Romero, C., Bacardit, J., Rivas, V. M., Fernández, J. C., & Herrera, F. (2009). Keel: A Software Tool to Assess Evolutionary Algorithms for Data Mining Problems. *Soft Computing*, *13*(3), 307–318. doi:10.100700500-008-0323-y

Dang, V. Q. (2014). *Evolutionary Approaches for Feature Selection in Biological Data* (PhD Thesis). Edith Cowan University, Perth, Australia.

Dash, S., & Patra, B. (2012). Study of Classification Accuracy of Microarray Data for Cancer Classification Using Hybrid, Wrapper and Filter Feature Selection Method. *Proceedings of the International Conference on Bioinformatics & Computational Biology (BIOCOMP)*, 268-275.

Hu, Y. (2010). *Personalised Modelling Framework and Systems for Gene Data Analysis and Biomedical Applications* (PhD Thesis). Auckland University of Technology.

Huang, C.-L., & Wang, C.-J. (2006). A Ga-Based Feature Selection and Parameters Optimization for Support Vector Machines. *Expert Systems with Applications*, *31*(2), 231–240. doi:10.1016/j.eswa.2005.09.024

Jäger, J., Sengupta, R., & Ruzzo, W. L. (2002). *Improved Gene Selection for Classification of Microarrays.* Paper presented at the Pacific Symposium on Biocomputing, Lihue, HI. 10.1142/9789812776303_0006

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning* (Vol. 112). Springer. doi:10.1007/978-1-4614-7138-7

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1137-1145.

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling* (Vol. 26). Springer. doi:10.1007/978-1-4614-6849-3

Kulkarni, A., Kumar, B. N., Ravi, V., & Murthy, U. S. (2011). Colon Cancer Prediction with Genetics Profiles Using Evolutionary Techniques. *Expert Systems with Applications*, *38*(3), 2752–2757. doi:10.1016/j.eswa.2010.08.065

Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons. doi:10.1002/0471660264

Scrucca, L. (2007). Class Prediction and Gene Selection for DNA Microarrays Using Regularized Sliced Inverse Regression. *Computational Statistics & Data Analysis*, *52*(1), 438–451. doi:10.1016/j.csda.2007.02.005

Shraddha, S., Anuradha, N., & Swapnil, S. (2014). Study of Feature Selection Techniques Using Microarray Data. *International Journal of Emerging Technology and Advanced Engineering*, *4*(1), 417–422.

Tan, F., Fu, X., Zhang, Y., & Bourgeois, A. G. (2006). *Improving Feature Subset Selection Using a Genetic Algorithm for Microarray Gene Expression Data*. Paper presented at the IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada. 10.1109/CEC.2006.1688623

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Zeng, X.-Q., Li, G.-Z., & Chen, S.-F. (2010). *Gene Selection by Using an Improved Fast Correlation-Based Filter*. Paper presented at the IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), Hong Kong. 10.1109/BIBMW.2010.5703874

APPENDIX 1

Classifiers Parameters Adjustments Trials

Extra results pertaining the data presented in Table1 (Section 4.1.2) are provided to validate the choice of parameter selection over the arbitrary values for the *CFSSubsetEval*, *WrapperSubsetEval*, and *FilteredSubsetEval* methods.

Table 3. GA parameter evaluation on normalised data using the CFSSubsetEval method

Parameter	**Genetic Algorithm**																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	77.4	79.0	75.8	75.8	82.3	80.7	79.0	82.3	82.3	85.5	85.5	83.9	83.9	82.3	80.6	74.2	77.4	79.0	80.6	77.4
NB	58.1	58.1	59.7	61.3	62.9	59.7	61.3	58.0	61.3	62.9	58.1	58.1	59.7	59.7	59.7	58.1	59.7	61.3	59.7	59.7
DT	75.8	80.7	79.0	80.7	88.7	79.0	75.8	77.4	79.0	83.9	71.0	67.7	64.5	77.4	80.6	77.4	79.0	77.4	80.6	82.3

Table 4. GA parameter evaluation on discretised data using the CFSSubsetEval method

Parameter	**Genetic Algorithm**																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	77.4	74.2	74.2	75.8	77.4	80.7	80.6	82.3	74.2	79.0	77.4	75.8	77.4	77.4	79.0	71.0	80.7	80.7	80.7	82.3
NB	74.2	74.2	75.8	77.4	79.0	72.6	75.8	75.8	75.8	75.8	74.2	74.2	74.2	80.6	80.6	69.4	79.0	80.7	75.8	77.4
DT	83.9	88.7	85.5	85.5	87.1	91.9	90.3	85.5	83.9	83.9	79.0	79.0	72.6	77.4	80.7	72.6	82.3	84.0	82.3	85.5

Table 5. GA parameter evaluation on normalised data using the WrapperSubsetEval method with SVM as a classifier

Parameter	**Genetic Algorithm**																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	83.9	85.5	85.5	85.5	85.5	80.7	80.7	82.3	85.5	83.9	85.5	85.5	87.1	87.1	88.7	79.0	77.4	85.5	83.9	85.5
NB	54.8	53.2	54.8	58.1	58.1	61.3	62.9	61.3	61.3	62.9	62.9	58.1	59.7	61.3	61.3	61.3	66.1	61.3	61.3	62.9
DT	79.0	80.7	79.0	82.3	79.0	83.9	83.9	85.5	79.0	74.2	83.9	87.1	87.1	88.7	85.5	80.7	79.0	80.7	80.7	82.3

Table 6. GA parameter evaluation on discretised data using the WrapperSubsetEval method with SVM as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	80.6	82.3	82.3	74.2	77.4	80.7	79.0	79.0	80.7	80.7	77.4	82.3	80.7	80.7	80.7	72.6	75.8	77.4	71.0	75.8
NB	80.7	79.0	77.4	77.4	77.4	80.7	75.8	75.8	74.2	74.2	75.8	74.2	79.0	79.0	75.8	71.0	72.6	71.0	69.4	67.7
DT	82.3	82.3	82.3	82.3	82.3	82.3	82.3	82.3	82.3	82.3	79.0	79.0	82.3	82.3	79.0	82.3	83.9	71.0	83.9	83.9

Table 7. GA parameter evaluation on normalised data using the WrapperSubsetEval method with NB as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	82.3	82.3	82.3	80.7	82.3	85.5	85.5	85.5	85.5	82.3	82.3	83.9	83.9	83.9	87.1	77.4	82.3	80.6	83.9	80.6
NB	61.3	66.1	66.1	66.1	67.7	61.3	64.5	64.5	62.9	62.9	62.9	64.5	62.9	61.3	62.9	61.3	53.2	56.5	58.0	58.1
DT	79.0	82.3	82.3	82.3	82.3	79.0	83.9	80.7	85.5	82.3	85.5	79.0	79.0	77.4	79.0	79.0	82.3	80.6	80.6	77.4

Table 8. GA parameter evaluation on discretised data using the WrapperSubsetEval method with NB as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	77.4	77.4	79.0	79.0	79.0	75.8	74.2	75.8	75.8	77.4	80.7	82.3	82.3	79.0	74.2	74.2	79.0	79.0	80.6	80.6
NB	77.4	77.4	77.4	77.4	79.0	82.3	82.3	82.3	82.3	79.0	80.7	75.8	77.4	74.2	74.2	75.8	80.7	80.7	80.6	80.6
DT	79.0	80.7	80.7	80.7	83.9	80.7	80.7	80.7	80.7	80.7	79.0	79.0	79.0	79.0	82.3	80.7	82.3	82.3	79.0	82.3

Table 9. GA parameter evaluation on normalised data using the WrapperSubsetEval method with DT as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	82.3	82.3	83.9	83.9	83.9	80.7	82.3	83.9	83.9	85.5	82.3	83.9	83.9	82.3	80.6	79.0	77.4	79.0	79.0	79.0
NB	59.7	59.7	59.7	59.7	59.7	58.1	61.3	59.7	58.1	56.5	56.5	54.8	54.8	54.8	56.5	59.7	58.0	58.0	58.1	56.5
DT	75.8	80.7	79.0	79.0	79.0	83.9	83.9	83.9	83.9	82.3	83.9	83.9	82.3	82.3	82.3	83.9	82.3	82.3	82.3	82.3

Table 10. GA parameter evaluation on discretised data using the WrapperSubsetEval method with DT as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	82.3	82.3	83.9	82.3	85.5	79.0	79.0	79.0	83.9	85.5	80.7	77.4	80.7	79.0	85.5	82.3	85.5	87.1	83.9	83.9
NB	77.4	72.6	72.6	72.6	72.6	77.4	79.0	79.0	80.7	79.0	74.2	71.0	72.6	74.2	75.8	71.0	75.8	75.8	75.8	75.8
DT	87.1	87.1	87.1	87.1	87.1	85.5	88.7	88.7	88.7	88.7	85.5	88.7	88.7	88.7	88.7	87.1	87.1	87.1	85.5	85.5

Table 11. PSO parameter evaluation on normalised data using the CFSSubsetEval method

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	80.6	80.7	75.8	82.3	82.3	80.6	77.4	85.5	82.3	80.6	80.6	83.9	82.3	80.6	80.6	83.9
NB	58.1	59.7	61.3	62.9	58.1	61.3	59.6	61.3	59.7	61.3	61.3	69.4	59.7	61.3	64.5	67.7
DT	82.3	85.5	77.4	71.0	85.5	82.3	83.9	80.6	83.9	83.9	83.9	77.4	83.9	83.9	83.9	79.0

Table 12. PSO parameter evaluation on discretised data using the CFSSubsetEval method

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	74.2	72.6	80.6	69.4	77.4	75.8	75.8	72.3	79.0	77.4	79.0	74.2	79.0	75.8	77.4	77.4
NB	79.0	77.4	75.8	69.4	77.4	75.8	79.0	72.6	77.4	77.4	77.4	75.8	79.0	77.4	77.4	74.2
DT	77.4	85.5	85.5	72.6	77.4	82.3	83.9	75.8	80.6	82.3	88.7	79.0	80.6	82.3	87.1	77.4

Table 13. PSO parameter evaluation on normalised data using the WrapperSubsetEval method with SVM as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	88.7	88.7	85.5	87.1	88.7	88.7	82.3	87.1	88.7	87.1	83.9	83.9	88.7	88.7	82.3	85.5
NB	58.1	61.3	58.1	61.3	58.1	58.1	53.2	62.9	56.5	56.5	56.5	62.9	56.5	56.5	56.5	59.7
DT	88.7	83.9	83.9	83.9	88.7	87.1	80.6	79.0	88.7	87.1	79.0	79.0	88.7	87.1	79	79.0

Table 14. PSO parameter evaluation on discretised data using the WrapperSubsetEval method with SVM as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	74.2	75.8	72.3	74.2	72.6	74.2	77.4	74.2	72.6	75.8	75.8	71.0	72.6	75.8	75.8	71.0
NB	79.0	82.3	79.0	72.6	79.0	82.3	75.8	74.2	77.4	82.3	79.0	75.8	77.4	82.3	79.0	75.8
DT	80.6	79.0	82.3	80.6	79.0	80.6	80.6	74.2	79.0	80.6	79.0	77.4	79.0	80.6	79.0	77.4

Table 15. PSO parameter evaluation on normalised data using the WrapperSubsetEval method with NB as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	83.9	82.3	82.3	82.3	85.5	82.3	85.5	80.6	82.3	82.3	83.9	83.9	82.3	82.3	83.9	85.5
NB	56.5	62.9	62.9	64.5	56.5	62.9	67.7	67.7	56.5	64.5	71.0	69.4	56.5	64.5	72.6	69.4
DT	74.2	72.6	74.2	83.9	72.6	72.6	74.2	80.6	72.5	72.6	71.0	77.4	72.6	72.6	70.9	77.4

Table 16. PSO parameter evaluation on discretised data using the WrapperSubsetEval method with NB as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	83.9	79.0	83.9	87.1	83.9	77.4	82.3	85.5	83.9	77.4	83.9	83.9	83.9	77.4	83.9	85.5
NB	79.0	67.7	74.2	82.3	77.4	69.4	74.2	85.5	75.8	69.4	74.2	83.9	75.8	69.4	74.2	83.9
DT	82.3	75.8	77.4	88.7	82.3	72.6	77.4	87.1	82.3	72.6	77.4	87.1	82.3	72.6	77.4	87.1

Table 17. PSO parameter evaluation on normalised data using the WrapperSubsetEval method with DT as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	79.0	80.6	80.6	83.9	80.6	79.0	80.6	80.6	82.3	82.3	80.6	79.0	82.3	82.3	80.6	77.4
NB	54.8	56.5	59.7	61.3	56.5	56.5	58.1	61.3	56.5	56.5	59.7	59.7	56.5	56.5	61.3	61.3
DT	82.3	83.9	83.9	82.3	82.3	85.5	82.3	82.3	82.3	85.5	82.3	82.3	82.3	85.5	82.3	82.3

Table 18. PSO parameter evaluation on discretised data using the WrapperSubsetEval method with DT as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	82.3	80.6	82.3	88.7	82.3	82.3	82.3	85.5	82.3	82.3	83.9	85.5	82.3	85.5	83.9	85.5
NB	82.3	80.6	82.3	80.6	80.6	79.0	82.3	83.9	82.3	82.3	80.6	80.6	82.3	80.6	80.6	79.0
DT	87.1	88.7	85.5	87.1	87.1	88.7	85.5	87.1	87.1	88.7	85.5	87.1	87.1	88.7	85.5	87.1

Table 19. GA parameter evaluation on normalised data using the FilteredSubsetEval method with SVM as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	82.3	82.3	82.3	82.3	82.3	82.3	82.3	82.3	82.3	82.3	83.9	83.9	83.9	83.9	83.9	85.5	85.5	85.5	85.5	85.5
NB	66.1	66.1	66.1	66.1	66.1	66.1	66.1	66.1	66.1	66.1	58.1	58.1	58.1	58.1	58.1	54.8	54.8	54.8	54.8	54.8
DT	80.6	80.6	80.6	80.6	80.6	82.3	82.3	82.3	82.3	82.3	85.5	85.5	85.5	85.5	85.5	80.6	80.6	80.6	80.6	80.6

Table 20. GA parameter evaluation on discretised data using the FilteredSubsetEval method with SVM as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	71.0	71.0	71.0	71.0	71.0	74.2	74.2	74.2	74.2	74.2	77.4	77.4	77.4	77.4	77.4	77.4	77.4	77.4	77.4	77.4
NB	79.0	79.0	79.0	79.0	79.0	79.0	79.0	79.0	79.0	79.0	75.8	75.8	75.8	75.8	75.8	72.5	72.6	72.6	72.6	72.6
DT	77.4	77.4	77.4	77.4	77.4	75.8	75.8	75.8	75.8	75.8	75.8	75.8	75.8	75.8	75.8	71.0	71.0	71.0	71.0	71.0

Table 21. GA parameter evaluation on normalised data using the FilteredSubsetEval method with NB as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	80.6	80.6	82.3	80.6	82.3	80.6	79.0	79.0	79.0	80.6	85.5	83.9	83.9	80.6	83.9	80.6	82.3	82.3	82.3	82.3
NB	61.3	59.7	59.7	63.0	63.0	64.5	64.5	64.5	62.9	59.7	61.3	58.1	59.7	62.9	61.3	59.7	64.5	59.7	59.7	59.7
DT	77.4	77.4	77.4	77.4	80.6	83.9	82.3	82.3	83.9	83.9	83.9	82.3	80.6	79.0	77.4	79.0	80.6	79.0	79.0	80.6

Table 22. GA parameter evaluation on discretised data using the FilteredSubsetEval method with NB as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	83.9	83.9	85.5	82.3	83.9	72.6	72.6	72.6	72.6	72.6	79.0	77.4	77.4	79.0	79.0	80.6	79.0	79.0	79.0	77.4
NB	77.4	75.8	74.2	77.4	77.4	77.4	71.0	71.0	72.6	72.6	77.4	77.4	77.4	77.4	77.4	79.0	77.4	75.8	75.8	75.8
DT	80.6	82.3	82.3	82.3	82.3	79.0	82.3	80.6	82.3	80.6	82.3	83.9	83.9	85.5	85.5	79.0	77.4	79.0	79.0	77.4

Table 23. GA parameter evaluation on normalised data using the FilteredSubsetEval method with DT as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	83.9	83.9	83.9	83.9	83.9	80.6	80.6	80.6	80.6	80.6	82.3	82.3	82.3	82.3	82.3	74.2	74.2	74.2	74.2	74.2
NB	59.7	59.7	59.7	59.7	59.7	61.3	61.3	61.3	61.3	61.3	58.1	58.1	58.1	58.0	58.0	54.8	54.8	54.8	54.8	54.8
DT	79.0	79.0	79.0	79.0	79.0	64.5	64.5	64.5	64.5	64.5	75.8	75.8	75.8	75.8	75.8	66.1	66.1	66.1	66.1	66.1

Table 24. GA parameter evaluation on discretised data using the FilteredSubsetEval method with DT as a classifier

Parameter	Genetic Algorithm																			
Population Size	10	10	10	10	10	20	20	20	20	20	50	50	50	50	50	100	100	100	100	100
No. of Generations	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100	10	20	30	50	100
Classifier	Classification Accuracy [%]																			
SVM	75.8	71.0	71.0	71.0	71.0	75.8	77.4	75.8	75.8	75.8	79.0	75.8	75.8	75.8	77.4	71.0	71.0	72.6	71.0	71.0
NB	79.0	82.3	82.3	82.3	82.3	80.6	82.3	82.3	82.3	82.3	75.8	75.8	75.8	74.2	74.2	75.8	75.8	75.8	79.0	79.0
DT	79.0	80.6	80.6	80.6	80.6	80.6	82.3	83.9	83.9	80.6	74.2	74.2	74.2	74.2	74.2	72.6	72.6	72.6	72.6	72.6

Table 25. PSO parameter evaluation on normalised data using the FilteredSubsetEval method with SVM as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	83.9	80.6	85.5	82.3	83.9	80.6	83.9	75.8	85.5	80.6	88.7	75.8	85.5	80.6	87.1	77.4
NB	58.1	53.2	58.1	53.2	58.1	53.2	61.3	53.2	58.1	53.2	59.7	56.5	58.1	53.2	59.7	58.1
DT	83.9	85.5	72.6	66.1	83.9	85.5	75.8	71.0	83.9	85.5	69.4	77.4	83.9	85.5	69.4	77.4

Table 26. PSO parameter evaluation on discretised data using the FilteredSubsetEval method with SVM as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	79.0	80.6	69.4	74.2	79.0	79.0	71.0	75.8	80.6	82.3	74.2	74.2	80.6	82.3	74.2	74.2
NB	79.0	79.0	72.6	67.7	79.0	82.3	72.6	75.8	79.0	83.9	74.2	74.2	79.0	83.9	74.2	75.8
DT	83.9	83.9	71.0	77.4	83.9	85.5	75.8	80.6	83.9	83.9	75.8	79.0	83.9	83.9	75.8	79.0

Table 27. PSO parameter evaluation on normalised data using the FilteredSubsetEval method with NB as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	75.8	80.6	82.3	79.0	79.0	79.0	83.9	75.8	80.6	80.6	85.5	77.4	80.6	80.6	83.9	77.4
NB	58.1	64.5	56.5	59.7	58.1	64.5	61.3	62.9	59.7	64.5	61.3	61.3	59.7	64.5	59.7	61.3
DT	82.3	77.4	79.0	79.0	80.6	77.4	79.0	75.8	80.6	80.6	80.6	80.6	80.6	80.6	82.3	82.3

Table 28. PSO parameter evaluation on discretised data using the FilteredSubsetEval method with NB as a classifier

Parameter	Particle Swarm Optimisation															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	77.4	85.5	79.0	77.4	80.6	80.6	75.8	80.6	82.3	80.6	72.6	75.8	82.3	80.6	74.2	74.2
NB	77.4	75.8	74.2	74.2	74.2	75.8	77.4	71.0	74.2	75.8	77.4	72.6	74.2	75.8	77.4	71.0
DT	87.1	82.3	79.0	80.6	87.1	83.9	79.0	75.8	87.1	83.9	79.0	75.8	87.1	83.9	79.0	75.8

Table 29. PSO parameter evaluation on normalised data using the FilteredSubsetEval method with DT as a classifier

Parameter	**Particle Swarm Optimisation**															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	80.6	79.0	80.6	74.2	80.6	79.0	75.8	75.8	82.3	79.0	79.0	74.2	82.3	79.0	75.8	74.2
NB	59.7	58.1	59.7	53.2	59.7	58.1	56.5	51.6	61.3	58.1	56.5	50.0	61.3	58.1	54.8	50.0
DT	82.3	77.4	67.7	72.6	82.3	77.4	72.6	72.3	82.3	77.4	72.6	71.0	82.3	77.4	72.6	71.0

Table 30. PSO parameter evaluation on discretised data using the FilteredSubsetEval method with DT as a classifier

Parameter	**Particle Swarm Optimisation**															
Population Size	10	20	100	200	10	20	100	200	10	20	100	200	10	20	100	200
No. of Generations	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
C1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Classifier	Classification Accuracy [%]															
SVM	80.6	83.9	64.5	75.8	77.4	85.5	69.4	75.8	79.0	85.5	71.0	77.4	79.0	85.5	71.0	77.4
NB	74.2	80.6	69.4	77.4	72.6	79.0	69.5	75.8	72.6	77.4	71.0	77.4	72.6	77.4	71.0	77.4
DT	77.4	79.0	72.6	69.4	77.4	80.6	69.4	74.2	77.4	80.6	72.6	74.2	77.4	80.6	72.6	74.2

APPENDIX 2

ML Classifiers Parameters

Table 31. Decision Tree - C4.5 class default parameters

Confidence factor for pruning	**Minimum number of instances per field**	**Seed**	**Subtree operation considered when pruning**	**Pruned/ unpruned decision tree**
0.25 (default)	2 (default)	1 (default)	True	Using pruned tree

Table 32. Naïve Bayes class default parameters

Use Kernel Estimator	Use Supervised Discretization to convert numeric attributes to nominal ones
False	False

Table 33. SVM class default parameters

The complexity constant C	Epsilon for round-off error	Kernel Type	Normalize/ standardize/ neither	Tolerance	Kernel option: the Exponent to use	Random seed for the cross validation	Kernel option: the size of the cache
1 (default)	1.0E-12 (default)	Polynomial (default)	Normalize (default)	1.0E-3 (default)	1.0 (default)	1 (default)	250007 (max) (default)

Table 34. Genetic Programming class default parameters

Bias constant for exponential ranking selection	Size of the elite population	Fitness evaluation method	Maximum depth of the program tree	The size of the children population	Method for initializing a population of program trees	Population size
0.5 (default)	5 (default)	Standard classifier	5 (default)	100 (default)	initialized with Ramped Half and Half method	100 (default)

Chapter 5
Findings for the Conducted Investigations

ABSTRACT

This chapter focuses on the results produced from each case study experiment. For case one, the experiments were conducted in three phases. Phase one implemented GA, PSO, and IG as the gene/feature selection algorithms over the entire dataset. Phase =two2 utilised the original dataset to implement only the cancer classification algorithms without involving any gene/feature selection algorithms. Four recognised classification algorithms are employed: SVM, NB, GP, and DT. The third phase implemented the combined approach of gene selection and cancer classification algorithms. The results of these phases are presented in the next subsections. For case two, these experiments were implemented in two phases. Phase one implemented the classification algorithms over the features selected by the hybridised selection algorithms (GA+IG), whereas Phase two classified the features using the proposed two-stage multifilter selection system. In this section, the results are presented as follows

1. INTRODUCTION

In Chapter 3 and Chapter 4, a complete description of each proposed system was presented along with design and procedures. This chapter focuses on the results produced from each case study experiment.

2. RESULTS OF CASE STUDY ONE: 'EXAMINING THE ACCURACY AND EFFICIENCY (TIME COMPLEXITY) OF HIGH-PERFORMANCE GENE SELECTION AND CANCER CLASSIFICATION ALGORITHMS'

As discussed in Chapter 4.5.1, the experiments were conducted in three phases. Phase one implemented GA, PSO and IG as the gene/feature selection algorithms over the entire dataset. Phase two utilised the original dataset to implement only the cancer classification algorithms without involving any gene/feature selection algorithms. Four recognised classification algorithms are employed: SVM, NB, GP and

DOI: 10.4018/978-1-7998-7316-7.ch005

DT. The third phase implemented the combined approach of gene selection and cancer classification algorithms. The results of these phases are presented in the next subsections.

2.1. Phase One

During this phase, the experimentation examined the variance between the different gene selection methods of GA, PSO and IG, compared to the number of selected genes from the standard pre-processed dataset. Tables 1 through 6 present the number of genes selected after employing GA and PSO algorithms using diverse attribute evaluation approaches and two popular pre-processing data techniques discussed in Chapter 2 and Chapter 4.

Table 1 and Table 2 show the number of selected genes/features employing both GA and PSO algorithms using the CfsSubsetEval attribute evaluator method. From these tables, it is apparent that, when applying different data pre-processing techniques, the PSO algorithm reduces the number of selected genes to approximately half (1255 genes/features out of 2000, 62.8% remaining), compared to the result of GA, which showed better reduction (444 genes/features, 22% remaining).

Table 1. FS Method Results using – CfsSubsetEval (Data Normalised)

	Search Technique	
	Genetic Algorithm	**Particle Swarm Optimisation**
No. of genes selected	444	1255
Percentage of selected genes	22.2%	62.8%

Table 2. FS Method Results using – CfsSubsetEval (Data Discretised)

	Search Technique	
	Genetic Algorithm	**Particle Swarm Optimisation**
No. of genes selected	444	1255
Percentage of selected genes	22.2%	62.8%

Table 3 presents normalised data from the WrapperSubsetEval method. PSO resulted in a lower number of genes/features when WrapperSubsetEval method was applied with the SVM classifier (187 genes/features, 9.4% remaining), using normalised data compared to other classifiers and the same method. The GA algorithm reduced the data to 400 genes/features using the DT classifier. Generally, PSO showed better reduction when employing the WrapperSubsetEval method. Table 4 presents the results when the data was pre-processed using the discretisation technique. PSO selects the least number of related features (317 genes/features, 15.8% remaining) compared to GA (579 genes/features) using the NB classifier. Generally, PSO selected fewer features/genes compared to the GA algorithm using the wrapper method.

Table 3. FS Method Results using – WrapperSubsetEval (Data Normalised)

Search Technique	**Learning Scheme Embedded**		
	Support Vector Machine	**Naïve Bayes**	**Decision Tree**
Genetic Algorithm	780	657	400
Percentage of selected genes	39.0%	32.9%	20.0%
Particle Swarm Optimisation	187	596	257
Percentage of selected genes	09.4%	29.8%	12.9%

Table 4. FS Method Results using – WrapperSubsetEval (Data Discretised)

Search Technique	**Learning Scheme Embedded**		
	Support Vector Machine	**Naïve Bayes**	**Decision Tree**
Genetic Algorithm	778	579	917
Percentage of selected genes	38.9%	29.0%	45.9%
Particle Swarm Optimisation	713	315	730
Percentage of selected genes	35.7%	15.8%	36.5%

Table 5 shows that the GA algorithm using the SVM classifier produced fewer features when used with FilteredSubsetEval (205 genes/features, 10% remaining) with normalised data, compared to PSO, which had the least feature selection applied with SVM (211 genes/features, 11% remaining). Table 6 presents the results of using the same attribute evaluation method when the data were pre-processed using the discretised technique. GA showed a very reduced number of features using the NB classifier, leaving only 370 genes/features and 18.5% remaining, compared to PSO with the same classifier, which resulted in 572 genes/features, 28.6% remaining.

Table 5. FS Method Results using - FilteredSubsetEval (Data Normalised)

Search Technique	**Learning Scheme Embedded**		
	Support Vector Machine	**Naïve Bayes**	**Decision Tree**
Genetic Algorithm	205	295	485
Percentage of selected genes	10.3%	14.8%	24.3%
Particle Swarm Optimisation	211	634	449
Percentage of selected genes	10.6%	31.7%	22.5%

Note that IG is used as a feature selection ranking method. A threshold value is required for this algorithm. Thus, a threshold value of 0 was chosen, as discussed in Section 4.5. IG selected 135 genes/features using both normalised and discretised pre-processing techniques. Thus, genes with discriminative values equal to zero were discarded. Appendix 1 presents the top 135 ranked genes/features.

Table 6. FS Method Results using - FilteredSubsetEval (Data Discretised)

Search Technique	Learning Scheme Embedded		
	Support Vector Machine	**Naïve Bayes**	**Decision Tree**
Genetic Algorithm	438	370	959
Percentage of selected genes	21.9%	18.5%	48.0%
Particle Swarm Optimisation	619	572	674
Percentage of selected genes	31.0%	28.6%	33.7%

Figure 1 summarises the different feature selection methods applied to normalised data. In the case of CfsSubsetEval, GA selected less data (444 genes/features) compared to PSO, which selected 1255 genes/features. In the case of WrapperSubsetEval, GA selected more data (400–780 genes/features) compared to PSO, which selected 187–596 genes/features. When applying FilteredSubsetEval, GA also selected less data (205–485 genes/features) compared to PSO, which selected 211–634 genes/features. However, IG selected 135 genes/features. In summary, IG selected the lowest number of features, followed by PSO with the SVM classifier, which selected 187 genes/features.

Figure 1. Number of features using selection algorithms (normalised genes)

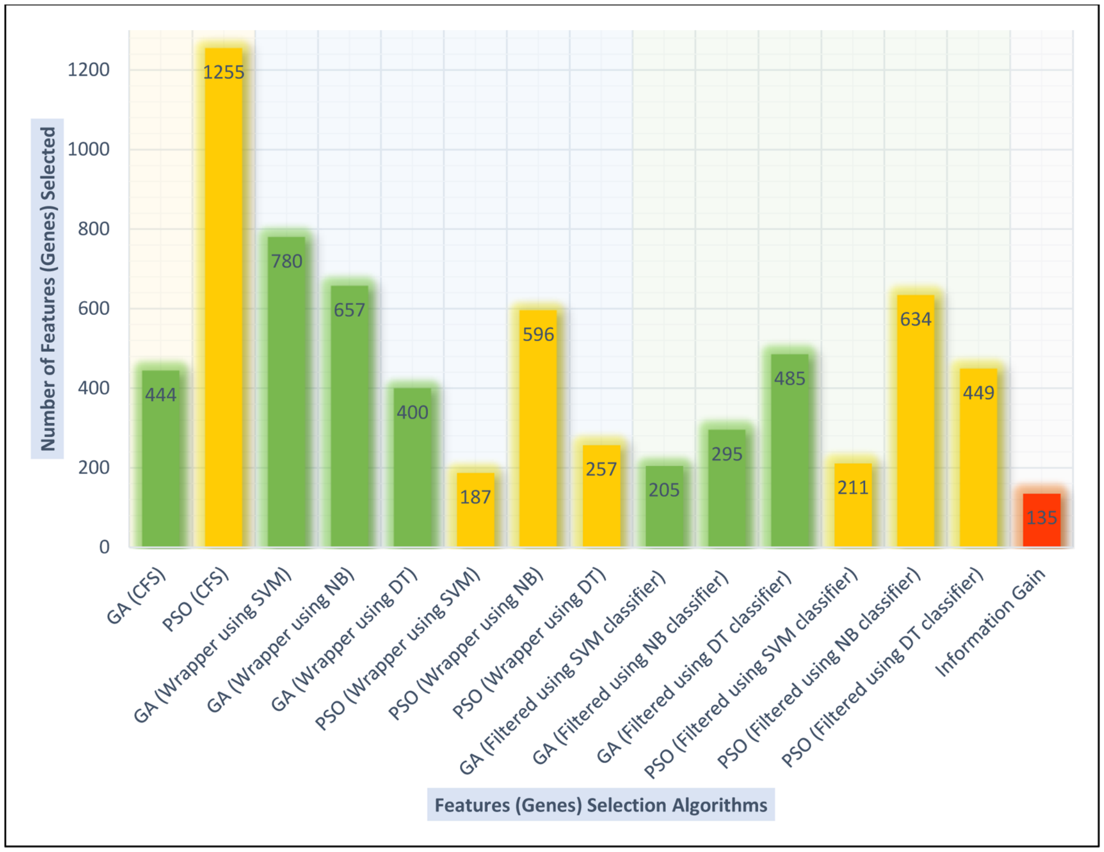
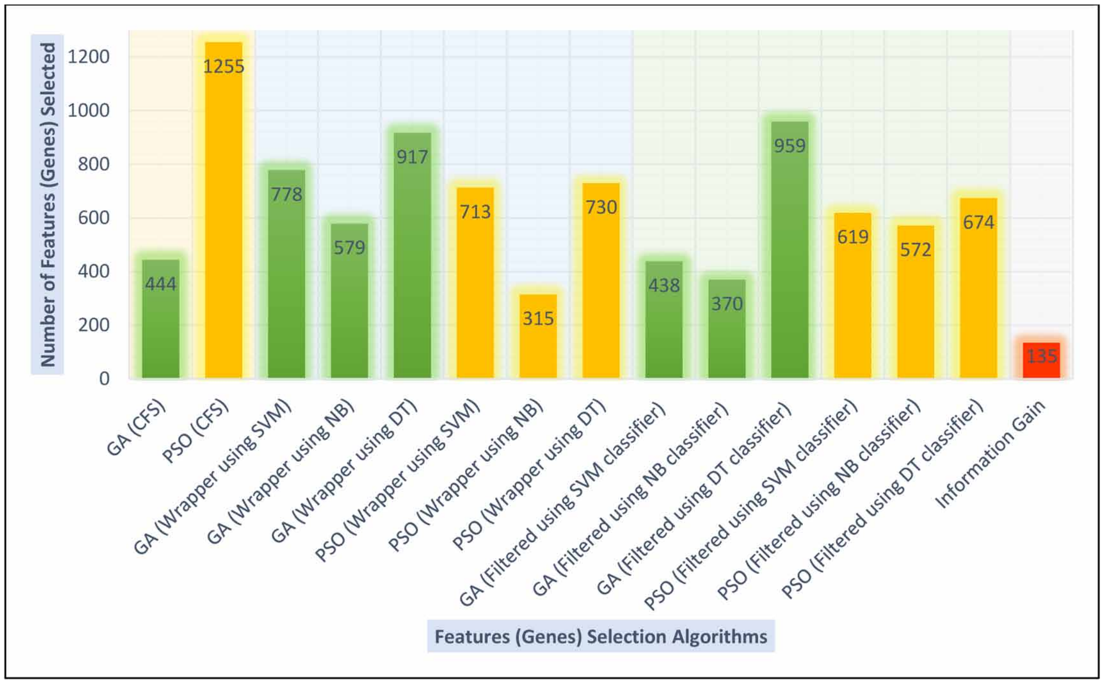


Figure 2 summarises the different feature selection methods applied on the discretised data. In the case of CfsSubsetEval, generally, GA selected less data (444 genes/features) compared to PSO, which selected 1255 genes/features. However, with WrapperSubsetEval, GA selected more data (579–917 genes/features) compared to PSO, which selected 315–730 genes/features. Moreover, in the case of applying FilteredSubsetEval, GA selected less data (370–438 genes/features), except when the DT classifier was used, resulting in the largest number of selected data (959 genes/features). When applying FilteredSubsetEval with PSO, 572–674 genes/features were chosen. However, IG selected 135 genes/features. In summary, IG had the lowest number of genes/features selected, followed by PSO and NB, which selected 315 genes/features.

When applying CfsSubsetEval and IG, the same results were obtained when utilising different preprocessing techniques: no change in the number of data selected.

Figure 2. Number of features using selection algorithms (discretised genes)



2.2. Phase Two

In this phase, SVM, NB, DT and GP classification algorithms were executed using Weka default parameters, without incorporating any gene selection algorithm (i.e., no filtering applied). Thus, it only implemented ML classifiers.

Figure 3 records the classification accuracy obtained compared with the number of normalised genes input. SVM resulted in better accuracy compared to other classification algorithms (86%), when applied to the full dataset. DT had a classification accuracy of 82%, GP had an accuracy of 76% and NB had the lowest accuracy (52%), when applied to the same sized dataset.

Figure 3. Different normalised gene sample classification accuracy

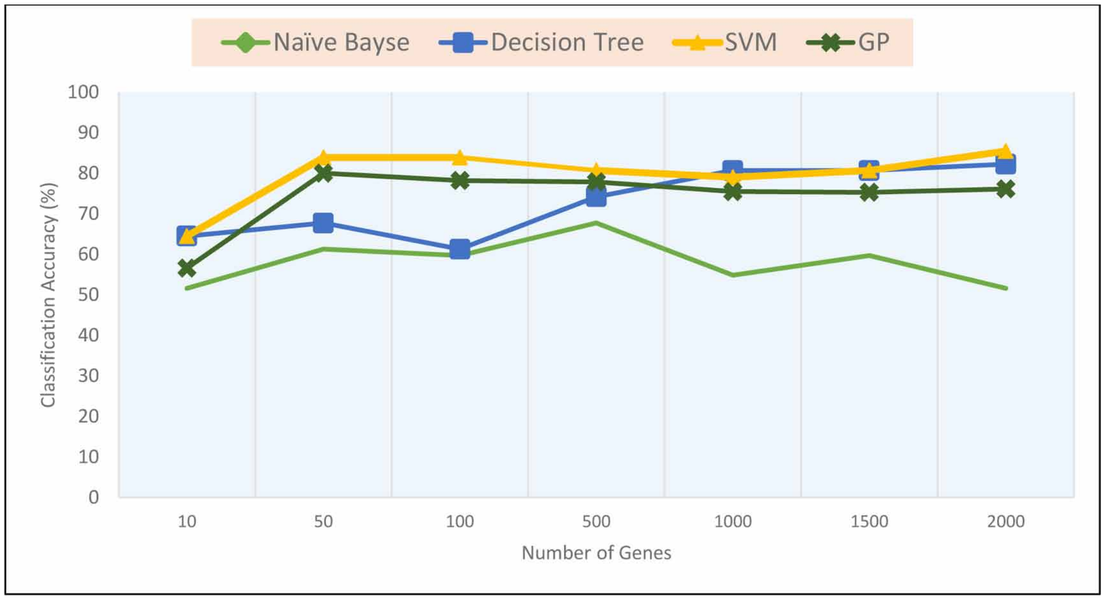


Figure 4 shows the time required to classify various normalised gene samples. For a smaller number of genes, almost all classification algorithms required less time, because processing was almost instantaneous. However, for many genes/features GP took more time (>7 s). Thus, when the amount of input increases, the function of GP reaches a big *O*(*n log n*).

Figure 4. Normalised gene sample classification time

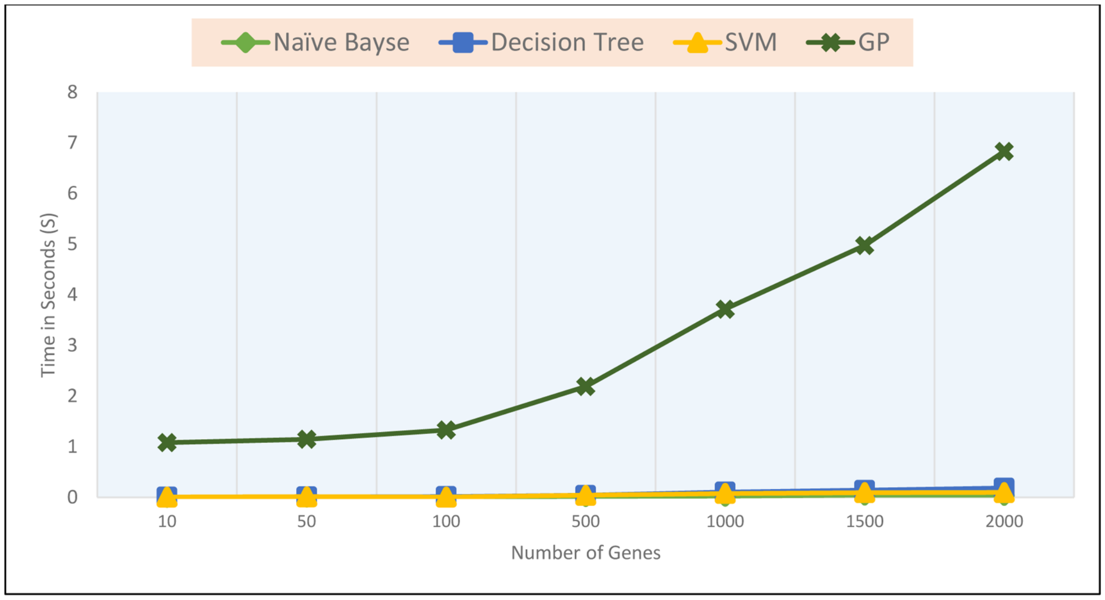


Figure 5 records the classification accuracy obtained compared to the number of discretised genes/features input. DT resulted in better accuracy over other classification algorithms (87%) when applied to the full dataset. However, GP had the lowest accuracy (51%). SVM and NB had classification accuracies of 82% and 79%, respectively, when applied to the same sized dataset.

Figure 5. Different discretised gene sample classification accuracy

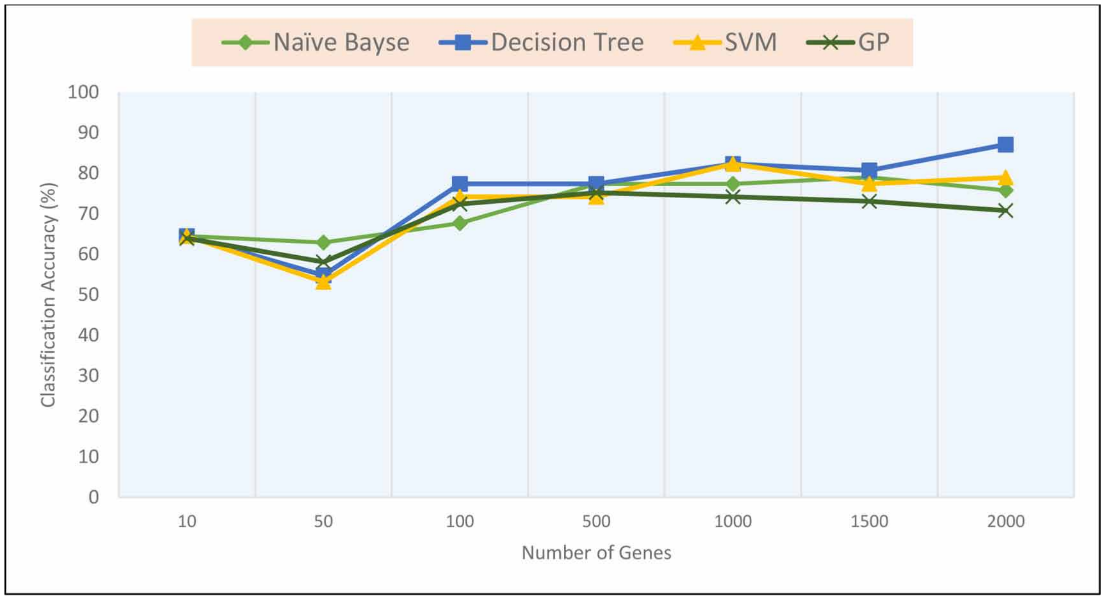


Figure 6 shows the time required to classify the various discretised genes samples. For a smaller number of genes, almost all classification algorithms required less time, because processing was almost instantaneous. However, for many genes, GP took more time (>6 s). Thus, when the amount of input increases, the function of GP reaches a big $O(n \log n)$.

2.3. Phase Three

This phase applied classification algorithms after the implementation of the gene selection algorithms (i.e., Phase one). The selected subsets were supposed to contain the most related and informative data for the classification process. The experiments were conducted using the selected genes/features from the colon dataset attributes, which had been reduced. The time computed was the one used to build the classification model. As shown in Figure 7, applying CfsSubsetEval using GA/DT achieved the highest classification accuracy of 92% when the data were pre-processed using the discretization technique. If the data were pre-processed using a normalised technique, the highest accuracy was 81% using the GA/SVM of the same method. When the data were pre-processed using a discretization technique, it achieved better classification accuracy compared to normalised pre-processing. The analysis of time complexity function for GA/DT is $T(n)=n^2+ mn^2$ with $O(n^2)$. The DT has a polynomial/quadratic time complexity. GA/NB had the least classification accuracy when the data were normalised (60%), whereas GA/NB had the least accuracy when the data were discretised (73%).

Figure 6. Discretised gene sample classification time

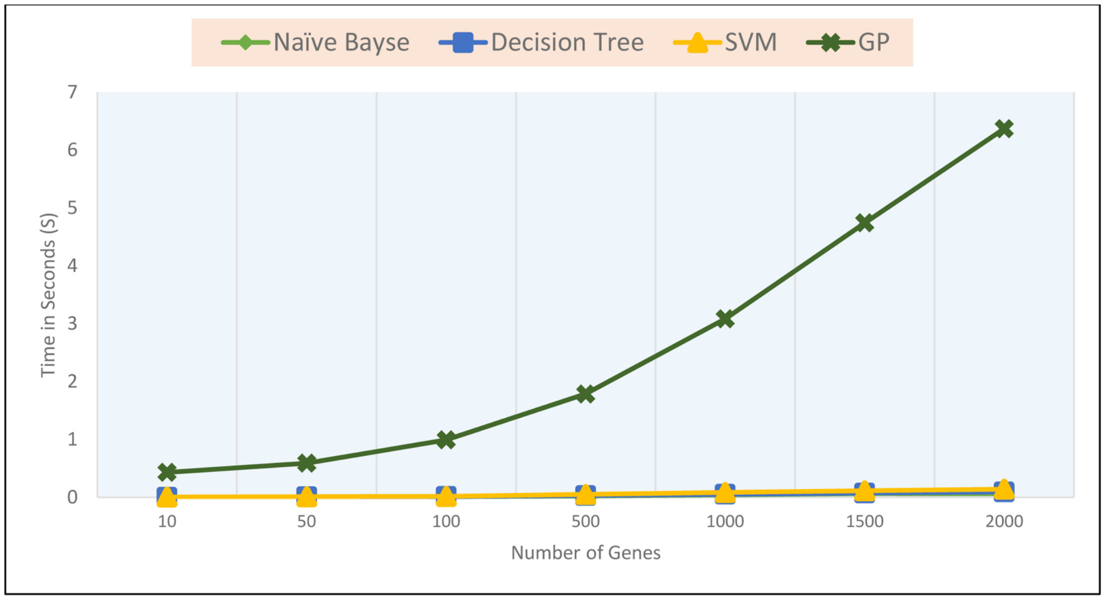
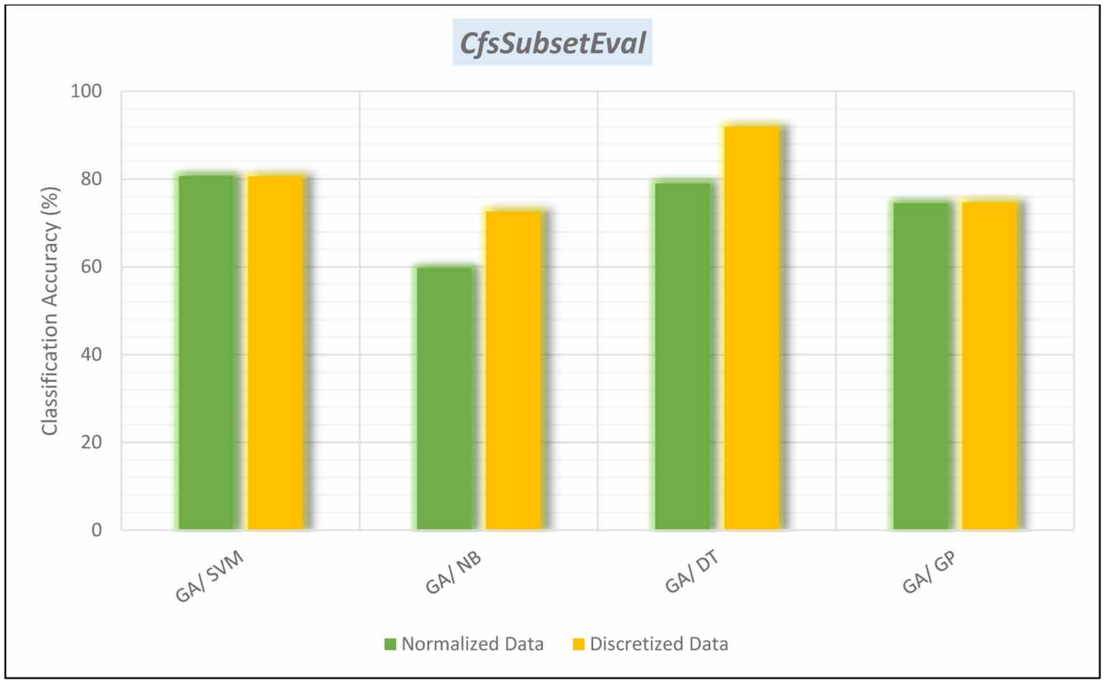


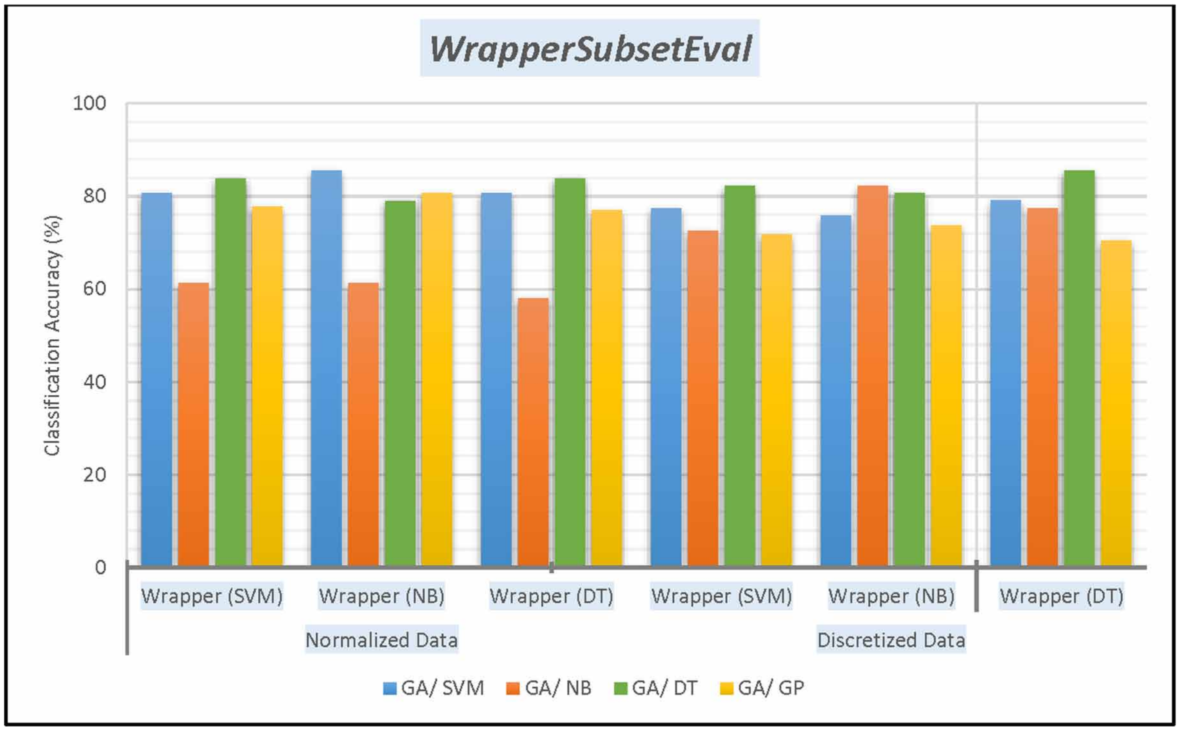
Figure 7. Classification accuracy average of multiple ML algorithms applied by the CfsSubsetEval method employing GA as a feature selection algorithm



From Figure 8, we see that, by applying the WrapperSubsetEval method, GA/DT achieved the highest classification accuracy when the data were discretised (86%) and wrapped by the DT classifier. However, GA/SVM achieved the same accuracy (86%) when the data were normalised and wrapped by the NB classifier. The analysis of time complexity function for GA/DT is $T(n)=n^2+mn^2$, with $O(n^2)$. The DT has a quadratic time complexity notation, whereas the time complexity function for GA/SVM is $T(n)=n^2+n^3$, i.e., $O(n^3)$. It is apparent that GA/NB had the least classification accuracy when the data were normalised (58%), whereas GA/GP had the least classification accuracy when the data were discretised (71%). Generally, GA/DT was more efficient than GA/SVM, although they result in same classification accuracy.

In Figure 9, we find that if the FilteredSubsetEval evaluation method was applied along with many

Figure 8. Classification accuracy average of multiple ML algorithms applied by the WrapperSubsetEval method employing GA as a feature selection algorithm



classifiers, GA/DT had achieved the highest classification accuracy of 84% by filtering the data with the NB classifier on the normalised data. However, if the data were discretised, the filtering evaluation method of GA/DT and GA/NB had the highest accuracy of 81% when it was filtered with the DT classifier. The time complexity analysis function of GA/DT is $T(n)=n^2+mn^2$, i.e., $O(n^2)$. The DT has a quadratic time complexity notation. GA/NB had the least classification accuracy (61%) when the data were normalised, whereas GA/GP had the least when the data were discretised (72%).

Figure 9. Classification accuracy average of multiple ML algorithms applied by the FilteresSubsetEval method employing GA as a feature selection algorithm

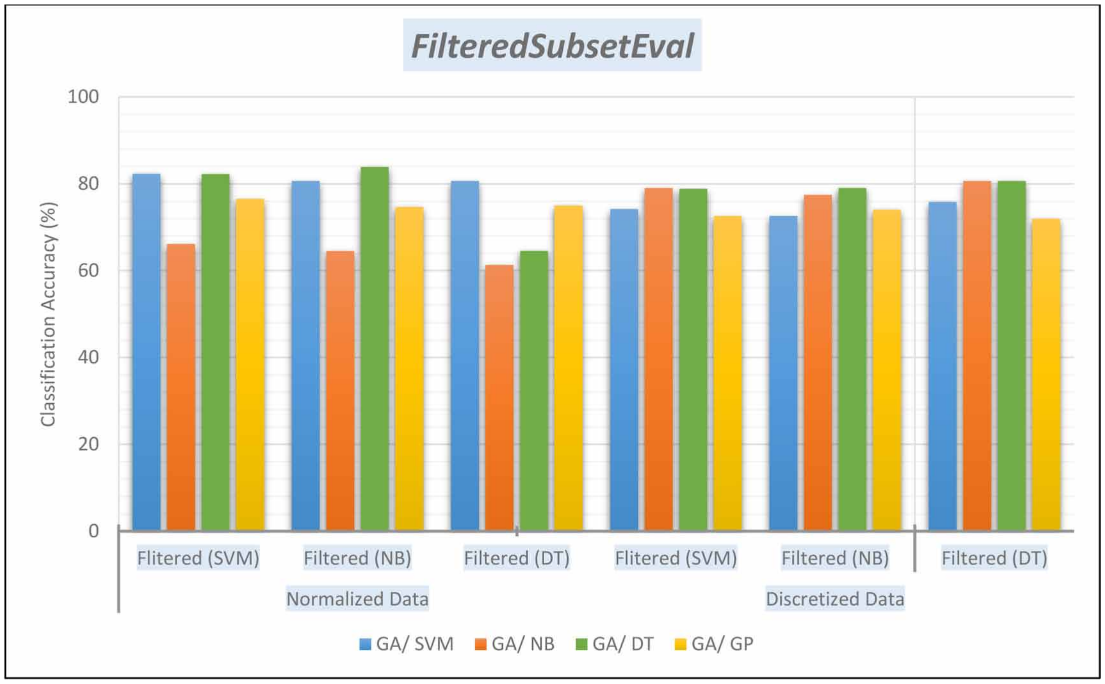


Figure 10 (a and b) shows the time required to evaluate the GA algorithm using the CfsSubsetEvel method for normalised and discretised data. GA/GP took more time ~5–6 swhen the number of genes/ features increased with a time complexity function of $T(n)=n^2+n \log n$, i.e., $O(n^2)$ resulting in lower classification accuracy compared to other algorithms (75%). However, GA/DT required less time (up to 3.5 s) with a time complexity function of $T(n)=n^2+mn^2$ and, i.e., $O(n^2)$, resulting in better accuracy (92%). GA/NB had the least classification accuracy (60%) with a time complexity function of $T(n)=n^2+mn$, i.e., $O(n^2)$. Whereas all algorithms had the same $O(n^2)$ complexity, GA/DT was more accurate.

Figure 11 (a, b and c) presents the time taken to evaluate the GA algorithm using the WrapperSubsetEval method on normalised data. Figure 5-11 (e, f and g) presents the same on the discretised data. The time taken by the discretised data was much lower than the normalised data. Thus, it was more efficient. When the data were discretised, the best accuracy was found by GA/DT (86%) with a time ~14 s and $O(n^2)$. Whereas, if the data were normalised, the best accuracy was found by GA/SVM (86%) with a time ~32 s and $O(n^3)$. However, the accuracy is the same for normalised and discretised data. The last has a time complexity $O(n^2)$, which is more efficient.

Figure 12 (a, b and c) presents the time taken to evaluate the GA algorithm using the FilteredSubsetEval method on normalised data. Figure 12 (e, f and g) presents the same for the discretised data. The time taken by the discretised data was much lower than the normalised data. Thus, it was more efficient. When the data were discretised, the best accuracy was found by both GA/DT and GA/NB (81%) with a time ~1 s. However, if the data was normalised, the best accuracy was found by GA/DT (84%) with a time ~4 s. GA/DT had a big $O(n^2)$, whereas GA/ NB had $O(n^2)$, because GA used polynomial quadratic notation. Generally, whereas the accuracy was better for the normalised data, it took more time than discretised data.

Figure 10. Average time of classification for different gene samples using the CfsSubsetEval method with GA

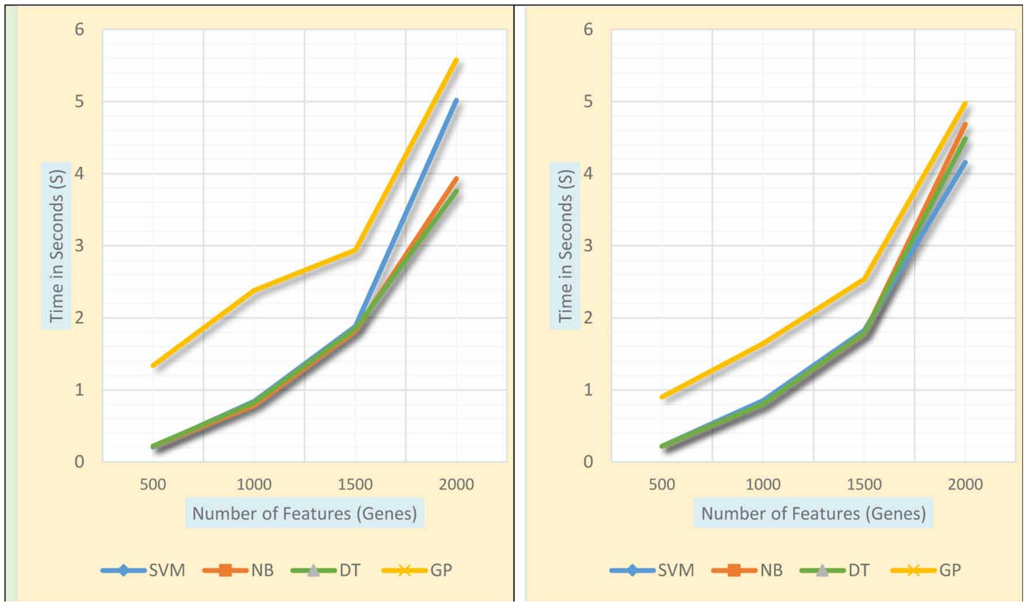


Figure 13 illustrates the classification accuracy results of applying the CfsSubsetEval method using the binary PSO algorithm. PSO/DT achieved the highest classification accuracy (86%) when the data were pre-processed using the normalisation technique. However, if the data were pre-processed using the discretization technique the highest accuracy was 77% with PSO/SVM, PSO/NB and PSO/DT, using the same method. The analysis of time complexity function for PSO/DT was $T(n)=mn+ mn^2$, i.e., $O(n^2)$. The DT has polynomial/quadratic time complexity notation. It is apparent that PSO/NB had the least classification accuracy when the data were normalised (58%), whereas GA/GP was the least when the data were discretised (75%).

When the WrapperSubsetEval method was applied with binary PSO, as presented in Figure 14, PSO/DT and PSO/SVM achieved the highest classification accuracy when the data were normalised (89%) and wrapped by the SVM classifier. However, PSO/DT achieved the highest accuracy (87%) when the data were discretised and wrapped by the DT classifier. The analysis of time complexity function for PSO/DT is $T(n)=mn+ mn^2$, i.e., $O(n^2)$. The DT had a quadratic time complexity notation. The time complexity function of PSO/SVM is $T(n)=mn+n^3$, i.e., $O(n^3)$. The SVM had a cubic time complexity notation. It is apparent that PSO/NB had the least classification accuracy when the data were normalised (57%), whereas GA/GP had the least classification accuracy when the data were discretised (70%). Generally, PSO/DT was more efficient than PSO/SVM, whereas they resulted in same classification accuracy.

From Figure 15, we see that, if the FilteredSubsetEval evaluation method is applied with the binary PSO with many classifiers, PSO/DT achieved the highest classification accuracy (87%) when the data were discretised by filtering the data with the NB classifier. However, if the data were normalised, the filtering evaluation methods of PSO/DT and PSO/SVM had the highest accuracy of 84% when filtered with the SVM classifier. The time complexity analysis function for PSO/DT is $T(n)=mn+ mn^2$, i.e., $O(n^2)$. The DT had a quadratic time complexity notation. PSO/NB had the least classification accuracy (58%) when the data was normalised, whereas PSO/GP had the least when the data was discretised (70%).

Figure 11. Average time of classification for different gene samples using the WrapperSubsetEval method with GA

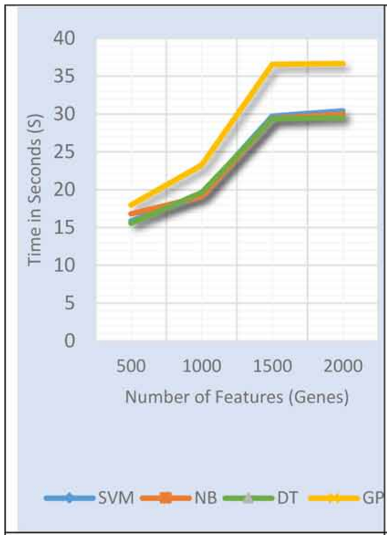
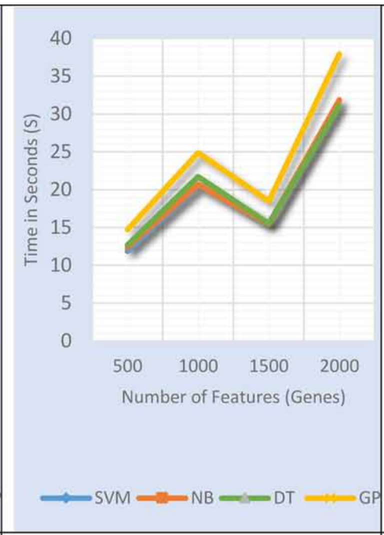
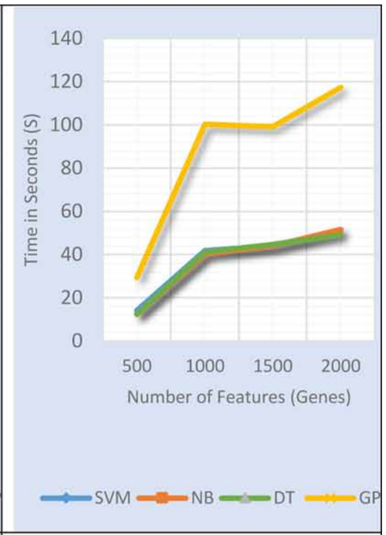
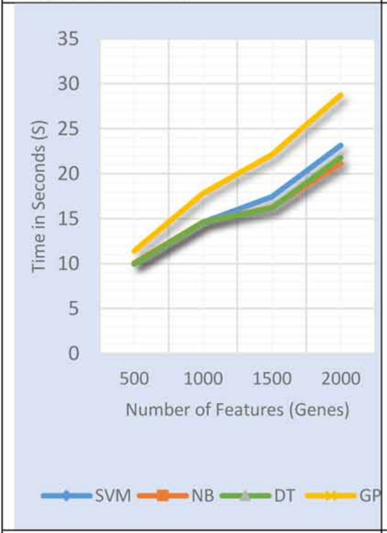
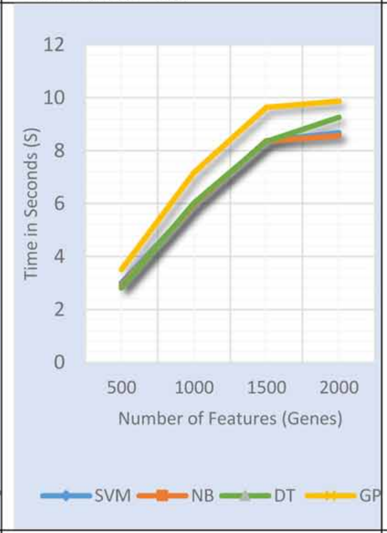
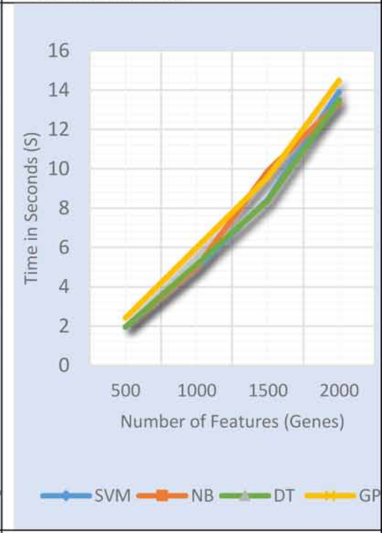
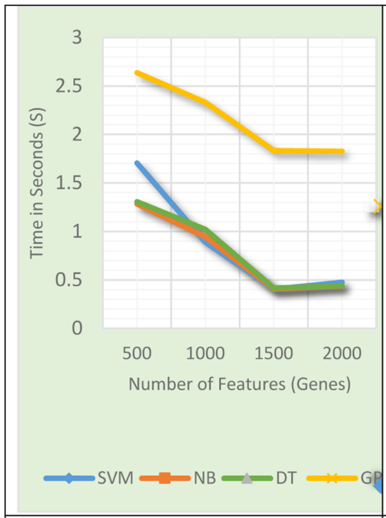
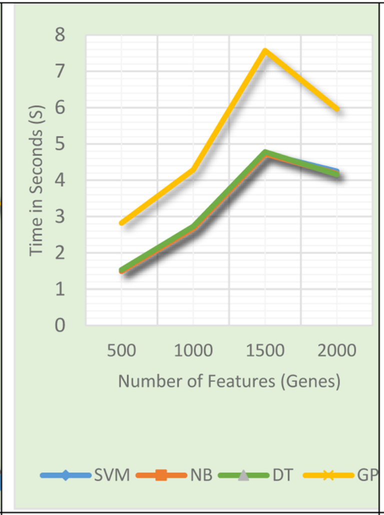
		
(a) WrapperSubsetEval (SVM) -Normalised Data	(b) WrapperSubsetEval (NB) -Normalised Data	(c) WrapperSubsetEval (DT) -Normalised Data
		
(d) WrapperSubsetEval (SVM) -Discretised Data	(e) WrapperSubsetEval (NB) -Discretised Data	(f) WrapperSubsetEval (DT) -Discretised Data

Figure 16 (a and b) presents the time analysis of the PSO algorithm using the CfsSubsetEvel method for normalised and discretised data. PSO/GP took more time (~8 s) when the number of genes/features increased with a time complexity function, $T(n)=mn+n \log n$, i.e., $O(n \log n)$, resulting in low classification accuracy. However, PSO/DT required less time (~5 s) with a time complexity function, $T(n)=mn+mn^2$, i.e., $O(n^2)$, resulting in better accuracy (85%). However, PSO/SVM and PSO/NB presented competitive times (~5 s) but resulted in less classification accuracy compared to PSO/DT, whereas PSO/GP showed efficient time complexity compared to PSO/DT, it resulted in less accurate classification output.

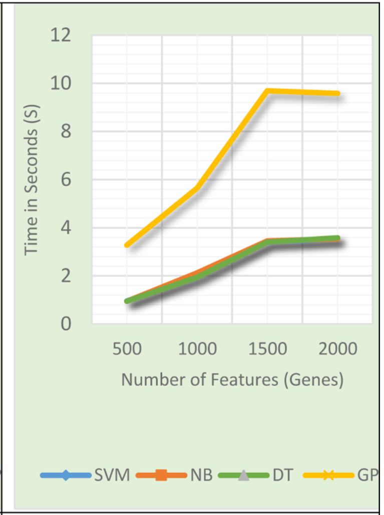
Figure 12. Average time of classification for different gene samples using the FilteredSubsetEval method with GA



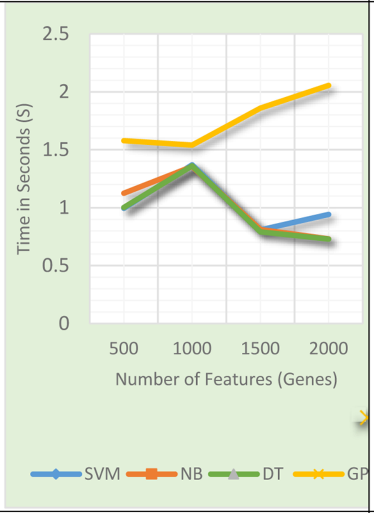
(a) FilteredSubsetEval (SVM) -Normalised Data



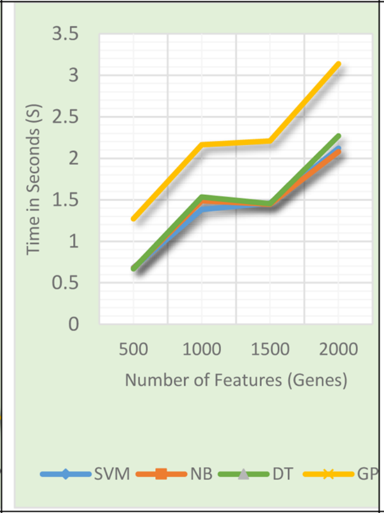
(b) FilteredSubsetEval (NB) -Normalised Data



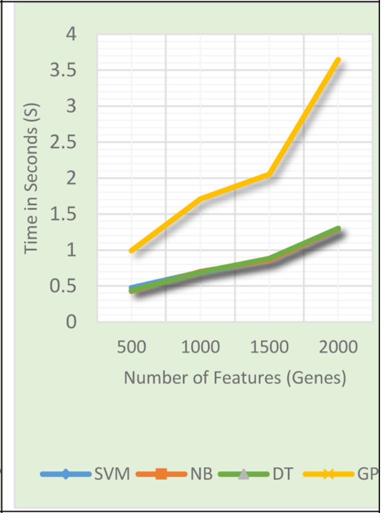
(c) FilteredSubsetEval (DT) -Normalised Data



(d) FilteredSubsetEval (SVM) -Discretised Data



(e) FilteredSubsetEval (NB) -Discretised Data



(f) FilteredSubsetEval (DT) -Discretised Data

Figure 17 (a, b and c) presents the time taken to evaluate the PSO algorithm using the WrapperSubsetEval method on normalised data. Figure 17 (e, f and g) presents the same for discretised data. The time taken by the discretised data was lower than the normalised data. When the data were discretised, the best accuracy was found by PSO/DT (87%) with a time (~14 s), and $O(n^2)$ However, if the data were normalised, the best accuracy (89%) was found by PSO/SVM with $O(n^3)$, and PSO/DT with $O(n^2)$ and times ~22–23 s. Whereas the accuracy differed little for normalised data, it took more time, compared to discretised data. Generally, the discretised data in PSO/DT showed more efficient results than PSO/SVM.

Figure 13. Classification accuracy average of multiple ML algorithms applied by the CFS method employing the PSO as a feature selection algorithm

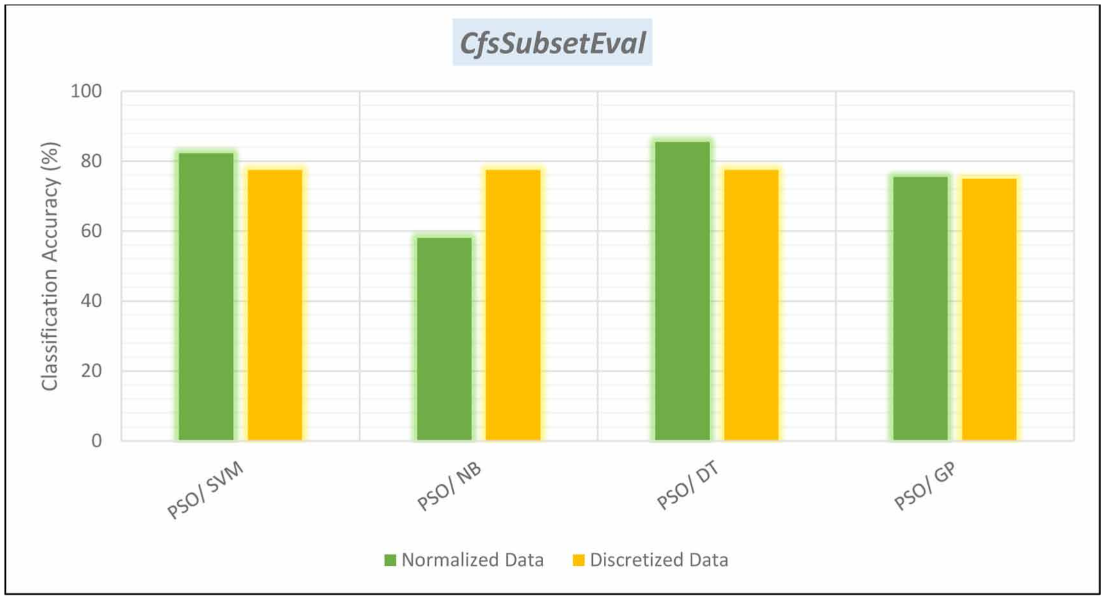


Figure 14. Classification accuracy average of multiple ML algorithms applied by the WrapperSubsetEval method employing PSO as a feature selection algorithm

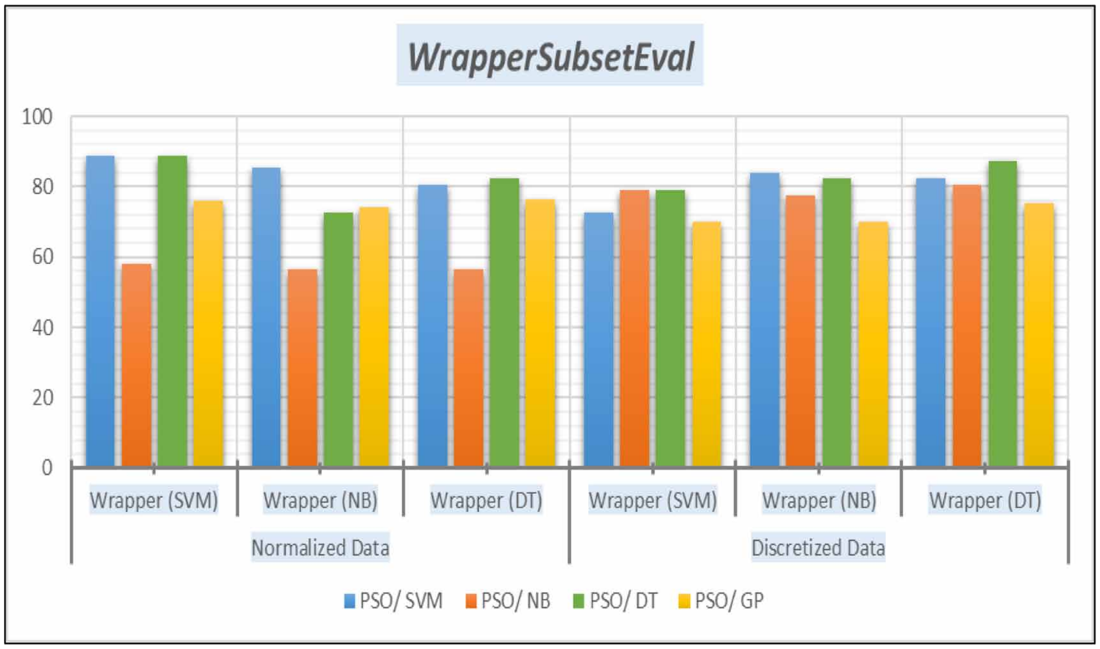


Figure 15. Classification accuracy average of multiple ML algorithms applied by the Filter method employing PSO as a feature selection algorithm

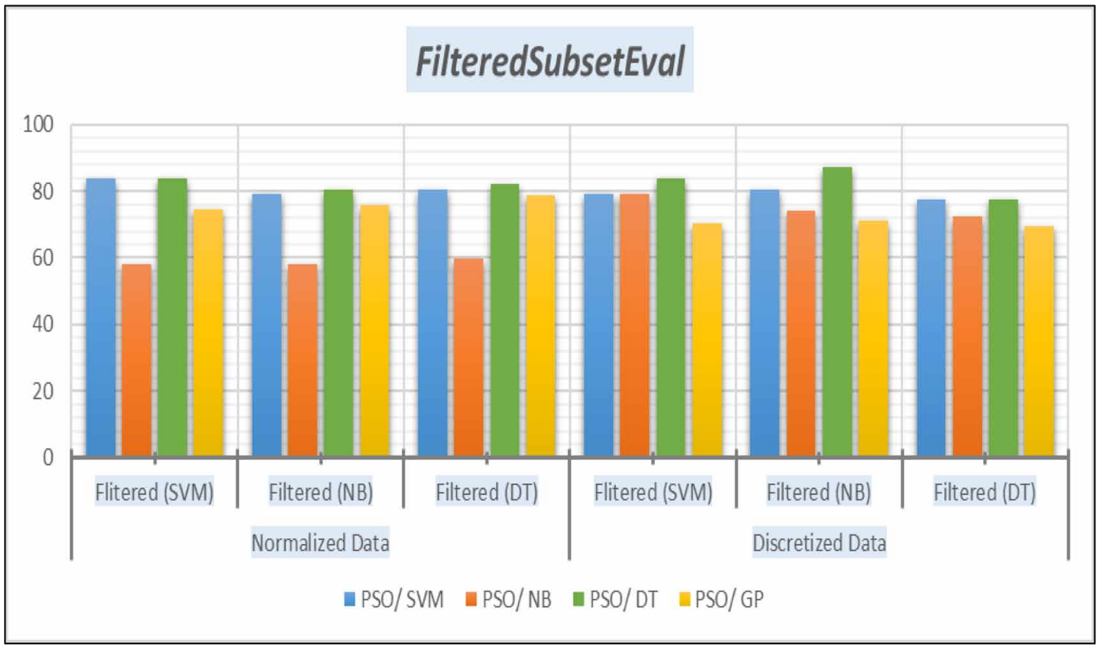
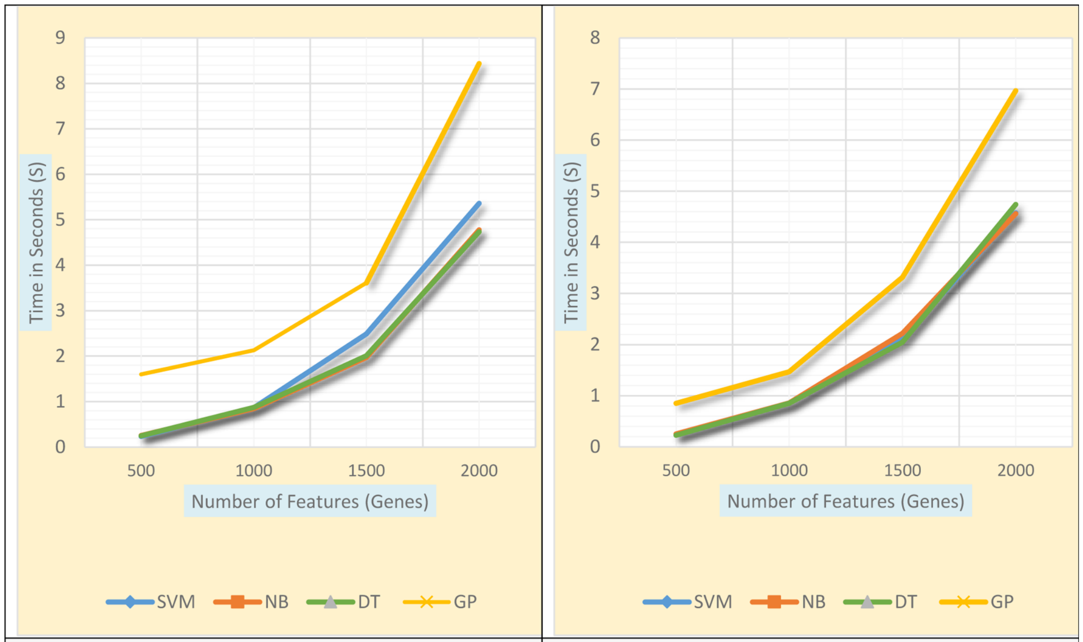
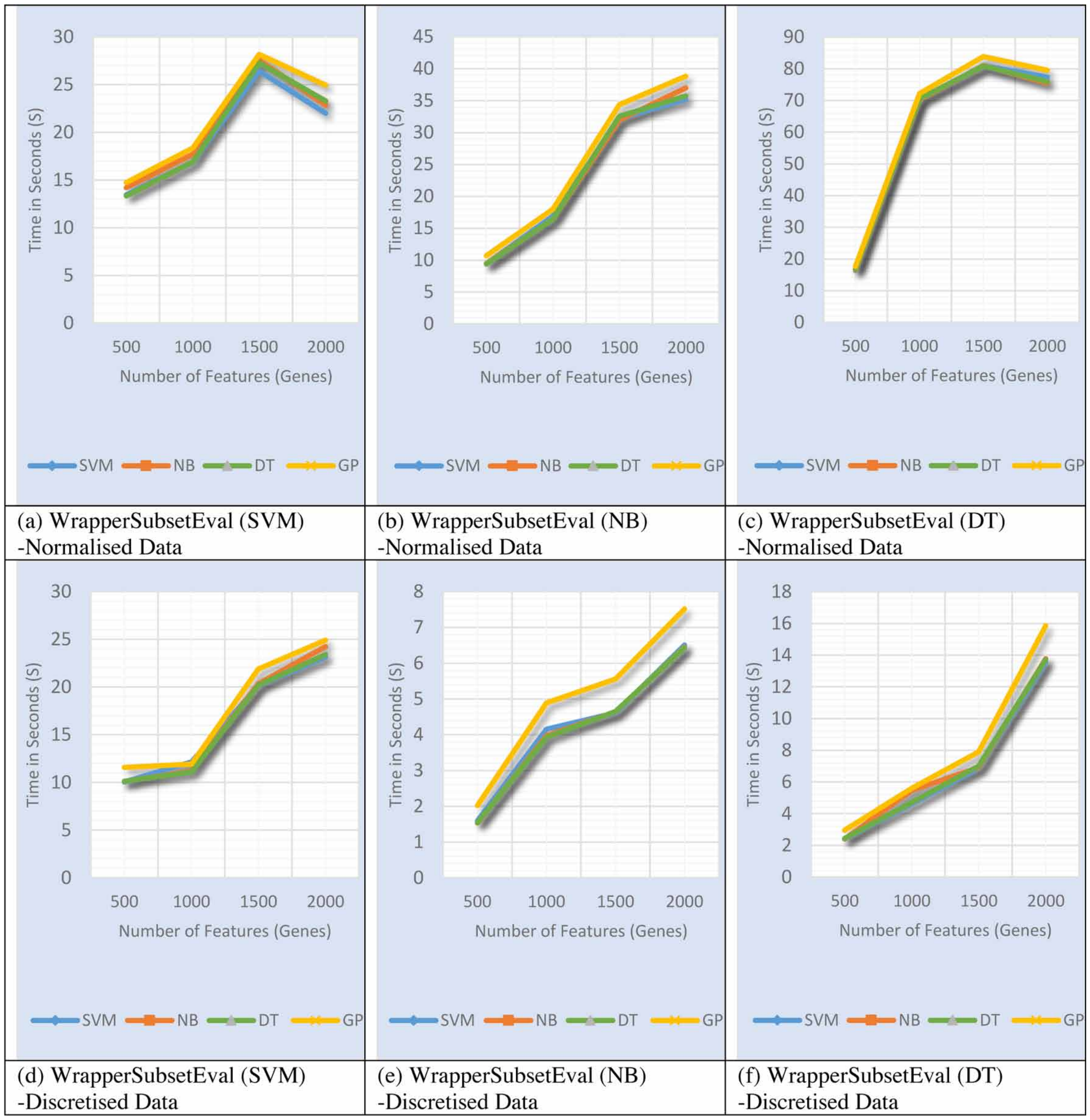


Figure 16. Average time of classification for different gene samples using the CfsSubsetEval method with PSO



(a) Using CfsSubsetEvel method – (Normalised Data)	*(b) Using CfsSubsetEvel method – (Discretised Data)*

Figure 17. Average time of classification for different gene samples using the WrapperSubsetEval method with PSO



(a) WrapperSubsetEval (SVM) -Normalised Data

(b) WrapperSubsetEval (NB) -Normalised Data

(c) WrapperSubsetEval (DT) -Normalised Data

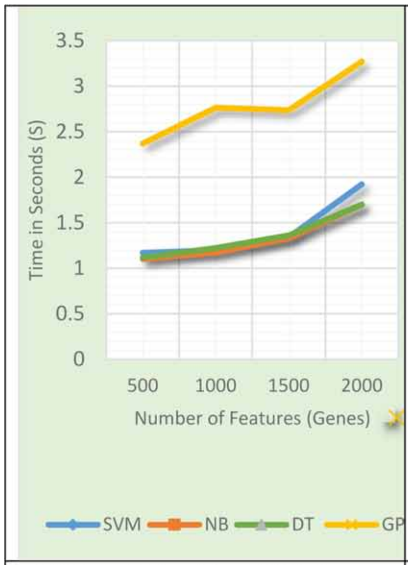
(d) WrapperSubsetEval (SVM) -Discretised Data

(e) WrapperSubsetEval (NB) -Discretised Data

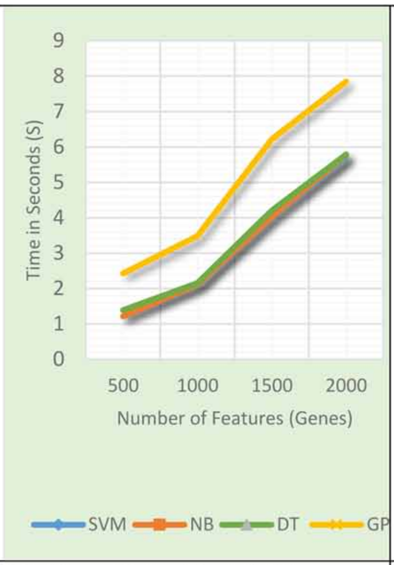
(f) WrapperSubsetEval (DT) -Discretised Data

Figure 18 (a, b and c) presents the time taken to evaluate the PSO algorithm using the FilteredSubsetEval method on normalised data. Figure 18 (e, f and g) presents the same for discretised data. The time taken by discretised data was much lower than the normalised data. When the data was discretised, the best accuracy was found by PSO/DT (87%) with a time ~ 2 s. However, if the data were normalised, the best accuracy was found by PSO/SVM and PSO/DT (84%) with time ~ 2 s, whereas the time required was the same for both pre-processing techniques, the discretised data showed better accuracy than the normalised data. PSO/ DT was $O(n^2)$, whereas PSO/ SVM was $O(n^3)$.

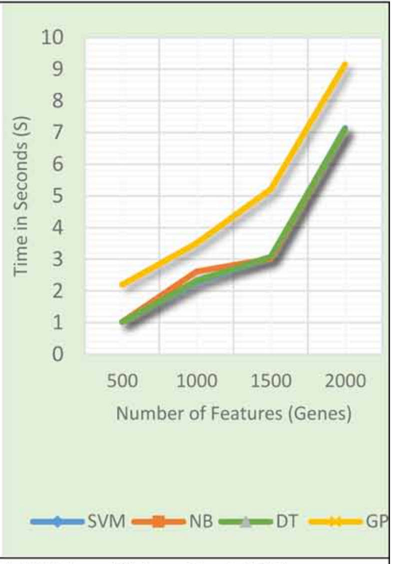
Figure 18. Average time of classification for different gene samples using the FilteredSubsetEval method with PSO



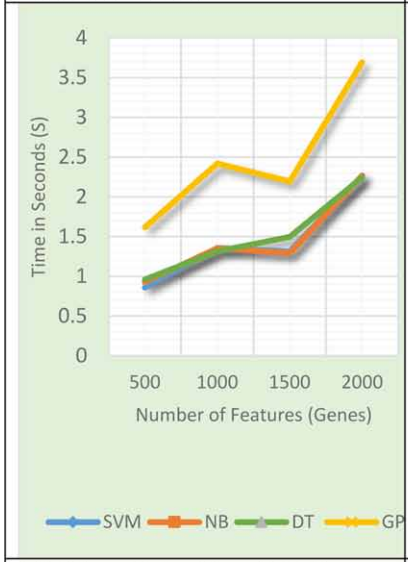
(a) FilteredSubsetEval (SVM) -Normalised Data



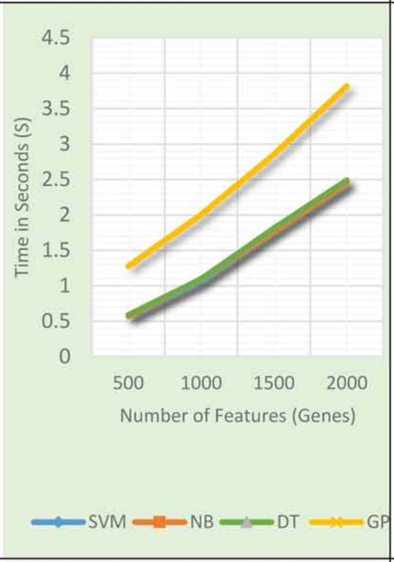
(b) FilteredSubsetEval (NB) -Normalised Data



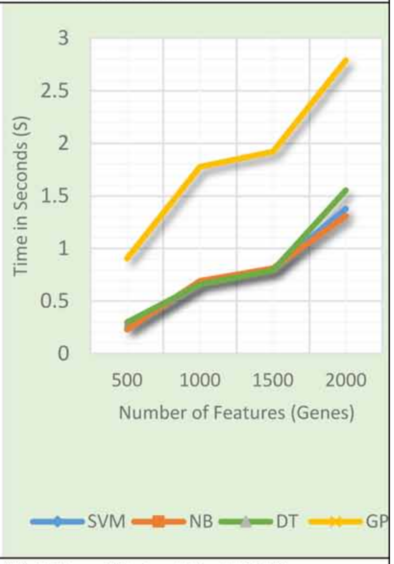
(c) FilteredSubsetEval (DT) -Normalised Data



(d) FilteredSubsetEval (SVM) -Discretised Data



(e) FilteredSubsetEval (NB) -Discretised Data



(f) FilteredSubsetEval (DT) -Discretised Data

When applying IG, Figure 19 shows that IG/DT (89%) performed more accurately on average than others if the data was pre-processed using a normalised technique. The time complexity function of IG/DT is $T(n)=n \log n+mn^2$, i.e., $O(n^2)$. If the data was pre-processed using the discretization techniques (though IG performs self descritisation), the IG/DT still had a good classification accuracy of 87%. Overall, IG/DT outperformed all other algorithms in relation to classification accuracy with the full dataset. However, IG/SVM and IG/GP presented good results (80%–82%) on average.

Figure 19. Average of classification accuracy employing IG (normalised data)

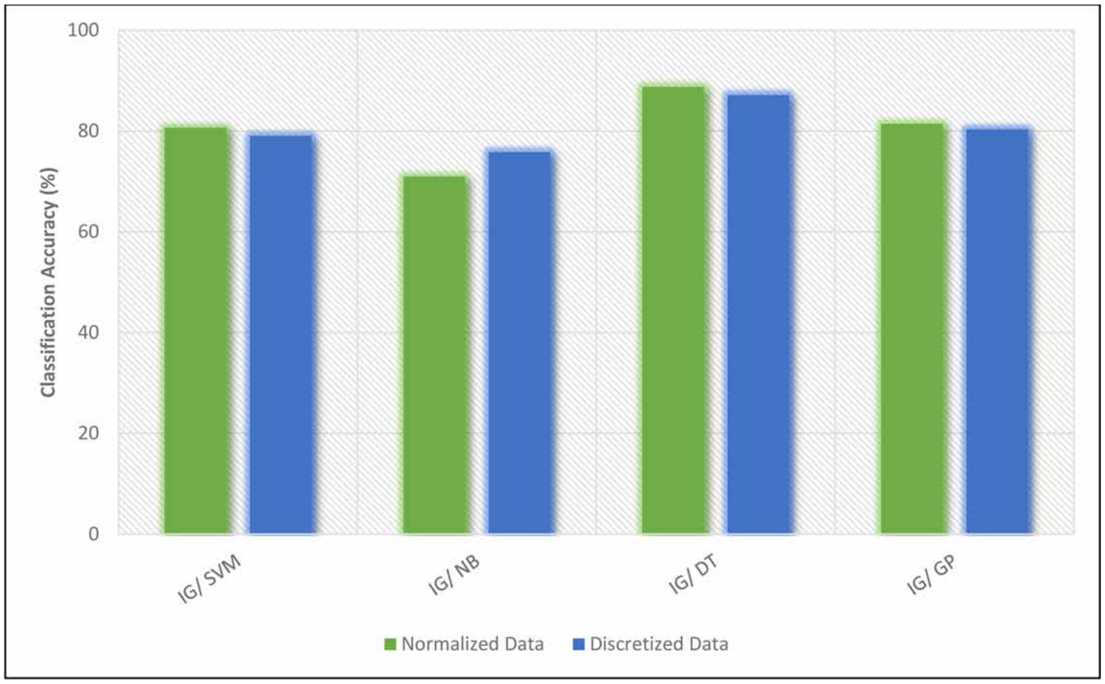
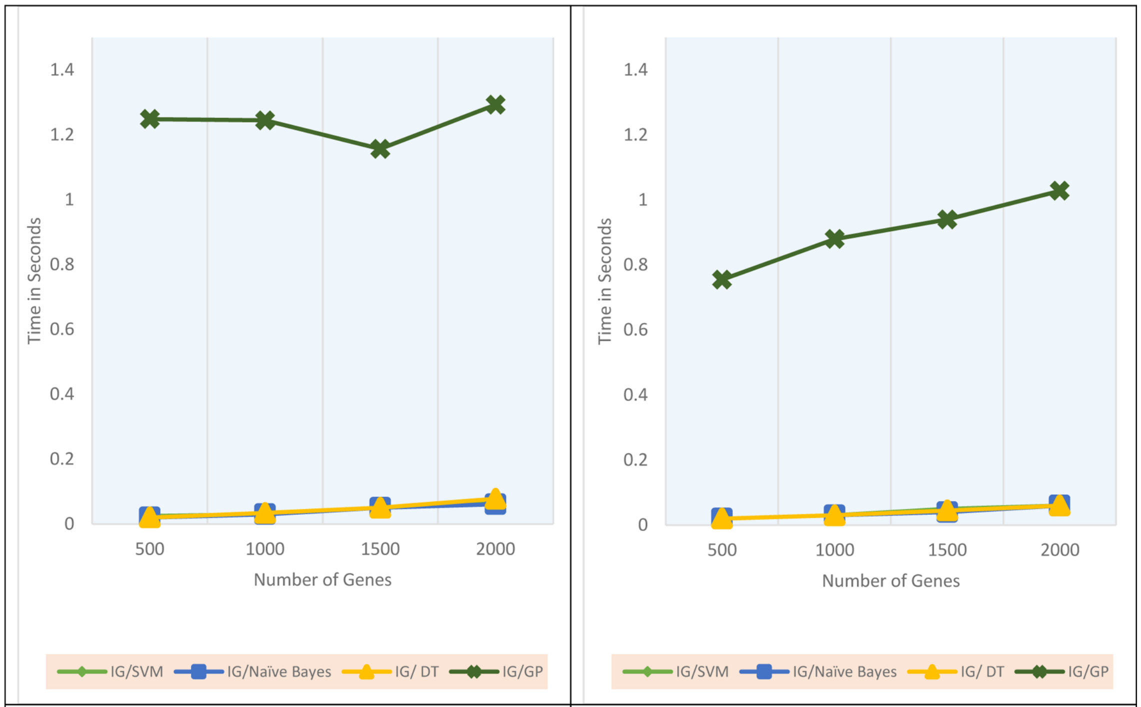


Figure 20 demonstrates that, with either data pre-processing technique and a full dataset, IG/SVM, IG/NB and IG/DT were nearly instantaneous. However, IG/GP took more time (>1 s) if the full entire dataset was selected. More details on the time recorded for all ML algorithms implemented to evaluate the classifiers and build up the model can be found in Appendix 2.

3. RESULTS OF CASE STUDY TWO: 'A TWO-STAGE HYBRID MULTIFILTER FEATURE SELECTION METHOD FOR HIGH COLON-CANCER CLASSIFICATION'

These experiments were implemented in two phases. Phase one implemented the classification algorithms over the features selected by the hybridised selection algorithms (GA+IG), whereas Phase two classified the features using the proposed two-stage multifilter selection system. In this section, the results are presented as follows. Section 3.1 demonstrates the number of selected features for both phases, and Section 3.2 examines the classification accuracy of each phase.

Figure 20. Average time of classification for different gene samples using the CfsSubsetEval method with PSO



(a) Using Information Gain – (Normalised Data)

(b) Using Information Gain – (Discretised Data)

3.1. Number of Selected Features

Phase One

The number of selected features is presented in Table 7 using both phases. Phase one selected 62 related and informative genes/features from the original data set (2000 genes/features) when applying hybridised between GA and IG, thus reducing 97% of the data.

Table 7. Number of selected features by the proposed method

	Colon Features
Full Data Set	2000
Phase 1 (GA+IG)	62
Phase 2 (Phase 1 + mRMR)	15

Phase Two

In this phase, the mRMR method was applied to refine the data selected from Phase one. This method evaluated each gene/feature and sorted them according to the mRMR procedure to enhance the result, eliminating redundant genes/features and selecting more informative and related ones. The top relevant genes were then selected to formulate a new filtered set. From Table 7, the number of genes/features selected was 15. These were more related and the dataset was reduced 24% more.

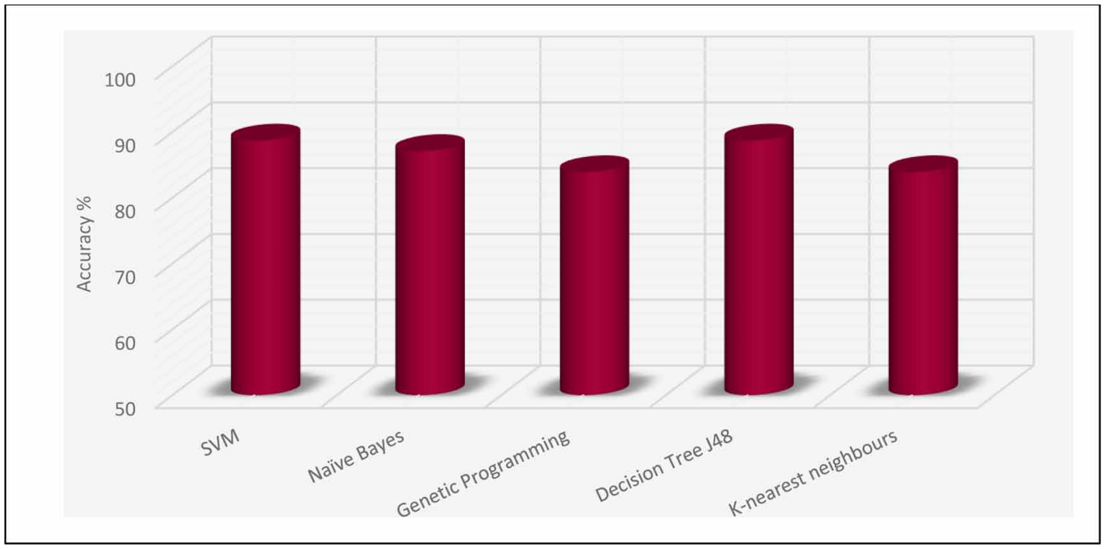
3.2. Classification Accuracy

To evaluate the classification performance of cancer detection using our proposed method, we tested the selected data using five related ML algorithms (i.e., SVM, NB, GP, DT and K-NN) as analysed below.

Phase One

These classification results are those observed after applying the selection algorithms of Phase one (GA + IG). As shown in Figure 21, we notice that classifying the data using SVM or DT had the highest classification accuracy of 89%. However, classifying with ML of K-NN and GP resulted in the lowest accuracy (84.00%).

Figure 21. Classification accuracy using hybrid stage method only – phase one



Phase Two

Figure 2 presents the classification results recorded at this phase. We found that that SVM classification algorithm achieved a higher classification accuracy (94%) when implemented using the proposed method (i.e., two-stage selection prior to classification), whereas DT and GP classification algorithms had the lowest classification accuracy (89%). However, they showed generally good results. Particularly, NB and K-NN had an acceptable classification accuracy of 90%.

4. SUMMARY OF THE RESULTS

4.1. Case Study One

This case study examined the most common feature/ gene selection and machine learning classification algorithms performance applied over the colon cancer dataset. The main target was to establish a systematic methodology to compare the accuracy and analyse the time efficiency for the variety of algorithms implemented in order to specify which algorithm is accurate and has fast prediction analysis toward the colon cancer detection and tumour classification. A three-phased experimental design was implemented as discussed earlier and following are the summary of the results:

Phase One examined the difference between the variety of feature selection algorithms and found that IG had selected the least number of features (135 genes/ features). However, using the CfsSubsetEval evaluation method, it is found that GA had significantly reduced the genes (444 genes/ features) compared to PSO algorithm (1255 genes/ features). In addition, it is found that applying the FilteredSubsetEval evaluation method, GA had also showed a better reduction (205 genes/ features) in comparison to the PSO algorithm (211 genes/ features). While the PSO had resulted in less reduction (187 genes/ features) compared to GA (400 genes/ features) when applying the WrapperSubsetEval evaluation method.

In Phase Two, classification algorithms were implemented independently without any contribution of the feature selection algorithms. It is found that SVM algorithm resulted in better accuracy (86%) if the data were normalized with $O(n^3)$, but if the data were discretized, then the DT algorithm resulted with better classification accuracy (87%) compared to the others with $O(n^2)$.

In Phase Three, a comparison between the integration of both feature selection and classification algorithms was originated. It is found that the integration of the GA/ DT had achieved the best performance with better accuracy (92%) and time analysis of $T(n)=n^2+ mn^2$ with $O(n^2)$. However, IG/DT had showed an accuracy of (89%) with time complexity function of $T(n)=n \log n+mn^2$, i.e., $O(n^2)$. Moreover, PSO/DT and PSO/SVM achieved the highest classification accuracy (89%), but PSO/DT is considered more efficient than PSO/SVM with $O(n^2)$.

4.2. Case Study Two

A new two-stage feature selection model was presented as an approach to construct a better classification accuracy for the colon cancer dataset. The experiments of this research were conducted in two phases as illustrated earlier and following are the summary of the results:

Phase one, applied a hybrid feature selection algorithm (GA+IG), which dramatically reduced the number of features to almost (97%) selecting only (62 genes/features) from the original data set (2000 genes/features). It is found that SVM and DT had a better accuracy of (89%) at this phase.

In Phase two, number of genes was reduced to (15 genes out of 2000 original genes) using the mRMR method. It is found that SVM achieved the highest classification accuracy of (94%) at this phase.

APPENDIX 1

Information Gain Top Ranked Genes

Table 8. Top genes ranked and selected by the Information Gain algorithm

No.	Ranked Attribute	No.	Ranked Attribute	No.	Ranked Attribute	No.	Ranked Attribute
1	0.435 attribute1671	35	0.229 attribute1900	69	0.184 attribute779	103	0.168 attribute1897
2	0.384 attribute249	36	0.229 attribute1047	70	0.184 attribute266	104	0.168 attribute1674
3	0.375 attribute493	37	0.229 attribute377	71	0.182 attribute1775	105	0.166 attribute360
4	0.356 attribute765	38	0.228 attribute571	72	0.182 attribute763	106	0.166 attribute1381
5	0.334 attribute1772	39	0.219 attribute581	73	0.182 attribute639	107	0.166 attribute1790
6	0.32 attribute625	40	0.216 attribute47	74	0.182 attribute1200	108	0.165 attribute1473
7	0.317 attribute1042	41	0.213 attribute1972	75	0.182 attribute1248	109	0.159 attribute830
8	0.315 attribute1423	42	0.213 attribute427	76	0.182 attribute15	110	0.159 attribute228
9	0.305 attribute513	43	0.213 attribute515	77	0.182 attribute1466	111	0.159 attribute1562
10	0.305 attribute1771	44	0.213 attribute143	78	0.177 attribute286	112	0.159 attribute201
11	0.303 attribute267	45	0.211 attribute1411	79	0.177 attribute1110	113	0.159 attribute853
12	0.303 attribute245	46	0.208 attribute365	80	0.174 attribute391	114	0.159 attribute570
13	0.298 attribute258	47	0.207 attribute964	81	0.174 attribute1002	115	0.159 attribute576
14	0.293 attribute780	48	0.203 attribute1414	82	0.172 attribute834	116	0.159 attribute682
15	0.28 attribute399	49	0.201 attribute1967	83	0.172 attribute1387	117	0.159 attribute1875
16	0.275 attribute897	50	0.2 attribute62	84	0.171 attribute802	118	0.159 attribute279
17	0.27 attribute1582	51	0.199 attribute1494	85	0.171 attribute1843	119	0.159 attribute1406
18	0.262 attribute1293	52	0.198 attribute1917	86	0.171 attribute1637	120	0.154 attribute1965
19	0.252 attribute652	53	0.197 attribute1346	87	0.169 attribute187	121	0.154 attribute347
20	0.245 attribute1227	54	0.197 attribute163	88	0.169 attribute31	122	0.154 attribute1197
21	0.245 attribute1153	55	0.197 attribute590	89	0.169 attribute1659	123	0.154 attribute105
22	0.245 attribute43	56	0.197 attribute994	90	0.169 attribute1672	124	0.154 attribute1043
23	0.245 attribute1867	57	0.197 attribute1442	91	0.169 attribute1943	125	0.154 attribute1030
24	0.245 attribute1808	58	0.197 attribute138	92	0.169 attribute1770	126	0.154 attribute79
25	0.245 attribute66	59	0.197 attribute824	93	0.169 attribute992	127	0.154 attribute433
26	0.245 attribute822	60	0.19 attribute1067	94	0.168 attribute1959	128	0.154 attribute1412
27	0.245 attribute1060	61	0.19 attribute812	95	0.168 attribute91	129	0.154 attribute317
28	0.235 attribute1679	62	0.188 attribute1115	96	0.168 attribute287	130	0.154 attribute1560
29	0.235 attribute1325	63	0.188 attribute1472	97	0.168 attribute54	131	0.154 attribute1870
30	0.235 attribute467	64	0.186 attribute1730	98	0.168 attribute127	132	0.154 attribute453
31	0.233 attribute1892	65	0.186 attribute1935	99	0.168 attribute1039	133	0.154 attribute1366
32	0.23 attribute415	66	0.184 attribute1328	100	0.168 attribute1887	134	0.154 attribute495
33	0.229 attribute1635	67	0.184 attribute440	101	0.168 attribute26	135	0.154 attribute107
34	0.229 attribute72	68	0.184 attribute75	102	0.168 attribute1256		

APPENDIX 2

Time to Build Classifier Models

Time (s) recorded for all ML algorithm implemented to evaluate the classifiers and build up the model.

Table 9. Time taken (s) by the GA to classify the data and build up the model

Applying the Genetic Algorithm										
Average CFS Normalised Timing						Average CFS Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	0.207	0.217	0.22	1.338		500	0.219	0.218	0.22	0.901
1000	0.842	0.786	0.823	2.383		1000	0.855	0.805	0.805	1.643
1500	1.879	1.82	1.849	2.94		1500	1.827	1.782	1.788	2.538
2000	5.019	3.931	3.759	5.579		2000	4.157	4.686	4.485	4.976
Average Wrapper (SVM) Normalised Timing						Average Wrapper (SVM) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	15.859	16.809	15.566	17.971		500	10.001	10.059	9.946	11.401
1000	19.405	19.113	19.644	23.254		1000	14.606	14.637	14.6	17.81
1500	29.736	29.396	29.381	36.576		1500	17.419	16.221	16.173	22.13
2000	30.421	29.982	29.506	36.69		2000	23.153	21.15	21.791	28.745
Average Wrapper (NB) Normalised Timing						Average Wrapper (NB) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	11.866	12.352	12.68667	14.718		500	3.005	2.897	2.817	3.509
1000	20.635	20.668	21.733	24.885		1000	5.942	5.969	6.027	7.174
1500	15.558	15.433	15.522	18.473		1500	8.383	8.371	8.345	9.635
2000	31.754	31.874	31.068	37.945		2000	8.678	8.565	9.262	9.872
Average Wrapper (DT) Normalised Timing						Average Wrapper (DT) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	14.115	12.381	12.411	29.481		500	1.957	1.945	1.943	2.414
1000	41.828	40.258	41.141	100.274		1000	4.982	4.985	5.138	6.001
1500	43.993	44.046	44.724	99.213		1500	8.437	9.898	8.38	9.565
2000	51.071	51.597	49.079	117.301		2000	13.915	13.342	13.501	14.495
Average Filtered (SVM) Normalised Timing						Average Filtered (SVM) Discretized Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	1.705	1.283	1.306	2.636		500	0.995	1.124	1.001	1.578
1000	0.889	0.942	1.02	2.33		1000	1.37	1.355	1.361	1.54
1500	0.4041	0.408	0.416	1.833		1500	0.808	0.811	0.791	1.858
2000	0.474	0.436	0.44	1.826		2000	0.942	0.731	0.729	2.055

continues on following page

Table 9. Continued

Applying the Genetic Algorithm										
Average Filtered (NB) Normalised Timing						Average Filtered (NB) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	1.5	1.494	1.535	2.822		500	0.683	0.675	0.669	1.271
1000	2.672	2.672	2.726	4.286		1000	1.383	1.489	1.532	2.163
1500	4.733	4.726	4.78	7.568		1500	1.459	1.445	1.452	2.21
2000	4.245	4.181	4.16	5.966		2000	2.12	2.079	2.269	3.138
Average Filtered (DT) Normalised Timing						Average Filtered (DT) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	0.956	0.944	0.946	3.275		500	0.475	0.427	0.436	0.991
1000	1.954	2.127	1.948	5.652		1000	0.69	0.7	0.693	1.71
1500	3.404	3.457	3.402	9.691		1500	0.849	0.856	0.879	2.05
2000	3.547	3.538	3.58	9.578		2000	1.284	1.291	1.3	3.645

Table 10. Time taken (s) by the Binary PSO to classify the data and build up the model

Applying the Particle Swarm Optimisation										
Average CFS Normalised Timing						Average CFS Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	0.234	0.255	0.247	1.6		500	0.226	0.252	0.227	0.851
1000	0.865	0.851	0.872	2.131		1000	0.85	0.855	0.85	1.468
1500	2.489	1.973	2.008	3.611		1500	2.123	2.22	2.044	3.316
2000	5.361	4.777	4.735	8.435		2000	4.579	4.562	4.742	6.967
Average Wrapper (SVM) Normalised Timing						Average Wrapper (SVM) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	13.456	14.206	13.339	14.74		500	10.049	10.106	10.119	11.577
1000	16.874	17.699	16.922	18.381		1000	12.161	11.196	11.116	11.903
1500	26.421	27.581	27.282	28.171		1500	20.149	20.346	20.195	21.892
2000	22.01	22.852	23.275	24.935		2000	23.145	24.23	23.395	24.911
Average Wrapper (NB) Normalised Timing						Average Wrapper (NB) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	9.423	9.48	9.397	10.699		500	1.597	1.532	1.544	2.021
1000	16.88	16.406	16.418	17.995		1000	4.156	3.975	3.93	4.892
1500	32.149	31.963	32.547	34.355		1500	4.614	4.639	4.65	5.558
2000	35.194	36.99	35.729	38.827		2000	6.506	6.451	6.434	7.515

continues on following page

Table 10. Continued

Applying the Particle Swarm Optimisation										
Average Wrapper (DT) Normalised Timing						Average Wrapper (DT) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	16.337	16.506	16.761	17.684		500	2.401	2.399	2.431	2.956
1000	70.304	70.247	70.496	72.212		1000	4.614	5.465	4.702	5.606
1500	81.038	81.032	80.888	83.869		1500	6.869	6.908	6.995	7.894
2000	77.357	75.382	75.935	79.561		2000	13.562	13.774	13.711	15.849
Average Filtered (SVM) Normalised Timing						Average Filtered (SVM) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	1.17	1.101	1.119	2.371		500	0.857	0.926	0.959	1.617
1000	1.2	1.173	1.223	2.763		1000	1.355	1.352	1.312	2.417
1500	1.342	1.332	1.36	2.736		1500	1.311	1.29	1.496	2.192
2000	1.923	1.7	1.701	3.271		2000	2.263	2.267	2.244	3.695
Average Filtered (NB) Normalised Timing						Average Filtered (NB) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	1.216	1.214	1.389	2.426		500	0.571	0.568	0.586	1.275
1000	2.111	2.118	2.15	3.483		1000	1.078	1.097	1.095	2.002
1500	4.039	4.042	4.201	6.222		1500	1.788	1.773	1.811	2.854
2000	5.707	5.768	5.787	7.847		2000	2.464	2.452	2.488	3.816
Average Filtered (DT) Normalised Timing						Average Filtered (DT) Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	1.026	1.026	1.018	2.198		500	0.25	0.231	0.297	0.905
1000	2.288	2.619	2.325	3.507		1000	0.681	0.691	0.66	1.776
1500	3.062	3.02	3.086	5.215		1500	0.806	0.811	0.791	1.92
2000	7.146	7.11	7.112	9.158		2000	1.378	1.31	1.553	2.789

Table 11. Time taken (s) by the IG to classify the data and build up the model

Applying the Information Gain Algorithm										
Average CFS Normalised Timing						Average CFS Discretised Timing				
#Genes	SVM	NB	DT	GP		#Genes	SVM	NB	DT	GP
500	0.025	0.02	0.02	1.249		500	0.02	0.02	0.02	0.755
1000	0.03	0.03	0.034	1.245		1000	0.03	0.03	0.03	0.88
1500	0.05	0.05	0.05	1.157		1500	0.05	0.04	0.045	0.94
2000	0.066	0.061	0.078	1.293		2000	0.06	0.06	0.06	1.028

Chapter 6
Analysis, Discussion, and Evaluations for the Case Studies

ABSTRACT

The purpose of this chapter is to discuss and analyse the results produced in Chapter 5. To evaluate the proposed models, this chapter compares the models with others existing in the literature. Additionally, the chapter discusses the evaluation measures used to validate the experimental results of Chapter 5. For example, from experiments, GA/DT demonstrated the highest average accuracy (92%) for classifying colon cancer, compared with other algorithms. PSO/DT presented 89%, PSO/SVM presented 89%, and IG/DT presented 89%, demonstrating very good classification accuracy. PSO/NB presented 57% and GA/NB presented 58%: less classification accuracy. Table 1 lists all accuracies resulting from experiments of case study one, as applied to the full data set. There are 45 algorithmic incorporation methods that have accuracy above 80% when applied to the full dataset. One algorithm presents an accuracy of 92%. Nine others scored below 60%.

1. INTRODUCTION

The purpose of this chapter is to discuss and analyse the results produced in Chapter 5. To evaluate the proposed models, this chapter compares the models with others existing in the literature. Additionally, the chapter discusses the evaluation measures used to validate the experimental results of Chapter 5.

2. CASE STUDY ONE: ANALYSIS AND DISCUSSION

Based on the first case study results Chapter 5, it was found that most GA selection algorithms outperform other selection methods in extracting and choosing the most relative and informative genes/features. This algorithm can be used to enhance the performance of other classification algorithms. PSO is a competitive algorithm but results in less accurate classification results, whereas IG is efficient for gene-ranking and selecting top-ranked genes.

DOI: 10.4018/978-1-7998-7316-7.ch006

From experiments, GA/DT demonstrated the highest average accuracy (92%) for classifying colon cancer, compared with other algorithms. PSO/DT presented 89%, PSO/SVM presented 89% and IG/DT presented 89%, demonstrating very good classification accuracy. PSO/NB presented 57% and GA/NB presented 58%: less classification accuracy. Table 1 lists all accuracies resulting from experiments of case study one, as applied to the full data set. There are 45 algorithmic incorporation methods that have accuracy above 80% when applied to the full dataset. One algorithm presents an accuracy of 92%. Nine others scored below 60%.

Many methods have been tested for colon-cancer classification accuracy in the literature, as listed in Table 2, Chapter_2, some achieving 90% (Fang, Mustapha, & Sulaiman, 2010; Mohamad, Deris, & Illias, 2005). However, Yeh et al. (2007) achieved 89%. Moreover, one contribution in (Mohamad, Omatu, Deris, & Yoshioka, 2009) achieved 87%. Mohamad, Deris, and Illias (2005), Salem, Attiya, and El-Fishawy (2017), and Kulkarni et al. (2011) reached 85%. Other studies attained 84%, as with Alladi et al. (2008). Many contributors showed lower results (<84%). From case study one (GA/DT achieved a better result (92%). That is, it achieved a better classification accuracy employing all selected features in the colon dataset. It showed better outcome performance compared to other results in the literature, especially when compared with Yeh et al. (2007), who achieved 89%.

To the author's knowledge, the colon-cancer classification studies listed in Table 6, Chapter 2, did not state the algorithmic time efficiency, except for one study by Salem, Attiya, and El-Fishawy (2017), where the time complexity was shown to correspond to $O(n(n \log n{+}n))$ with a classification accuracy of 85.5%. They applied a proposed model composed of IG, followed sGA as a reduction algorithm. They also applied GP. They did not, however, present a comparative relationship study with other models in reference to accuracy and time analysis. The current research examined the accuracy and efficiency (i.e., time complexity) of algorithms using three experimental phases, finding that applying only an individual gene selection algorithm, followed by a classification algorithm (GA/DT), enhanced the classification accuracy (92%), resulting in a more efficient time, $O(n^2)$, as compared to other results in the literature, particularly the accuracy and time analysis work of Salem, Attiya, and El-Fishawy (2017) using the same dataset.

This research presented nearly instantaneous gene classification with $O(n^2)$, whereas Salem, Attiya, and El-Fishawy (2017) is considered slower when applied to perform the same job with $O(n^2\log n)$,. Thus, the proposed algorithm is computationally better. Moreover, both DT and SVM algorithms reflect very good classification accuracy results. However, DT was more efficient and less expensive in terms of time and accuracy.

3. CASE STUDY TWO: ANALYSIS AND DISCUSSION

The experiments applied to the colon cancer dataset showed that the proposed methodology is powerful and stable when assessing classification accuracy, (Table 2. Summary of phase one and phase two results, presents a summary of the classification accuracy outcomes for phases one and two. Phase two (the proposed hybrid multifilter feature selection method) achieved promising results compared to phase one, except with DT, which presented the same result. It is interesting that SVM correctly classifies the data to 94%. The next highest was NB and K-NN (90%). The lowest result was DT and GP (89%).

Table 1. Average accuracy applied on full colon-cancer dataset using all evaluation methods

Selection Methodology	**Data**	**No. of Selected Genes**	**Accuracy (%)**			
			SVM	**NB**	**DT**	**GP**
Genetic Algorithm (GA)						
CFS	Normalised	**444**	80.7	59.7	79.0	74.5
Wrapper (SVM)		**780**	80.7	61.3	83.9	77.7
Wrapper (NB)		**657**	85.5	61.3	79.0	80.7
Wrapper (DT)		**400**	80.7	58.1	83.9	76.9
Filtered (SVM)		**205**	82.3	66.1	82.3	76.5
Filtered (NB)		**295**	80.7	64.5	83.9	74.7
Filtered (DT)		**485**	80.7	61.3	64.5	75.0
CFS	Discretised	**444**	80.7	72.6	**91.9**	74.7
Wrapper (SVM)		**778**	77.4	72.6	82.3	71.8
Wrapper (NB)		**579**	75.8	82.3	80.7	73.7
Wrapper (DT)		**917**	79.0	77.4	85.5	70.5
Filtered (SVM)		**438**	74.2	79.0	78.8	72.6
Filtered (NB)		**370**	72.6	77.4	79.0	74.0
Filtered (DT)		**959**	75.8	80.6	80.6	71.9
Binary Particle Swarm Optimisation (PSO)						
CFS	Normalised	**1255**	82.3	58.1	85.5	75.5
Wrapper (SVM)		**187**	88.7	58.1	88.7	76.1
Wrapper (NB)		**596**	85.5	56.5	72.6	74.2
Wrapper (DT)		**257**	80.6	56.5	82.3	76.3
Filtered (SVM)		**211**	83.9	58.1	83.9	74.7
Filtered (NB)		**634**	79.0	58.1	80.6	75.8
Filtered (DT)		**449**	80.6	59.7	82.3	78.7
CFS	Discretised	**1255**	77.4	77.4	77.4	75.0
Wrapper (SVM)		**713**	72.6	79.0	79.0	70.2
Wrapper (NB)		**315**	83.9	77.4	82.3	70.2
Wrapper (DT)		**730**	82.3	80.6	87.1	75.3
Filtered (SVM)		**619**	79.0	79.0	83.9	70.5
Filtered (NB)		**572**	80.6	74.2	87.1	71.0
Filtered (DT)		**674**	77.4	72.6	77.4	69.5
Information Gain (IG)						
IG Ranker	Normalised	**135**	80.6	71.0	88.7	81.5
IG Ranker	Discretised	**135**	79.0	75.8	87.1	80.3

Finally, to highlight the adequacy of the proposed method, the results were compared to those obtained by related approaches implementing similar algorithms from the literature (Table 7, Chapter 2), including the hybrid selection algorithms and the mRMR filter selection algorithm. It was found that our algorithm achieved better and higher classification accuracies when compared to Abdi, Hosseini, and Rezghi (2012), who used the mRMR and PSO with SVM, achieving a result of 90.32%. Li, Wu, and Tan (2008) applied a hybrid of PSO/GA using SVM as a classifier, achieving an accuracy result of 91.9%. Additionally, a comparison was made with El Akadi et al. (2011) who applied mRMR/GA using SVM, achieving 85.48%. The method of Abdi, Hosseini, and Rezghi (2012) selected fewer genes (10.3), but showed lower classification accuracy than our results. The methods of Li, Wu, and Tan (2008) and El Akadi et al. (2011) selected more genes: 18 and 40, respectively. The classification accuracy achieved by Li, Wu, and Tan (2008) was 91.9%, whereas for El Akadi et al. (2011) it was 85.5%. In contrast, the proposed method, which selected 15 genes/features, achieved a classification accuracy of 94%. To summarise, the existing methods in the literature tended to select either fewer genes or more genes, compared to the proposed method. However, they produced lower classification accuracy utilising the same colon cancer dataset. Thus, our method keeps only the most relevant and informative genes that produce higher classification accuracy than those in the literature.

Table 2. Summary of phase one and phase two results

Classifier	Phase I Acc. [%]	Phase II Acc. [%]
SVM	89	94
NB	87	90
GP	84	89
DT	89	89
K-NN	84	90

4. SUMMARY AND PERFORMANCE RESULTS EVALUATION ANALYSIS

The performance evaluation measures employed for validating the results of the classification tests for case study one are presented in Appendix, due to the many tests conducted during the experiments.

For case study two, Table 3 presents the confusion matrix for the five different classification algorithms evaluated on a dataset of 62 entries/samples. SVM had the highest value in the confusion matrix, following the proposed method for the colon dataset. It correctly classified 20 positive samples and correctly classified 38 negative samples. However, it misclassified 2 positive samples as negative and 2 negative samples as positive.

The confusion matrix in Table 3 helps draw a constructive ROC curve using TPR and FPR. From Figure 1, the *x-axis* is the FPR, and the *y-axis* is the TPR. From the confusion matrix, each prediction result reflects a point on the ROC graph. The result of SVM clearly demonstrates the best performance over NB, GP, DT and KNN. However, GP and DT had the worst predictive performance, as indicated.

Figure 2 illustrates the result of correctly classified instances for all implemented classification algorithms. SVM had the highest correctly classified samples, followed by NB and KNN, each consisting of 19 positive samples and 37 negative samples.

Table 3. Confusion matrix for the proposed model

Confusion Matrix	SVM		NB		GP		DT		K-NN	
Positive	20	2	19	3	18	4	18	4	19	3
Negative	2	38	3	37	3	37	3	37	3	37
TPR	0.2		0.19		0.18		0.18		0.19	
FPR	0.02		0.03		0.03		0.03		0.03	

Figure 1. ROC Space for case study two

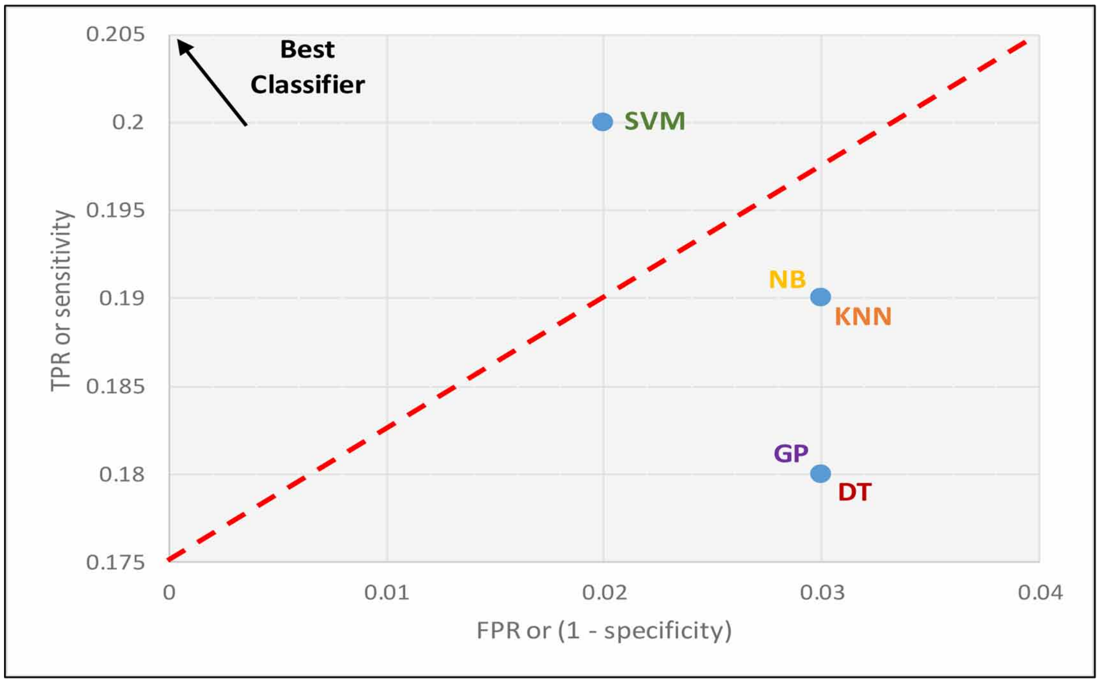


Table 4 presents a result evaluation, including accuracy, sensitivity, specificity, MCC and ROC values of the proposed feature selection method of all classifying algorithms. Sensitivity and specificity range between 81% and 95%, despite the small subset selected genes. However, the sensitivity and specificity of the SVM classifier was higher, obtaining a sensitivity of 0.9 and a specificity of 0.95. Thus, approach is competitive when using the SVM algorithm as a classifier. DT provides sensitivity and specify of 0.818, which is a low performance result.

SVM has the best classification accuracy (94%), but its ROC area (93%) is lower than NB's ROC area of 97.7%. Thus, NB is a good classification option, because its ROC validation result is higher. However, the SVM had the highest MCC of 0.859, validating its ability to predict the correct labels better than other classifiers. DT had the lowest MCC value (0.751), and is considered the least accurate classifier.

Figure 2. Positive and negative samples which are correctly classified

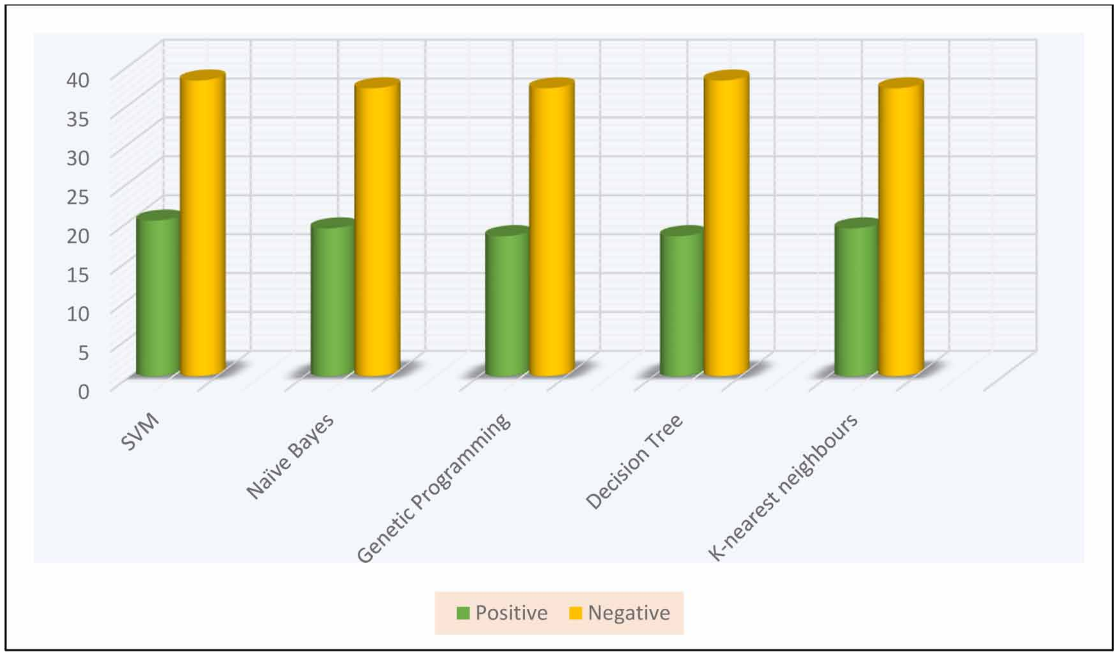


Table 4. Evaluation of the experiment results

Classifier	Accuracy	Sensitivity	Specificity	MCC	ROC
SVM	94	0.909	0.95	0.859	0.930
NB	90	0.864	0.925	0.789	0.977
GP	89	0.832	0.923	0.760	0.894
DT	89	0.818	0.925	0.751	0.868
K-NN	90	0.864	0.925	0.789	0.946

REFERENCES

Abdi, M. J., Hosseini, S. M., & Rezghi, M. (2012). A Novel Weighted Support Vector Machine Based on Particle Swarm Optimization for Gene Selection and Tumor Classification. *Computational and Mathematical Methods in Medicine*, *2012*, 1–7. doi:10.1155/2012/320698 PMID:22924059

Alladi, S. M., Shinde Santosh, P., Ravi, V., & Murthy, U. S. (2008). Colon Cancer Prediction with Genetic Profiles Using Intelligent Techniques. *Bioinformation*, *3*(3), 130–133. doi:10.6026/97320630003130 PMID:19238250

El Akadi, A., Amine, A., El Ouardighi, A., & Aboutajdine, D. (2011). A Two-Stage Gene Selection Scheme Utilizing Mrmr Filter and Ga Wrapper. *Knowledge and Information Systems*, *26*(3), 487–500. doi:10.100710115-010-0288-x

Fang, O. H., Mustapha, N., & Sulaiman, M. N. (2010). *Integrating Biological Information for Feature Selection in Microarray Data Classification*. Paper presented at the Second International Conference on Computer Engineering and Applications (ICCEA), Bali Island, Indonesia. 10.1109/ICCEA.2010.215

Kulkarni, A., Kumar, B. N., Ravi, V., & Murthy, U. S. (2011). Colon Cancer Prediction with Genetics Profiles Using Evolutionary Techniques. *Expert Systems with Applications*, *38*(3), 2752–2757. doi:10.1016/j.eswa.2010.08.065

Li, S., Wu, X., & Tan, M. (2008). *Gene Selection Using Hybrid Particle Swarm Optimization*. Academic Press.

Mohamad, M. S., Deris, S., & Illias, R. M. (2005). A Hybrid of Genetic Algorithm and Support Vector Machine for Features Selection and Classification of Gene Expression Microarray. *International Journal of Computational Intelligence and Applications*, *5*(1), 91–107. doi:10.1142/S1469026805001465

Mohamad, M. S., Omatu, S., Deris, S., & Yoshioka, M. (2009). Particle Swarm Optimization for Gene Selection in Classifying Cancer Classes. *Artificial Life and Robotics*, *14*(1), 16–19. doi:10.100710015-009-0712-z

Salem, H., Attiya, G., & El-Fishawy, N. (2017). Classification of Human Cancer Diseases by Gene Expression Profiles. *Applied Soft Computing*, *50*, 124–134. doi:10.1016/j.asoc.2016.11.026

Yeh, W., Wu, & Chang. (2007). *Applying Data Mining Techniques for Cancer Classification from Gene Expression Data*. Paper presented at the International Conference on Convergence Information Technology (ICCIT 2007), Gyeongju, South Korea. 10.1109/ICCIT.2007.153

APPENDIX

Case Study One Test Evaluation Results

The performance evaluation measures employed for validating the data classification tests for all ML algorithms.

*Table 5. Evaluation results for validating the data using the Genetic Algorithm, * with CFS (CfsrSubsetEval), W (WrapperSubsetEval) and F (FilteredSubsetEval)*

Selection Methodology	Data	GA/ SVM				GA/ NB			
		Sensitivity	Specificity	MCC	ROC	Sensitivity	Specificity	MCC	ROC
CFS	Normalized	0.636	0.9	0.565	0.768	0.682	0.55	0.222	0.686
W(SVM)		0.727	0.85	0.577	0.789	0.773	0.525	0.289	0.704
W(NB)		0.818	0.875	0.687	0.847	0.773	0.525	0.287	0.678
W(DT)		0.727	0.850	0.577	0.789	0.636	0.550	0.178	0.656
F(SVM)		0.682	0.900	0.604	0.791	0.727	0.625	0.337	0.727
F(NB)		0.727	0.850	0.577	0.789	0.864	0.525	0.382	0.760
F(DT)		0.682	0.875	0.570	0.778	0.727	0.550	0.267	0.592
CFS	Discretized	0.682	0.875	0.570	0.778	0.727	0.725	0.436	0.782
W(SVM)		0.682	0.825	0.507	0.753	0.773	0.700	0.453	0.811
W(NB)		0.545	0.875	0.451	0.710	0.773	0.850	0.617	0.813
W(DT)		0.636	0.875	0.531	0.756	0.773	0.775	0.511	0.787
F(SVM)		0.636	0.800	0.436	0.718	0.727	0.825	0.547	0.813
F(NB)		0.546	0.825	0.385	0.661	0.727	0.800	0.518	0.819
F(DT)		0.591	0.850	0.458	0.720	0.773	0.825	0.587	0.828
Selection Methodology	**Data**	**GA/ DT**				**GA/ GP**			
		Sensitivity	Specificity	MCC	ROC	Sensitivity	Specificity	MCC	ROC
CFS	Normalized	0.727	0.825	0.547	0.745	0.572	0.84	0.43	0.707
W(SVM)		0.773	0.875	0.648	0.805	0.599	0.86	0.482	0.73
W(NB)		0.591	0.900	0.527	0.745	0.640	0.898	0.568	0.769
W(DT)		0.727	0.900	0.642	0.797	0.582	0.873	0.480	0.727
F(SVM)		0.682	0.900	0.604	0.807	0.616	0.858	0.496	0.740
F(NB)		0.818	0.850	0.656	0.888	0.568	0.845	0.433	0.707
F(DT)		0.409	0.775	0.194	0.592	0.554	0.848	0.438	0.706
CFS	Discretized	0.909	0.093	0.826	0.924	0.554	0.853	0.428	0.703
W(SVM)		0.727	0.875	0.609	0.790	0.473	0.853	0.354	0.663
W(NB)		0.773	0.825	0.587	0.811	0.441	0.900	0.390	0.670
W(DT)		0.727	0.925	0.677	0.860	0.305	0.925	0.304	0.615
F(SVM)		0.636	0.825	0.466	0.684	0.446	0.880	0.368	0.663
F(NB)		0.682	0.850	0.538	0.760	0.436	0.908	0.399	0.672
F(DT)		0.682	0.875	0.570	0.766	0.436	0.875	0.349	0.656

*Table 6. Evaluation results for validating the data using the Binary PSO, * with CFS (CfsrSubsetEval), W (WrapperSubsetEval) and F (FilteredSubsetEval)*

Selection Methodology	Data	PSO/ SVM				PSO/ NB			
		Sensitivity	Specificity	MCC	ROC	Sensitivity	Specificity	MCC	ROC
CFS	Normalized	0.727	0.875	0.609	0.801	0.727	0.500	0.220	0.649
W(SVM)		0.864	0.900	0.756	0.882	0.864	0.425	0.295	0.713
W(NB)		0.727	0.925	0.677	0.826	0.727	0.475	0.197	0.606
W(DT)		0.727	0.850	0.577	0.789	0.682	0.500	0.175	0.621
F(SVM)		0.818	0.850	0.656	0.834	0.727	0.500	0.220	0.682
F(NB)		0.636	0.875	0.531	0.756	0.727	0.500	0.220	0.637
F(DT)		0.727	0.850	0.577	0.789	0.727	0.525	0.243	0.715
CFS	Discretized	0.682	0.825	0.507	0.753	0.773	0.775	0.531	0.821
W(SVM)		0.636	0.775	0.407	0.706	0.773	0.800	0.559	0.824
W(NB)		0.682	0.925	0.640	0.803	0.682	0.825	0.507	0.835
W(DT)		0.682	0.900	0.604	0.791	0.773	0.825	0.587	0.819
F(SVM)		0.636	0.875	0.531	0.756	0.773	0.800	0.559	0.809
F(NB)		0.773	0.825	0.587	0.799	0.682	0.775	0.449	0.852
F(DT)		0.591	0.875	0.491	0.733	0.682	0.750	0.421	0.816
Selection Methodology	**Data**	**PSO/ DT**				**PSO/ GP**			
		Sensitivity	Specificity	MCC	ROC	Sensitivity	Specificity	MCC	ROC
CFS	Normalized	0.727	0.925	0.677	0.807	0.577	0.853	0.450	0.715
W(SVM)		0.818	0.825	0.751	0.882	0.595	0.863	0.480	0.724
W(NB)		0.636	0.775	0.407	0.693	0.577	0.833	0.425	0.705
W(DT)		0.682	0.900	0.604	0.752	0.591	0.858	0.468	0.724
F(SVM)		0.773	0.875	0.648	0.866	0.559	0.840	0.416	0.705
F(NB)		0.636	0.900	0.565	0.785	0.536	0.880	0.451	0.708
F(DT)		0.591	0.950	0.604	0.770	0.614	0.883	0.524	0.748
CFS	Discretized	0.773	0.775	0.531	0.778	0.505	0.885	0.427	0.695
W(SVM)		0.636	0.875	0.531	0.711	0.368	0.885	0.304	0.627
W(NB)		0.727	0.875	0.609	0.764	0.305	0.920	0.285	0.613
W(DT)		0.818	0.900	0.718	0.827	0.455	0.918	0.433	0.686
F(SVM)		0.773	0.875	0.648	0.810	0.418	0.863	0.322	0.640
F(NB)		0.864	0.875	0.726	0.899	0.391	0.885	0.333	0.638
F(DT)		0.591	0.875	0.491	0.727	0.391	0.863	0.293	0.627

Chapter 7
Final Remarks for the Research With Advanced Machine Learning Methods in Colon Cancer Analysis

ABSTRACT

Generally, classification accuracy is very important to gene processing and selection and cancer classification. It is needed to achieve better cancer treatments and improve medical drug assignments. However, the time complexity analysis will enhance the application's significance. To answer the research questions in Chapter 1, several case studies have been implemented (see Chapters 4 and 5), each was essential to sustain the methodologies discussed in Chapter 3. The study used a colon-cancer dataset comprising 2000 genes. The best search algorithm, GA, showed high performance with a good efficient time complexity. However, both DTs and SVMs showed the best classification contribution with reference to performance accuracy and time efficiency. However, it is difficult to apply a completely fair comparative study because existing algorithms and methods were tested by different authors to reflect the effectiveness and powerful of their own methods.

1. INTRODUCTION

Generally, classification accuracy is very important to gene processing and selection and cancer classification. It is needed to achieve better cancer treatments and improve medical drug assignments. However, the time complexity analysis will enhance the applications' significance. To answer the research questions in Chapter 1, several case studies have been implemented (see Chapters 4 & 5), each was essential to sustain the methodologies discussed in Chapter 3. The study was used a colon-cancer dataset comprising 2000 genes.

DOI: 10.4018/978-1-7998-7316-7.ch007

The best search algorithm, GA, showed high performance with a good efficient time complexity. However, both DTs and SVMs showed the best classification contribution with reference to performance accuracy and time efficiency. However, it is difficult to apply a completely fair comparative study, because existing algorithms and methods were tested by different authors to reflect the effectiveness and powerful of their own methods.

2. CONTRIBUTIONS

The main contributions to the efficient algorithms for genes/feature search and classification algorithms shown as follows:

- Demonstrating the main algorithms/ methods applied for better colon cancer selection and classification, then proposing a novel inventive way of examining both the accuracy and efficiency using temporal analysis of the high performance of genes/feature selection and cancer classification algorithms for the colon cancer. This proposed model is being implemented as an alternative measure to compare between the ML algorithms instead of using the classification accuracy on its own for evaluating the ML algorithms.
- Developing a new model based on a two-stage hybrid and multifilter genes/ features selection to enhance the colon cancer classification accuracy. It is based on the process of reducing the number of features, eliminate noise, and select those genes/ features which are informative and related. This model proves to produce better results compared with other studies. These contributions were demonstrated in the following case studies.

2.1. Case Study One

The objectives of this case study were accomplished by examining the performance of the most common gene selection and cancer classification algorithms for the colon dataset. The novel incentive was to incorporate a comparison between the classification accuracy and the time complexity analysis and to demonstrate which algorithm is most ideal for efficient performance. This work employed three relevant and typical gene selection algorithms: GA, PSO and IG. This work tested the results by means of four related ML classifiers: SVM, NB, DT and GP. A 3-phase experiment was followed.

- Phase one examined the variance between several selection algorithms, showing that the GA algorithm was better than PSO when employed to select a feature subset that minimised the population size with related and informative data, which could be used as an input for the classification algorithm thereafter.
- In Phase two, classification algorithms were applied independently without integrating any selection algorithm. DT was found to have the best classification accuracy with a suitable time efficiency of big $O(n^2)$, compared to other ML classifiers.
- Phase three integrated selection and classification algorithms and compared their outcomes. The algorithm that outperformed others and expressed the highest performance with efficient time complexity was the integration of GA and DT (92%).

Generally, discretised data pre-processing showed better performance results than normalised data pre-processing, in terms of time efficiency, with better classification performance accuracy. The experiment results showed that the application or employment of feature selection methods, prior to classification algorithms, resulted in better accuracy than if the classifiers were applied independently. Without feature selection, the use of DT as a classifier returned 87% accuracy when the data was discretised. SVM, however, returned 86% when the data was normalised, compared to when the classifier was incorporated with a feature selection algorithm: GA/DT (92%) and PSO/SVM (89%). Furthermore, the wrapper methods presented more accurate results compared to filter models using the full microarray dataset. Thus, GA/DT is recommended as the most appropriate algorithm for colon cancer selection and classification in medical research. Moreover, DT showed a polynomial (quadratic) growth rate of big $O(n^2)$, whereas SVM showed a polynomial (cubic) growth rate of $O(n^3)$. Thus, DT is a more efficient classification algorithm for time complexity and has a higher predictive classification accuracy result as a classifier.

2.2. Case Study Two

This study achieved proposing a new model solution to the problem of colon cancer identification, leveraging the application of a two-stage hybrid multifilter selection method to enhance accuracy. The most significant advantage of the method proposed is its significant reduction of features, which results in better classification performance for colon cancer prediction. The experiments of this case study were conducted in two phases with the following achievements:

- Phase one applied a hybrid feature selection algorithm (GA+IG) that dramatically reduced the number of features to almost 97% of the original dataset. We found that SVM and DT had a better accuracy (89%) at this phase.
- Phase two showed that the methods not only decreases noise, removed irrelevant genes and reduced the number of genes to 15 genes out of 2000 original genes, but it also assisted ML algorithms with producing better accuracy rates, obtaining efficient results. SVM achieved the highest classification accuracy (94%) using the proposed system of two-stage hybrid feature selection.

The results proved that applying the proposed two-stage multifilter feature selection methods prior to ML classification results in much better accuracy. Moreover, the hybrid techniques played a significant impact in reducing features and selecting more informative genes.

This research plays a significant role towards a positive impact for treating colon cancer. The main target of the research was to investigate the efficient algorithms toward colon cancer gene selection and tumour classification. Thus, to find out the fastest algorithm(s) which take less time to do the cancer classification task and in parallel to provide high classification accuracy. This research has a positive impact to the society because it assists the early and fast colon cancer detection; which will possibly increase the chance of life, will provide successful better treatment, and will improve the quality of life. Also, it will allow for more treatment options and better chances for cure. The research is of importance for healthcare systems, and provides a kind of excellent health care to patients suffering from this disease through fast, valid, and nearly high accurate classification results. On the other hand, this research provides a very good feedback to physicians and researchers in the medical field to better understand and study cancer tumours utilising machine learning algorithms. Another equally important impact reflects the financial cost savings for governments toward providing treatment health services and also for patients who will notice a significant lower treatment cost. In summary fast and accurate cancer detection and classification is a priority for governments to enhance quality of human being life.

3. FUTURE WORK

The foundational results of this research provide several paths for future research.

- More efforts can be carried out to access the availability of other colon-cancer medical records.
- More ML tools and simulators can be employed in a comparison study using results obtained from the Weka learning tool.
- The current study can be extended to employ the gene selection and cancer classification algorithms with proven good results into the study of other datasets for different cancer types.
- An investigation about the parameters for each algorithm can commissioned to discover which parameter better affects the classification accuracy and efficient results.
- Classification algorithms can be improved by accessing and adjusting parameters.
- Other data pre-processing techniques can be investigated.
- A user interface for classifying colon cancer dataset can be developed.
- Image analysis algorithms can be integrated into study and classification of the colon cancer dataset.

Chapter 8
Importance of Research Into Big Data With Machine Learning Approach

ABSTRACT

This chapter provides some background information, highlights the motivations and problems resolved, and then discusses the aims and contributions of the research conducted. Over the past decades, there has been an explosion in the volume, variety, and velocity of data. Offering effective solutions as a resolution of some major problems this explosion brings has become ever more important. One such solution is big data machine learning (ML) classification or clustering. However, with the solutions offered, we are faced with several problems that include but are not limited to the context.

1. INTRODUCTION

Over the past decades, there has been an explosion in the volume, variety and velocity of data. Offering effective solutions as a resolution of some major problems this explosion brings has become ever more important. One of such solutions is big data machine learning (ML) classification or clustering. However, with the solutions offered we become faced with several problems that include but not limited to the following:

1. Varying domains: A classifier trained using a labelled dataset may not be suitable for another dataset.
2. Traditional methods cannot efficiently accommodate the large varieties of class types found in a dynamically growing dataset. This often leads to inaccurate classification results.
3. Traditional methods are not suitable for present day multiple learning or multi-varying data tasks (Suthaharan, 2014).

DOI: 10.4018/978-1-7998-7316-7.ch008

Data classification is a data mining process of allocating data into one or more categories. The original and traditional concepts of classification involves a process of allocating pre-labelled data input into their relevant category, deriving a classification function and then applying this function to correctly predict the class/category of un-labelled data input.

One of the most basic ways for organizations to determine the relative importance of the data they possess is through data classification. An interview of three chief information security officers (CISOs), from different organizations (Microsoft, Royal Bank of Scotland and dell incorporations) by Microsoft trustworthy computing in (Computing, 2014), confirms the relative importance of data classification in today's information security scenery.

The data many organizations must deal with in recent years is referred to as big data; hence it is important to reason data classification in terms of big data. Big data is a term usually defined in terms of *Volume*, *Variety* and *Velocity* (3 Vs). Definitions and discussions on big data can be found in (Chen, Mao, & Liu, 2014; Fan & Bifet, 2013; Mahmood & Afzal, 2013; Small, 2013). There are numerous benefits of big data, which have been discussed over the years in different literatures, some of them include: increased efficiency, better and improved services in different sectors e.g. healthcare, e-commerce, etc.

In the literatures, the classification problem is mostly communicated as follows. Given a set of class labels (Aggarwal, n.d.) and a random variable input **X** under consideration, determine correctly which label should be assigned to a new unlabelled instance of **X** (Aggarwal, 2014b). Clustering differs from classification in that it uses similarities between feature variables to perform separation into groups without prior understanding of the group's structure (i.e. it uses unlabelled dataset) (Jain, Murty et al. 1999, Aggarwal and Reddy 2013, Jacques and Preda 2014). While for classification, the separation is done based on training dataset that translates information concerning the construction of the clusters (i.e. it uses labelled data) (Sokal 1974, Aggarwal 2014a, Fabrico 2014). Classification is regularly denoted as supervised learning whereas clustering is often denoted as unsupervised Learning. Classification of big data has several advantages and benefits, some of which can be seen in Appendix 1.

There are several conventional tools for data classification, and one of such tools is waikato environment for knowledge analysis (Weka) (Hall, Frank et al. 2009). It is a data-mining tool designed mainly for research purposes. It contains a lot of support that allows for data mining tasks easily and can help assist in the development of new ML schemes or systems. The Weka API (application programming interface) provides various methods and function to help us build customisable ML systems.

ML is the field under which data classification resides. There is also no doubt that in data science, ML plays a very key and vital role in building smart and intelligent solutions using big data. From building an understanding of the most widely used ML schemes and algorithms, it has been observed that there are a lot of ML algorithms out there, and a model trained on one dataset might not be useful on another dataset. Also, data scientists spend an awful amount of time searching and selecting the best ML algorithm to use for a given data problem, which in turn brings about the need and growth in the automated ML (autoML) field. The autoML field is a fast-growing ML area, designed to automate tasks of data preparation, pre-processing, and model training to ease the tasks of both intermediate and experts in the field.

Although there are a lot of traditional data classifiers or clusterers, classification techniques and tools that can be used to achieve data classification, a majority still lack in their ability to effectively address the major challenges of big data on the fly. For example, some are not very effective in handling heterogeneous multi-datasets, or for handling large data streams. Secondly, some of the traditional classification methods are not flexible and scalable enough to handle large datasets or changes for which they were not trained to handle. A highly acceptable classification method or tool should be able to address the three major challenges of big data, should be flexible enough to adapt to changes within the organization. Lastly, another limitation of many classification systems, is the time and tedious process spent in finding the best ML algorithm to use on multi-varying datasets in a timely fashion. In ML, the decision about what learning algorithm to use, has been incorporated into the meta-learning (Learning to learn) research. Meta-learning has proven to have a major correlation with classification tasks. This connection is because as a researcher designing a classification system, one must empirically and analytically study existing algorithms (tons of algorithms exists) and in some cases even make use of some base concepts or hypothesis while designing the systems.

2. RESEARCH FOCUS AND VALUES

In effectively designing a classification system the first step after defining what the achievable goal is, usually entails the process of deciding what ML approach or model to select. Although some autoML systems (e.g. autoWeka and auto-sklearn) discussed in chapter 2, are efficient in their own ways for model selection, some limitations they still have include:

a. **Auto Learning mode as well as model selection**: Not considering and using more generic information and knowledge about various learning schemes (supervised, unsupervised or semi supervised) and algorithms (e.g. what happens if for a given scenario only a small amount of labelled training data instances is available?) to automatically decide on the mode or model algorithm to use of any given dataset.
b. **Complexity** of the various autoML systems, caused mainly by focussing heavily on the problem of hyper parameter search and selection.
c. **Supplying multi-varying datasets:** Inability to supply multiple datasets from different domains and sources at once to the tool for processing. This is mainly since because the systems are complex and consider not just the algorithm space but also the hyper parameter space and other parameters such as resource budget, etc. they need to consider only one dataset at a time.

These limitations listed above, form part of the problems and motivations for undergoing this research. The importance of studying, understanding, designing, conceptualizing, and analyzing various ML algorithms to develop autoML systems for big data is well-accepted in many application areas. The concept of meta-learning with hybrid autoML can play an important role with regards to the representations of such a system. However, there is a scarcity of research on the assessment of the practical usefulness of the new representations for automatic mode as well as model selection on single or multi-varying datasets. To achieve such an assessment, effective formal support including the use of more generic knowledge about various ML methods and formal verification are required, and appropriate tools facilitating the automatic selection and analysis of an appropriate algorithm for big data ML tasks are necessary.

The contributions help to create a simple and less time-consuming hybrid-autoML system, which is beneficial in the sub field of autoML, and the data science and ML research community at large.

3. AIMS AND CONTRIBUTIONS

Our research hypothesis is that the hybrid big data autoML model designed, supported by an appropriate toolkit can deliver an effective approach to automatically determine the best ML mode and model that can yield the best accuracy, given a heterogeneously large dataset, limited resources (i.e. limited time) and knowledge about various ML methods. To validate this hypothesis, a generic methodology involving both theory and practical research is employed. The main aims of the study are as follows:

Aim-1, Theory: To provide a formal foundation for hybrid autoML concepts, involving the extension of current formalization and proofs of several results concerning model selection which hence govern the correct use, manipulation and analysis of various types of autoML abstraction.

Aim-2, Toolkit: To develop a platform for uploading single or multi-varying datasets and provide automatic decision learning on the ML mode to use, dedicated auto ML model selection, training, prediction and analysis for all the datasets on the fly.

Aim-3, Evaluation: To assess the utility of hybrid-autoML models on practical use-cases faced by experts in the field.

With regards to Aim-1, we propose several additional properties of basic autoML structure incorporating meta-learning. We provide new learning execution semantics for varying multi dataset variants, and we design algorithmic functions for automatic clustering (an ‘autoProb’ function), automatic classification model selection (a generic rule based ‘model selection’ function) and the simulation of conventional ML for multi datasets. We extend the existing basic autoML model concept for Auto-Weka (Kotthoff, Thornton, Hoos, Hutter, & Leyton-Brown, 2017) to formally support alternative representations of a given behavior based on some ideas in meta-learning research and in auto-sklearn (Feurer et al., 2015).

The new structure allows one to model multiple alternative scenarios that can occur in practice. We also extend basic meta-learning algorithms to support new generic knowledge representations. We formally describe how the hybrid autoML model saves time in the first instance for any user of the toolkit, by pointing them to what ML algorithm they can start exploring.

We present a novel automatic clustering selection algorithm, that can take a decision to choose between existing clustering algorithms in Weka or use an autoProb clustering function designed based on varying distance/similarity measures e.g. Euclidean distance. We investigate the unfolding of a less complex solution that isn’t primarily focused on considering the set of the hyper parameter space, but simply on using general knowledge about different learning schemes and more generic features of the data to learn and automatically build models on various datasets from different domain sources. Such an unfolding contains a representation of all the possible running processes. We provide an algorithm for the construction of the unfolding.

In pursuit of Aim-2, we develop ‘Hybrid-autoML’, which is an open source tool for automatic learning mode, model selection, and model analysis. The tool is implemented as a Java based application or command line (CLI) platform which provides a flexible and extensible framework for the development and analysis of simple conventional auto ML for multi-datasets. Hybrid-autoML provides a user-friendly graphical interface that facilitates single or multi-datasets entry, supports visual simulation of various ML scenarios (e.g. presence of large labelled training data with little unlabeled test data, small unlabeled data with no specific training dataset, large unlabeled data with no training data, etc.), facilitates predictions, and integrates a set of analysis tools from Weka application programming interface (API). More specifically, for automatic model learning we implement the essential functionalities for their creation and visualization on multi-datasets, as well as facilities for their simulation, error analysis, performance verification and evaluation. We implement rule-based algorithms for visualizing dataset properties, choosing target features for model build consistency and estimating missing data information.

With regards to Aim-3, we apply 'Hybrid-AutoML' to five different practical ML scenarios related to big datasets to assess the practicality of the model.

REFERENCES

Aggarwal, C. C. (2014a). *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC. doi:10.1201/b17320

Aggarwal, C. C., & Reddy, C. K. (2013). *Data Clustering: Algorithms and Applications* (Vol. 31). Chapman and Hall/CRC. doi:10.1201/b15410

Aggarwal. (2014b). *The setwise stream classification problem*. Academic Press.

Aggarwal. (n.d.). *Instance-Based Learning: A Survey*. Academic Press.

Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, *19*(2), 171–209. doi:10.100711036-013-0489-0

Fabrico, L. (2014). *Data Mining Classification*. Academic Press.

Fan, W., & Bifet, A. (2013). Mining big data: Current status, and forecast to the future. *SIGKDD Explorations*, *14*(2), 1–5. doi:10.1145/2481244.2481246

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, *11*(1), 10–18. doi:10.1145/1656274.1656278

Jacques, J., & Preda, C. (2014). Functional data clustering: A survey. *Advances in Data Analysis and Classification*, *8*(3), 231–255. doi:10.100711634-013-0158-y

Jain, A., Murty, M., & Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*(3), 264–323. doi:10.1145/331499.331504

Mahmood, T., & Afzal, U. (2013). *Security Analytics: Big Data Analytics for cybersecurity: A review of trends, techniques and tools*. Paper presented at the 2013 2nd National Conference on Information Assurance (NCIA).

Small, M. (2013). Securing Big Data. *ITNOW*, *55*(3), 10–11. doi:10.1093/itnow/bwt036

Sokal, R. R. (1974). Classification: Purposes, principles, progress, prospects. *Science*, *185*(4157), 1115–1123. doi:10.1126cience.185.4157.1115 PMID:17835456

Chapter 9 Overview of Big Data With Machine Learning Approach

ABSTRACT

This chapter provides discussions on what is already known in the area of this research, touching particularly on the key concepts, theories, and factors and how they are relevant to this research. Some inconsistencies, limitations, and problem in existing literature are discussed. Discussions on why some of these limitations and inconsistencies occur, how the knowledge relates to this research, as well as issues still yet to study effectively are carried out. Finally, it sets the basis for what contributions this research makes and who will benefit from such a study.

1. BIG DATA MACHINE LEARNING

Data science is a science used to tackle big data and comprises of data cleansing, preparation and data analysis. Big data is a term usually discussed in terms of *Volume*, *Variety* and *Velocity* (3 Vs). Definitions and discussions on big data can be found in (Chen, Mao, & Liu, 2014; Fan & Bifet, 2013; Mahmood & Afzal, 2013; Small, 2013). There are numerous benefits of big data, which have been discussed over the years in different literatures, some of which include: increased efficiency, better and improved services in different sectors e.g. healthcare, e-commerce, security etc. Datasets from multiple sources are gathered and then machine learning, predictive analytics and sentiment analysis are used to extract vital information from the collected datasets. The field of data science acts as an umbrella under which data mining, data analytics, machine learning and various other related subject areas are included.

Machine Learning (ML) as one of the subject areas in the field of data science is described as the act of applying algorithms to data, in order to learn from it and then predict future trends in any topic or domain area such as the health domain. It focuses mainly on the application of algorithms and statistics to the data as opposed to data science which is the term used when referring to the whole data processing practise. Machine Learning comprises supervised learning (data classification) and unsupervised learning (data clustering) schemes. The characteristic of big data brings about new challenges and op-

DOI: 10.4018/978-1-7998-7316-7.ch009

portunities for classification algorithms, giving rise to a new era of classification algorithms that will be able to address and handle the challenges of *velocity*, *variety* and *volume* that comes with big data. One of which is proposed in this research.

1.1. Big Data Classification Related Works

The challenges that big data characteristics bring have led to new trends of classification algorithms to help address the challenges for effective data classification of big data. A lot of literatures are available on classification algorithms which is useful for big data classification. However, this section will focus on discussions of literatures that employ classification algorithms to address the velocity, variety & volume challenges of big data. Secondly, discussions on literatures that employ auto classification algorithms are carried out. Finally, Literatures that use semi-supervised classification techniques are discussed.

To address the *velocity* challenge of big data, 'online streaming classification algorithms' are being proposed and developed, while for addressing the challenges of *variety,* 'heterogeneous machine learning' or 'multi-view classification for data heterogeneity' algorithms are designed. For addressing the challenges of *volume*, much efforts are being made to scale up existing classification algorithms. Some algorithms however address either one or two of these challenges. Nevertheless, it is seemingly difficult to see an evolving, automatic, semi-supervised, hybrid probabilistic big data classification algorithm that can address the three challenges at the same time and in a simple and effective manner, like the one being proposed in this research.

A survey of stream classification algorithms is conferred in (Charu C Aggarwal, 2014c). In 2005, the authors in (Law & Zaniolo, 2005), proposed an adaptive nearest neighbour classification algorithm (ANNCAD) for data streams.

In more recent times however, (Bertini & Zhao, 2013) present a graph-based algorithm to discourse the problem of moderately labelled streaming data. Their algorithm extends a semi-supervised K-associated optimal graph algorithm (KAOGSS) and a purity measure transductive algorithm (PMTLA), which is also a graph-based model. The accuracy and processing time of the algorithms extended, where tested with real and artificial streams of data and the results compared. This differs from the proposed algorithm in this research in the sense that the proposed algorithm in this incorporates concepts from an unsupervised probabilistic Bayesian classification method called autoClass (eCheesman, Self, Kelly, & Stutz, 1996) and concepts from supervised rule-based methods.

Another interesting work is presented in (Sheikholesalmi, Mardani, & Giannakis, 2014), for the classification of streaming incomplete big data sets. A systematic model suitable for streaming big data, which makes use of the core low-dimensionality of feature vectors to design an SVM classifier that can handle relevant feature misses, is discussed. It is developed on the instinct that errors can be added using the core low-dimensionality of feature vectors, likewise the basic comparisons amongst data instances of similar class. Stochastic alternating minimization is used to design an online solution that renders the proposed approach operative for big scale dataset with probably numerous features. Computational challenges where mitigated by developing a first order 'stochastic sub-gradient descent (SGD)' structure for classifier update. However, their proposed design is quite a complex classifier for online streaming data.

The identification stage of the two stage, real time fault detection and identification system proposed in (Costa, Angelov, & Guedes, 2015) shows promising applicability to on-line streaming uses. The first stage in their approach is the *fault detection*, which is founded on the notion of the density in the data space for detection and measure of abnormalities. The second stage is the *identification/classification*, which is founded on a self-evolving fuzzy rule based (FRB) classifier system called the 'AutoClass'. It is a fully unsupervised rule-based classifier, where the learning phase starts from scratch with no need for pre-specified parameters (e.g. the fuzzy rules or the number of classes). The number of classes grows on its own with new class labels added automatically when there is a detection of considerable abnormalities. The autoClass can easily evolve an existing initial rule base. The autoClass works with the concept of data clouds and the structure follows the idea of an AnYa FRB (Angelov & Yager, 2012) classifier. A 'zone of influence' user definition is the starting point of the autoClass Algorithm. The rule base is completely empty at the start (i.e. there is no predefined rule, class label, number of steps, etc.), it is only after construing the first data instance, a data cloud class *nc* is created and a corresponding class label $class^{l}$ added (this completes the first inference rule). For subsequent iterations, autoClass works with the existing FRB, updating the current rules and adding new ones when needed. New classes are formed over time and a certain number of closely related abnormalities are grouped together to create a new cloud class. The autoClass classifier developed by the authors is similar in a way to the one designed in this research in the sense that it is an autonomous and self-evolving classifier, where a new class is created if one doesn't already exist for an incoming dataset. However, the one described in the literature is a fully unsupervised fuzzy rule-based classifier that depends on a previous fault detection stage that uses the concept of density (Recursive Density Estimation) in the data space to determine all possible faults (this concept of density used is not the same as probability density function). Secondly, the autoClass algorithm begins with a definition of an initial 'Zone of influence' by the user. Lastly, even though the autoClass classifier looks promising for resolving the *velocity* (i.e. online streaming capability) and *volume* (i.e. it is scalable) challenges of big data, it does not fully address or provide suggestions for resolving the *variety* challenges also brought about by big data. However, the classification system developed in this is a system that combines both supervised and un-supervised learning models, employing also the concept of evolving and automatic classification, as well as a hybrid classification method that combines various traditional classification algorithms such as Naive Bayes (Probabilistic) and Rule-based technique, which will help address the challenges of *velocity, variety and volume* that big data classification is faced with.

A similar fuzzy rule-based classification system to handle imbalanced big data is proposed in (Krawczyk, Stefanowski, & Wozniak, 2015), the authors aimed to get a system that is capable of handling imbalanced big data with good accuracy and no increase in the run time. They make use of the MapReduce Framework to deal with big data as well as considered the implementation of cost-sensitive learning. However, their intentions, the algorithms did not pass the scalability test for use with big data and the overall performance was poor.

Another similar evolving rule base classifier as described in (Costa et al., 2015) is the parsimonious classifier (pClass) proposed in (Pratama, Anavatti, Joo, & Lughofer, 2015). It applies a fully unsupervised method to drive its learning engine from scratch and can be easily used with online streaming instances.

In (Tekin & van der Schaar, 2013), the authors introduced a distributed online learning framework for the classification of big data from different data sources. The data is treated by a set of heterogeneous distributed classifiers. The classifiers operate in a discrete time setting where various events such as: a data stream with a specific context arriving to each classifier, each classifier makes use of its own classification function or other classifiers to create a label, etc. The authors assume the creation of a binary label. Probabilistic classifiers such as the naive Bayes classifier were among the set of classifiers used. However, the results of their experiments from running two different simulations on network security data failed to pass performance test based on classifier's accuracy.

In (Achcar et al., 2009), a system (AutoClass@IJM) for Bayesian classification of varying data in biology is developed. This system was made with a web interface to AutoClass, a prevailing unsupervised Bayesian classification scheme (Cheeseman et al., 1996) that forms part of the basic idea employed in this research. The AutoClas@IJM however, required a lot of human efforts e.g. preparation of the input data, sending the data files, providing an email address where the URL to the results is sent. It is also not very scalable to use with very large data sets, due to the return time involved.

A similar consideration of AutoClass is seen in (Pizzuti & Talia, 2003), where a parallel version of autoClass algorithm (P-AutoClass) is performed on distributed memory multi-computers. The algorithm divides the classification task among the processors of a parallel machine. This method of parallelization is meant to increase the speed at which classification results are obtained. P-autoClass is also intended for scalability in mining large data sets. Both a theoretical and experimental performance model of the algorithm is carried out. Which the authors use to prove that parallel processing of a classification process (especially if performed on multiple processors) speeds up the classification task. Therefore, making parallel implementation of classification or clustering algorithms very attractive when dealing with big data.

2. CLASSIFICATION AND CLUSTERING

Machine Learning algorithms can be divided into mainly two broad categories, namely classification and clustering. These are discussed in the following sections below.

2.1. Data Classification and Regression

Data classification (sometimes referred to as supervised learning) is a data mining process of allocating data into one or more categories. Traditional concepts of classification involve a process deriving a classification model from pre-labelled data instances and then applying this model to correctly predict the class label of un-labelled data instances in each dataset. Regression on the other hand, is data classification that focuses on predicting a quantity as opposed to a class label. A data instance can be classified into one of two or more classes. When two classes are involved it is often referred to as binary classification model, while when there are more than two classes it is referred to as multi-class classification. We refer to a classification model which has several classes assigned to a data instance as multi labelled. Some traditional classification methods are not flexible & scalable enough to handle large datasets or changes for which they were not previously trained to handle. Also, data scientists and machine learning experts tend to spend a huge amount of time deciding on which machine learning scheme and algorithm to select for a given dataset. Which is due to an enormous amount of supervised classification algorithms and a lack of more generic and robust automated machine learning systems in place to help them achieve this goal.

Traditional data classification algorithms normally comprise of two phases:

2.1.1. Training Phase

This is where a model is constructed from the pre-labelled training instances. However, there are some classification methods where the training phase may be replaced with a pre-processing phase instead.

For example: nearest neighbour classifiers (Yunck, 1976), auto classifiers (Cheeseman et al., 1996) etc. It has been observed from state of the art ML systems, that to obtain good classification results often requires a large labelled training dataset, which is not always available to the users.

In this phase, the function derived from the Training Phase is applied to a new unlabelled data instance, and a label (in a classifier) or quantity (in a regressor) is generated for that instance. However, it is important to note that the classification process itself usually comprises of more phases. For example, the classification process may usually start with a data mining task such as feature/attribute selection (which may consist of a pre-processing or filtering phase to remove irrelevant features and ensure that the data is in the right format needed).

A classification algorithm outcome may be represented for a test instance in either two ways:

- A Discrete label.
- A Numerical score which can be changed to a discrete label.

2.2. Data Clustering

Clustering differs from *classification* in that it uses similarities between feature variables to perform separation into groups without prior understanding of the group structure (i.e. it uses unlabelled data) (Aggarwal & Reddy, 2013; Jacques & Preda, 2014; A. Jain, Murty, & Flynn, 1999). While for classification, the separation is done based on a training data set that translates information about the structure of the groups (i.e. it uses labelled data) (Charu C. Aggarwal, 2014; Fabrico, 2014; Sokal, 1974). Clustering is referred to as *unsupervised* Learning. In recent decades however, a hybrid category emerged with the attention of the masses, which is referred to as the ***semi-supervised*** learning (Sinha, 2014). It is a combination of both the supervised and unsupervised methods thus allowing the use of both labelled and unlabelled data for learning the class label of a new data input. It is a very promising method to use when dealing with the classification of big data, because it can handle the classification process effectively with only a small number of labelled instances and a large set of unlabelled instances (which is usually the case with big data). Semi-supervised method helps bridge the cost overhead limitation (having to label a large set of data, can be very costly) of the pre-labelling process in supervised methods, and the limitation of the unknown (which increases the error rate) in the unsupervised methods.

The usefulness of class labels e.g. intrusion activity may be represented as a class label (supervised event detection), multimedia data analysis, biological data analysis, medical disease diagnosis, etc. are numerous.

Broad categories of data classification include:

- **Technique-centred** e.g. probabilistic, decision trees, rule-based method, neural networks, nearest neighbour, Support Vector Machine (SVM) methods, etc.
- **Data-type centred** e.g. text, multimedia, metadata, time series, sensor data, discrete sequence, network data, big data etc. Different data types may require the design of different methods, with each been quite different. This research is based mainly on the classification of big data type but the classification model designed will be scalable enough to apply on other data-types. Discussions on big data can be found in (Akerkar, 2013; Chen et al., 2014; Fan & Bifet, 2013; Suthaharan, 2014; Tankard, 2012).

- **Classification Analysis Variations:** e.g. semi-supervised learning, transfer learning, active learning, etc. Semi-supervised analysis variation is considered in this research.

2.3. Classification Methods

Before most classification methods are applied to a dataset, a method known as feature selection is often used. Data classification methods often used include:

- Decision trees
- Rule-based methods
- Probabilistic methods
- SVM methods
- Instance-based methods
- Neural networks

These methods along with the feature selection method will be discussed briefly below.

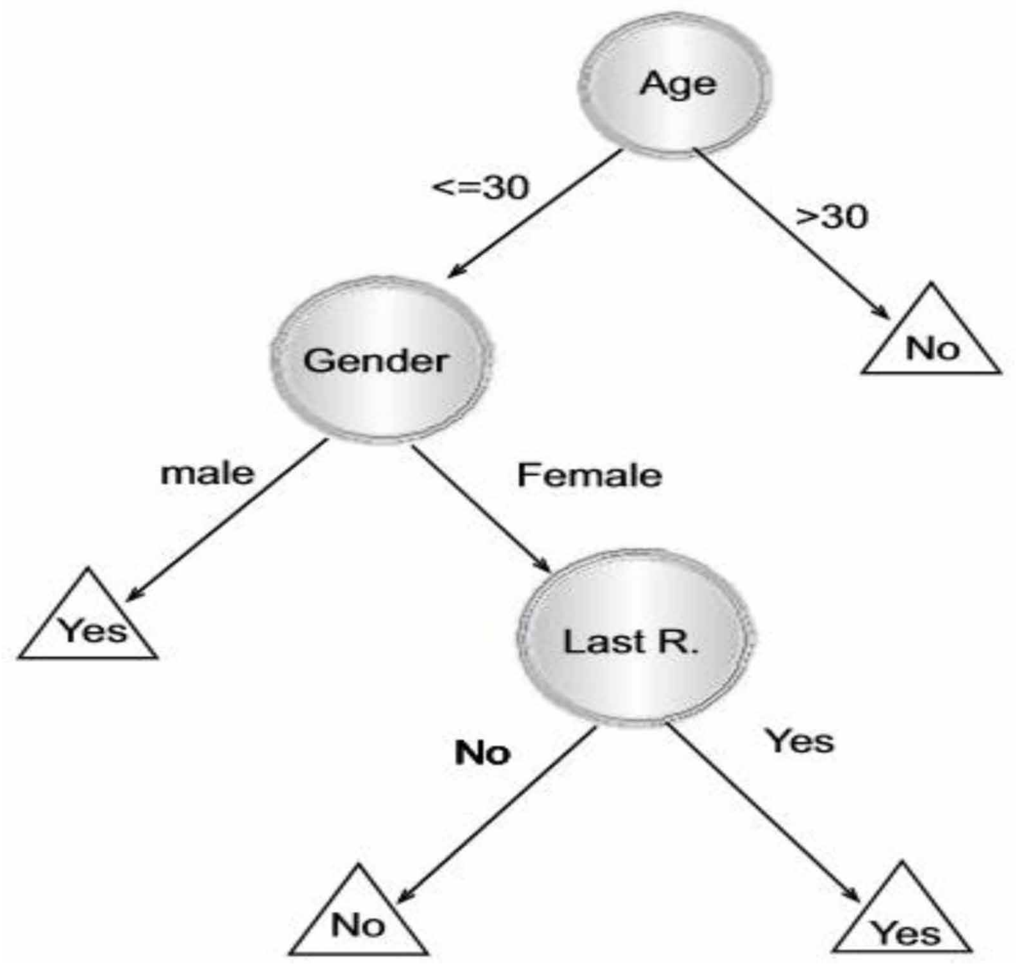
2.3.1. Feature Selection

Feature selection is a method which is usually the first phase of almost all classification tasks. It is critical to use the correct features during the training phase as this will help improve the classification results. However, the use of many features tends to decrease performance of the system. The two most general supervised feature selection methods include: - Filter models (here the technique is independent of the classification algorithm) and Wrapper models (here the process of selecting features is inserted into a classification algorithm and made profound to the classification algorithm, this tactic distinguishes the fact that diverse algorithms may work well with diverse features). When using Filter models, we must be able to measure the significance of a feature to the classification method with some form of evaluation measure. Other feature selection methods include the unsupervised feature selection method (no class label involved), semi-supervised method (which makes use of both labelled and unlabelled data to estimate feature relevance). Feature selection could be done from either flat features, streaming features or structured features. More details on feature selection methods, algorithms and applications is found in (Charu C. Aggarwal, 2014; Alelyani, Tang, et al, 2014; Forman, 2003; Haralick et al., 1973; Jain & Waller, 1978; Kwak & Choi, 2002; Li et al, 2004; Liu et al, 2002; Liu & Motoda, 1998; Liu & Yu, 2005; Mladeni'c & Grobelnik, 1998; Pal & Foody, 2010; Peng, et al, 2005; Punch III et al., 1993; Tang,et al, 2014; Zhao & Liu, 2007)

2.3.1.1. Decision Tree Method

It has a tree-like separation of the data and the various separations at the leaf level are related to the different classes. Separation at each level is done using a *split criterion*. Either univariate split (when a condition is placed on a single attribute) or Multivariate split (when a condition is placed on multiple attributes) technique can be used. A basic decision tree example is seen in Figure 1 below. It shows a scenario which aims to determine the response of prospective customers to direct mailing. The circles represent the internal/decision nodes (labelled with the test attribute) and the triangles represent the leaf node/class label. Moving down the tree progressively from the root to a leaf allows instances to be classified accordingly and predictions made. The split criterion is usually applied on each internal node to determine what the output node is (which could be another internal node or a leaf node (which is usually a class)).

Figure 1. A simple decision tree that represents responses to direct mailing (Rokach & Maimon, 2010)



Decision tree methods are popular and provide human readable rules, but it is important to keep the tree and splits simple enough to ensure that both the understanding of and stability of the tree does not suffer. More details on the decision tree method and algorithms are discussed in (Charu C. Aggarwal, 2014; Esposito, et al, 1997; Lin, et al, 2008; Murthy, 1998; Nielsen, et al, 2009; Quinlan, 1986; Vens, et al, 2008). Two very popular decision tree algorithms are the classification and regression trees (CART) (Breiman, et al, 1984; Loh, 2011) & the C4.5 algorithm (Quinlan, 2014). A decision tree growth is exponential to the number of attributes and distinct values per attribute. Hence for large data sets, it has been a difficult problem finding a practical, globally optimal decision tree solution. Some methods such as *pruning* of a decision tree to reduce the complexity and attributes have been proposed in many literatures such as (Esposito et al., 1997). However, the pruning method limits the accuracy of the classifier at the expense of reducing complexity. Also, the fact that a split criterion is required at each internal node of a decision tree (which has to match the training set appropriately to ensure high accuracy) means that the practicability of applying a split criterion used for a particular data set on another would be a complex and costly task. This also implies therefore that one would require various split criterions or various tree classifiers incorporated together to achieve accurate classification of big data (which will increase the complexity of the model as well as increase the run time). Asides the limitation of having a split criterion at each internal node and the challenge of growing a decision tree without making it too complex, another identified problem with decision trees is that in order to avoid inaccuracies it is hard predicting when to stop the tree growth. In cases where there is a tendency for many classes, unnecessarily large trees may result. Many standard decision tree algorithms such as the CART (Breiman et al., 1984) are deterministic in nature (i.e. if given the same input information, the same output information is produced with only one pre-determined outcome considered), as opposed to the non-deterministic characteristic of the approach proposed in this (i.e. where more than one possible outcome is considered even if give the same input information). Another limitation is that to decide the succeeding split, decision tree induction (i.e. building a decision tree automatically from a given data set) will need to compare all potential splits. Most standard decision tree algorithms are mainly supervised learning methods where it is compulsory to have a set of pre-labelled training data sets from which the tree can be built, and the accuracy is highly dependent on the amount of labelled test instances available. Having a large set of pre-labelled training instances is not the case in the real world, as the process is quite a costly one.

2.3.1.2. Rule-Based Method

Are methods like the decision tree method but differs in the sense that it allows overlaps (i.e. there is no strict hierarchical separation) to create a very robust training model. Some path in a decision tree may be understood as a rule which allocates a test instance to a specific label. For example, from the decision tree in Figure 2.1 above, the rule "if a customer's age is greater than 30, then the customer will not respond to the mail" can be deduced from one of the paths. Rule-based methods have the advantage of being simple, easy to explain and understand, can be easily improved by addition of more rules, etc. Logic forms (e.g. IF-THEN statements) can be used to represent the rules which human beings can easily understand. They can be seen as more general models than decision tree models. For rule-based methods, a set of rules is extracted from the training data in the training phase. Then in the testing phase, the rules which are important to the test instance are determined and the final output is based on a mixture of the class values anticipated by the various rules. Resolution methods should be designed as well, in order to resolve possible rule conflicts on a test instance. For example, a method of prioritizing the rules is a good resolution strategy to avoid conflicts. More in-depth discussions on rule based methods are seen in (Charu C. Aggarwal, 2014; Angelov & Yager, 2012; Jain et al, 2013; Nosofsky & Little, 2010; Pratama et al., 2015; Tung, 2009). Two well-known rule-based classification techniques is the *rule induction* and *association rule-based classification*. In rule induction algorithms, a small set of rules is developed straight from the data. Two fundamental rule induction algorithms in the literature are the CN2 Induction Algorithm (Clark & Niblett, 1989) and RIPPER (Cohen, 1995). In the CN2 algorithm, each rule is learnt without assigning a class for each iteration. While in the RIPPER algorithm, all the rules pertaining to a class is learnt first before the all the rules of the following class is learnt.

RIPPER has been employed mainly for classification of text. To achieve high accuracy, majority of the traditional rule induction algorithms e.g. CN2, RIPPER, etc. frequently contain a lot of conditions, thus making the rules unnecessarily long and hard to work with. Association rule classification proposed in (Ma, 1998) and in (Zhang & Zhang, 2002). It can help in detecting association rules from huge amount of data. Class association rules (CARs) as proposed in (Ma, 1998) is an example. It is required that the output of a CAR be a class label. Rule induction models identify only a subset of the rules needed for classification while classification based on association rule mining detects all the rules in the data. The rule-based methods on their own are quite slow and the rules could be sometimes misleading if proper care is not taken. This is because often the rules in the rule list are dependent of each other. A limitation of using only rule-based method for big data classification is that the quality of a rule may vary between data instances, therefore limiting the accuracy of the results. Also, we will be faced with the challenge of wasting meaningful time in generating a long rule list (as generated from rule-based induction methods) instead of just having basic generalized rules that can be applied on all instances. Or we will be faced with the challenge of detecting all the rules present (as observed with CARs). Though detecting all the association rules of big data will help improve the classification of an input instance correctly, it may however involve a high run time.

2.3.1.3. Probabilistic Methods

These are very common and fundamental amongst data classification methods. They make use of statistical interpretation to find the best class for a given sample. Probabilistic classification algorithms will often output an equivalent *posterior* probability $p(C|x)$ for each of the possible classes a test instance may belong to (Charu C. Aggarwal, 2014).

Posterior probability: conditional probability obtained after considering precise features of the test case.

Prior probability: probability distribution of training records that belongs to each specific class.

The two basic ways that the posterior class probability is estimated:

- Through defining the class conditional probabilities $p(x|C)$ for each class (C), after which the prior class probability $p(C)$ is then inferred and Bayes theorem used to determine $p(C|x)$.
- By modelling the joint distribution $p(x,C)$ directly and then normalizing it to obtain the $p(C|x)$.

We have both ***generative*** probabilistic models (where the joint distribution of inputs and outputs are modelled implicitly or explicitly) and ***discriminative*** probabilistic models (where a discriminative mapping function (equation (2.0)) is learnt and used to model the posterior probabilities directly). A comparison of both generative and discriminative models is discussed in (Jordan, 2002). Examples of the probabilistic *generative* model for classification is the 'Naïve Bayes Classifier' (Murphy, 2006) and the 'Hidden Markov Model'(Blunsom, 2004; Rabiner, 1989).

$$f(x) = p(C|x) \tag{2.0}$$

Simplification of the Bayes model is what leads to the Naive Bayes hypo (John & Langley, 1995). It is not only simple and fast but also commonly applicable. Its aim is to create a rule that will permit assigning imminent instances to a class with an assumption of attributes independence after establishing probabilities (Triguero, García, & Herrera, 2013). Examples of popular probabilistic *discriminative* model is the 'Logistic regression' model and the 'Conditional Random Fields' model.

Logistic Regression model is formally defined as:

$$P\left(Y(T) = i(X) = \frac{1}{1+e^{-\theta TX}}\right) \tag{2.1}$$

(Charu C. Aggarwal, 2014; W. Liu, Liu, Tao, Wang, & Lu, 2015; Tortajada et al., 2015), Where θ is the parameters vector to be measured.

A diversity of other probabilistic models are also known in literature, e.g. probabilistic graphical models (Koller & Friedman, 2009), and conditional random fields et al, 2001). More on probabilistic methods is discussed in (Bishop, 2006) and (Alsallakh, et al, 2014; Azar & El-Said, 2013; Bankert, 1994; Iounousse et al., 2015; Lu et al., 2010; Lukasiewicz, 2008; Maravall, et al, 2013; Murphy, 2012; Nielsen et al., 2009). Some common advantages of the probabilistic models observed in the literature include:

- The fact that each class's associated probability can easily qualify as a value of confidence of the input instance belonging a class.
- They can be easily and successfully incorporated into larger machine learning tasks while partially or totally avoiding the problem of error propagation.

Some limitations of traditional probabilistic model are:

- Majority of the models are deterministic in nature and do not consider other choices such as being able to adjust to change in the middle of model build.
- They are mainly for supervised learning where there is a high dependency on pre-labelled data instances at the learning/training phase. Although, to be adaptable for unsupervised classification or semi-supervised classification, they need enhancement and optimization.
- On their own they cannot effectively handle at the same time all three challenges (i.e. *volume, variety* and *velocity)* that big data brings. However, combining them with other methods (e.g. decision trees, SVM, etc.) and techniques to achieve a relatively high classification performance of big data is useful.

These limitations and many more are part of the reasons that researchers are constantly studying and experimenting on ways to build or enhance these traditional classification methods to handle evolving real-world situations effectively.

2.3.1.4. SVM Method

This classification method may be well-thought-out as a single level decision tree with a very carefully selected multivariate split condition (Charu C. Aggarwal, 2014). It uses linear conditions to separate the classes from one another as much as possible (Cortes & Vapnik, 1995; Li, 2015). Kernel methods (using similarity measures between two objects) are used for general non-linear SVM learning methods (Schölkopf & Smola, 2002). One important criterion for SVM is to achieve maximum margin separation of the hyper planes. An advantage of the kernel methods is its ability to be extended to random data types and its quality of generalization (Leiva-Murillo et al., 2013). A downside to SVM method is that if the numbers of attributes are much more than the numbers of samples, SVM methods are likely to perform poorly. Also, they are slow and do not directly make available probability estimations. The probability estimates are calculated using cross-validation techniques which in practice are quite expensive. A method to optimize the speed of SVM classifiers has been proposed in literatures such as (Fischetti, 2015). But the authors in a bid to optimize the SVM method with Gaussian Kernel, for it to run faster further created a NP hard complex problem. A method to map the SVM outputs into probabilities is also discussed in (Platt, 1999). A survey on SVM methods and applications is observed in (Wang & Pardalos, 2015) while a comparison of SVM methods against other classification and regression methods is seen in (Meyer, et al, 2003). SVM libraries (Chang & Lin, 2011) are also available for users to easily apply SVM method in their application. Another limitation of the SVM method is that it is designed mainly to be applied for a two-class situation, hence to use it for multi-class scenarios; one would have to apply reduction algorithms to reduce the multi-class model into numerous binary problems. This would likely increase the complexity of the model and the run time.

3. TESTING THE PERFORMANCE OF CLASSIFICATION ALGORITHMS

The performance of most classification algorithms is usually determined by a number of parameters or measures such as: accuracy of the output, the integrity of the model, the run time of the model, simplicity in terms of computational cost, etc. The most fundamentally common one being accuracy of the results. There are various methods that have being designed over the years for evaluating the performance of classification systems. Validation methods are usually chosen, after which the classification model is built and then evaluation measures are used to describe how properly the classification performed with regards to other existing models.

Some methods for accuracy validation of a classification process include:

3.1. Hold-Out Method Validation Method

A statistical method that requires the data is split into two segments (one for training the classifier and one for testing the classifier). The training data set is usually larger than the test data set. A disadvantage of this method is that the test is performed on a smaller portion of the data, thus increasing the tendency for false accuracy measurements (Charu C. Aggarwal, 2014).

3.2. Cross Validation Method

To address the problems of the hold out method, a more logical approach to the hold out method eventually got developed. It is known as the cross validation method (Refaeilzadeh, et al, 2009), which involves the data being split equally and the hold-out evaluation method is performed two times by using the training data set from the first iteration as the test data set in the second iteration and vice versa. The simple form of the cross validation is the k-fold cross validation.

3.3. Bootstrap Method

Creates bootstrap dataset by sampling with replacement the original dataset. This bootstrap data set is what is then used to build the classification model which is then applied to the original data used as the test set. The optimistic ensuing presentation of the bootstrap method is improved by applying a factor 0.632 in (Efron & Tibshirani, 1997). A study and comparison of the cross validation method and the bootstrap evaluation method is observed in (Kohavi, 1995). A more detailed explanation of the bootstrap method is given in (Efron & Tibshirani, 1994).

3.4. Confusion Matrix

Since the resulting output of a discrete classifier (e.g. K-nearest neighbours) is usually an actual class label for each situation and that of a probabilistic classifier (e.g. Bayes classifier) is usually a probability function of belonging to a class, it is important to differ between the evaluation methods used for each. However, a more general evaluation measure might be applicable in a situation where the resulting output of a discrete classifier is transformed into a weighted function or when the output of a probability classifier is related to a label.

For discrete classifiers, a confusion matrix is usually used for evaluating accuracy measurements.

Some terminologies derived from a confusion matrix include:

- **True positive (tp)**: correctly classified positive instances e.g. sensitive information correctly classified as sensitive.
- **False positive (fp)**: falsely classified positive instances e.g. insensitive information being classified as sensitive.
- **True negative (tn)**: correctly classified negative instances e.g. insensitive information being classified as insensitive.
- **False negative (fn)**: falsely classified negative instances which are expected as positive e.g. sensitive information being classified as insensitive. This is a situation that we don't want to happen.

3.5. Discrete Classifier Evaluation Measures

Additional well-known evaluation metrics are only defined for binary classifiers but also easy to use for multi class problems. They include the following.

3.5.1. Classification Accuracy (acc)

Accuracy equals the ratio of correctly classified instances OR can be expressed as the summation of the diagonal features in the confusion matrix. A common measure that gives an idea of the overall performance of the classifier, represented as:

$$acc = \frac{tp + tn}{tp + fp + tn + fn} \tag{2.1}$$

3.5.2. Mean Absolute Error (MAE)

Mean absolute error is finding the absolute errors of the dataset by calculating the absolute difference between each observed versus predicted value, find the sum of the differences and then divide that value by the number of errors. Lower values of the MAE are better when analysing the performance and comparing the performance of different classification models. It is represented mathematically as:

$$MAE = \frac{1}{n}\sum\left|X_o - X_p\right| \tag{2.2}$$

Where n = errors count, X_o = the observed value and X_p = the predicted value.

3.5.3. Recall

It is also known as the sensitivity or true positive rate. It compares the number of true positives with the actual number of truly positive cases. It answers the question of "how many relevant items are selected?" Represented mathematically as:

$$Recall = \frac{tp}{tp + fn} \tag{2.3}$$

3.5.4. Precision

It compares the number of the true positives with the number of predicted positive cases. It answers the question of "how many selected items are relevant?"

$$Precision = \frac{tp}{tp + fp} \tag{2.4}$$

3.5.5. Specificity

Also known as **true negative rate**. It compares the correctly classified negative cases with the total number of truly negative cases and represented as follows:

$$Specificity = \frac{tn}{fp + tn} \tag{2.5}$$

3.5.6. Fall-Out

This is also known as the **false positive rate**, and is represented mainly as follows:

$$FallOut = 1 - specificity \tag{2.6}$$

3.5.7. F-Score

F-score (or **F-measure**) can be used to test the performance of a statistical system. It is often referred to as the harmonic mean of precision and sensitivity, and is based on the precision and recall expressed as:

$$FScore = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right) \tag{2.7}$$

Another evaluation measure that can address multi-class problems is discussed in (Ben-David, 2008). It is a measure that compensates for classification that may be due to chance and is based on Cohen's Kappa function. The authors greatly recommend using sensitivity evaluation measures with weighted kappa in situations when the cost of having an error is unknown.

3.5.8. Receiver Operating Characteristics (ROC)

For probabilistic classifiers, the most significant evaluation measures are correlated to the receiver operating characteristics (ROC) analysis (Majnik & Bosnic, 2013). ROC curves are wonderful tools for picturing and analysing the performance of classifiers. They have the advantage of being independent of the class distribution. ROC analysis technique places classifiers in the ROC space. The ROC space is derived by plotting a graph with the true positive rate (tpr) on the vertical (y) axis and the false positive rate (fpr) on the horizontal (x) axis of a graph.

For example: consider 2 different classifier outputs (classifier's A & B) below from 50 positive and 50 negative instances.

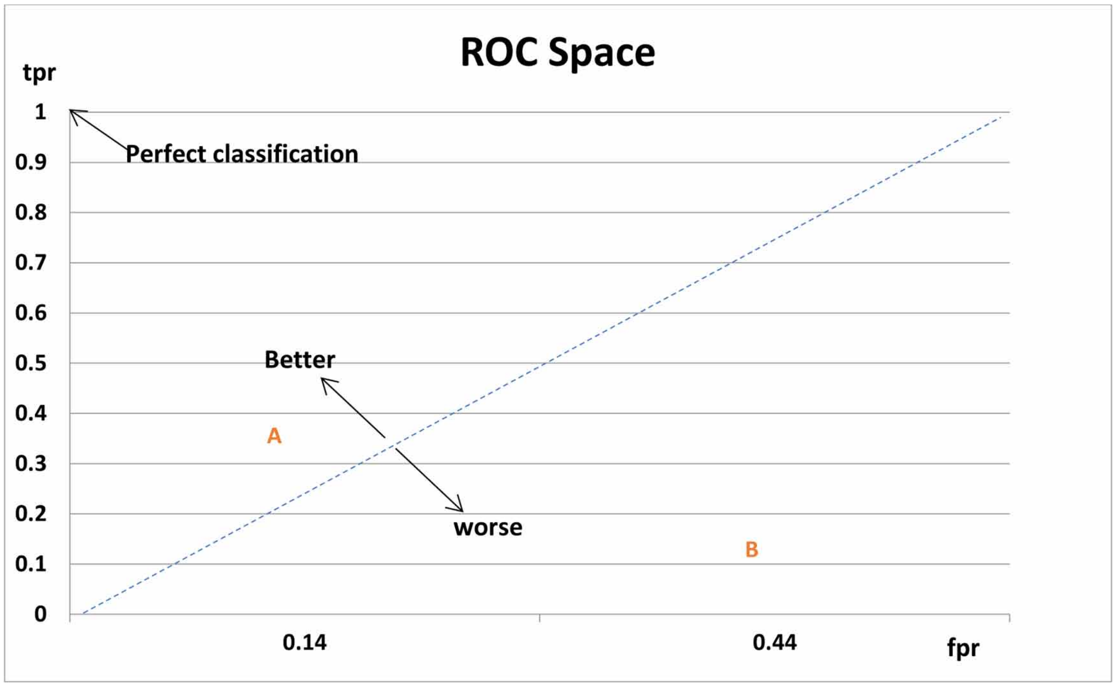
$$A \rightarrow \begin{matrix} t_p = 32 & f_p = 14 \\ f_n = 18 & t_n = 36 \end{matrix} \quad \text{with its tpr = 0.32 \& fpr = 0.14.}$$

$$B \rightarrow \begin{matrix} t_p = 12 & f_p = 44 \\ f_n = 38 & t_n = 6 \end{matrix} \quad \text{with tpr = 0.12 \& fpr = 0.44}$$

From the ROC space as seen in Figure 2.2, we say classifier A performs better than classifier B according to the ROC analysis methodology because it has a higher true positive rate value than B. However, probabilistic classifiers require a threshold to signify the final choice for each class (Charu C. Aggarwal, 2014). Evaluating a large dataset requires more efficient algorithms like the algorithm 24.1 shown in (Charu C. Aggarwal, 2014). Area under the curve (AUC) is a measure that often uses a single value to assess the performance of a classifier. It is the area between a ROC curve and the y axis. In more practical scenarios ROC curves usually expose more information than AUC single value. However, the advantage of using ROC curves in performance analysis. A disadvantage is that it does not measure the complete performance of the classifier but more or less gives us the relative probability ranks. Therefore, the need for effective probabilistic classifier evaluation methods arises. This could be in the form of useful modifications and extensions performed on the ROC methods. There are ROC analysis extensions in literatures e.g. one that is extended for a three class situation is discussed in (Mossman, 1999). Another example of a more recent approach that designed a graphical visualization of the performance of multi-class situations, is seen in (Hassan, et al, 2010). Computational cost is another issue to consider when designing multi-class ROC evaluation measures.

Other measures of the performance of a classifier are discussed as follows:

Figure 2. The ROC space and plots of the two prediction cases above



3.6. Integrity of the Model

Answers the questions “how soundly constructed is the classification algorithm?” or “how stable is the model?” or “what is the consistency in the classifier?”

3.7. Simplicity

The simplicity of a model, shows “how easy it is to understand the model?” or “how uncomplicated the design of the model is?”

3.8. Run Time

The classification model run time, could be discussed from two different viewpoints. It could be viewed in terms of “the time taken to build or train the model” and the “time taken to test the model with new instances”. When building a classifier for big data, run time is important to consider, because it is important to build a high performing classifier in the best time possible. Time measurements during training and testing phases of a classification model will give a more practical evaluation of the run time and not just theoretical.

3.9. Reliability

The reliability of a ML model evaluates "how consistent it is in producing the same results, over and over again?". An example of how one can estimate the reliability of a classification algorithm is discussed in (Gurov, 2013).

3.10. Storage Requirements

Another measure as discussed in (Charu C. Aggarwal, 2014), is to consider the storage requirements of the model.

In comparing classifiers, statistical tests are essential to verify that indeed a new classifier outperforms other existing classifiers. There is the parametric and non-parametric statistical test, pairwise or multiple comparison tests (description is seen in (Charu C. Aggarwal, 2014)), transductive or inductive tests (as carried out in (Triguero et al., 2013)). In (Triguero et al., 2013), an experimental study in semi supervised classification is carried out using the KEEL (Knowledge Extraction based on Evolutionary Learning) software tool (Alcalá et al., 2010).

4. CLASSIFICATION TOOLS

There are severaldata mining tools that incorporate both data classification and clustering algorithms. However, this considers and discusses a few open source tools/applications, written in Java programming language, supports all operating system platforms and permits the use with big data. These include:

- Waikato Environment for Knowledge Analysis (WEKA) (Hall et al., 2009): Open Source tool that was first designed in 1993 at the University of Waikato in New Zealand. Supports many data mining tasks such as: feature selection, pre-processing/filtering, classification, clustering, regression and visualization. It only deals with flat files in ARFF format, even though various formats of file can be imported. Provides access to SQL databases. Has four interfaces: The Explorer, Experimenter, Knowledge Flow & Simple Command line interface. The Explorer is the main interface with tabs: Pre-process, Classify, Cluster, Association Rules, Attribute Selection & Data Visualization tabs. Weka also allows the installation of extension packages, and data can be imported from ARFF, CSV, C4.5, binary, etc. file formats, or it can be read from a URL or SQL database. It has some in built file converters, for example to convert from a csv file format to the arff file format.
- Apache Mahout (Ingersoll, 2009; Owen, et al, 2011): Open source project of Apache Software Foundation. It has some scalable machine learning algorithms. But it does not really focus on many data mining tasks. However, it primarily focuses on collaborative filtering, classification and clustering. It isimplemented in the Apache Hadoop platform and has a math environment to help rethink the scalability of the machine learning algorithms built with it.

- Apache Scalable Advanced Massive Online Analysis (SAMOA) (Morales & Bifet, 2015): Is an open source project of Apache Software Foundation. It is a platform for mining big data streams. It is still at its early stages. It is a distributed Streaming Machine Learning framework that contains a programming abstraction for distributed streaming ML algorithms.
- Massive Online Analysis (MOA) (Bifet, et al, 2010): Is an open source tool, specific for data stream mining with concept drift (unforeseen changes over time, in the quantity to be predicted) and supports bi-directional interaction with Weka. It includes a collection of ML algorithms e.g. classification, regression, clustering, etc. It includes evaluation tools. It can be extended with new mining algorithms, evaluation measures or stream generators. Has one interface with 5 tabs e.g. Classification, Regression, Clustering, Outliers & Concept drift. It has a Command Line Interface as well. It is the most popular data stream mining software.
- KEEL (Alcalá et al., 2010): Open Source tool used for various Knowledge discovery tasks. It pays special attention to the implementation of solutions based on data mining techniques e.g. classification, clustering, etc. It can be extended with new algorithms. Has pre-processing methods incorporated.

A comparison and contrast of the various tools are shown in Table 1 below. The representation of what each column stands for in the table is shown below the table.

Table 1. A comparison of some tools used for data mining experimentations

Tool	Link	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
WEKA	https://www.cs.waikato.ac.nz/ml/Weka/	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Mahout	http://mahout.apache.org/	✗	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
SAMOA	https://samoa.incubator.apache.org/	✗	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
MOA	https://moa.cms.waikato.ac.nz/	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓
KEEL	http://www.keel.es/	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓

A =Open Source, ***B***=Easy Setup and Install, ***C***=Has a Graphical User Interface (GUI) plus the API, ***D***=Used with Big Data, ***E***=Has a Collection of Pre-processing techniques such as filtering, etc. ***F***=Over 100 classification and 50 clustering algorithm, ***G***=Various Evaluation metrics present, ***H***=Visualize results, ***I***=Identify statistical dependencies between groups of attributes, ***J***=Search and Evaluation method for attribute selection, ***K***=Useful Educational and Research purposes/communities, ***L***=Algorithms are applied directly onto a dataset or called from your own code, ***M***=Requires user to Identify and select appropriate algorithm for each dataset or collection of datasets, ***N***=Can be run on Apache Spark, which increases the speed up to 10 times more, ***O***=Easy implementation and Extension capability, ***P***=Allows a complete analysis of new proposed algorithm in comparison to existing ones, ***Q***=Graphical visualisations of the dataset. ***ü*** = Yes and ***û*** = No.

5. THE ALGORITHM SELECTION PROBLEM

Making the right decision about the best learning algorithm(s) to use in designing a classification system is a time consuming, tedious and costly process. In machine learning, the decision about what learning method (*supervised learning/classifier* OR *unsupervised learning/clusterer*) has been incorporated into the meta-learning (*Learning to learn*) research. Meta-learning has proven to have a major correlation with classification tasks.

An interesting fact observed in the design of an effective classification system is that, there is a major distinct connection between the meta-learning paradigm and data mining classification. This connection is because while designing a classification system, one must empirically & analytically study existing algorithms (tons of algorithms exists) and in some cases even make use of some base concepts or hypo. When designing the classification system, the process of deciding what machine learning approach (supervised and unsupervised) to be used in next after defining the goal. There are many trends and knowledge shown over the years about supervised and unsupervised machine learning, which can be formally harnessed in reducing the time spent in taking such decisions.

This research proposes a hybrid classification system architecture that comprises of three different layers. The second layer which is a decision learning level, automates the decision-making process on what learning method to adopt at any point in time, given a heterogeneously large stream of data sets. This decision-making process is a Meta-learning (learning to learn) process. The Weka (Waikato Environment for Knowledge Analysis) tool is used in this research for the experimental study. It is a data-mining tool designed mainly for research purposes and widely accepted in the data mining community. It contains a lot of tools that allows for performing data mining tasks easily and can help assist in the development of new machine learning schemes.

An earlier formal abstraction where the algorithm selection problem is considered is discussed in (Rice, 1975). The author aims to answer the question: "what algorithm is best to use in a particular scenario?" by formalizing four criteria (the problem space **P**, the feature space **F**, the algorithm spaces **A** & the performance space **Y**) and five main steps as a possible solution for the algorithm selection problem. It turns out from observations by the author that s*election mapping* echoes as a single most important part of the algorithm selection problem solution.

Later on in (Aha, 1992), the term 'meta-learning' is coined. In the paper, the author discusses ways in which we can draw more general conclusions from the results of machine learning experiments, to give us a set of rules that unfolds situations in which certain algorithms significantly outdo others based on some needful measures. However similar some concepts are, the meta-learning hypo discussed in this research distinguishes from the above study in the sense that it considers case studies involving both supervised and unsupervised learning and not only supervised learners. The set of Meta rules derived in this paper is as a result of empirical studies carried out to determine situations in which using a supervised learning algorithm might be more beneficial than using an unsupervised algorithm.

The field of Meta-learning and Automated Machine Learning (AutoML) have become very useful tools in solving the algorithm selection problem. This two fields are discussed in the following two sections below.

6. META-LEARNING

There are varying views of meta-learning in literatures. In (Vilalta & Drissi, 2002), the authors provide a survey of different meta-learning views with regards to machine learning. The authors also discuss their own viewpoint of meta-learning from the point of constructing self-adaptive leaners, which gathers its Meta knowledge by analyzing the whole instance and updates the knowledge base according to the characteristics of individual instances. They however point out an important fact, which states that despite the varying views on meta-learning, a constant question: "how can knowledge about learning be exploited to improve the performance of learning algorithms?" remains unchanged. The process of learning to learn involves studying ways to improve learning by discovering, mining, and taking advantage of the *invariant transformations* across multiple domains. *Invariant transformations* gives a more general understanding of the nature of patterns across domains (Vilalta & Drissi, 2002).

We can see also in (Smith-Miles, 2009) a unified framework that is used for analyzing various research developments that aims to tackle the algorithm selection problem as a general learning problem across different domains.

Some literatures refer to meta-learning algorithms as one in which learning improves in each iterative run of a base classifier. In some, it is referred to as the process of putting together a set of characteristics or meta-features specific to a domain and with respect to the classifier's performance. For example, in (Cruz, et al, 2015), the authors use meta-learning to propose a novel dynamic ensemble selection framework, where five sets of meta-features capturing different properties of the base learner is proposed for classifier selection. Their classification selection rule is learned by a meta-classifier making use of the training data. Which then enables an induced set of rules by using a meta-learner to observe what conditions makes a learning algorithm perform better than others. This is limited as the meta-learner used for this analysis is related to only specific domain characteristics and not characteristics that can cut across domains.

Another example of a most recent meta-learning approach is the ensemble classifier system for classifying multimedia big data designed in (Yan, et al, 2016). In their approach, the authors integrate the outputs of different classifiers using their confusion matrices to arrange a set of judgers in a hierarchical structured decision model.

However, in this research, a meta-learning concept is used to enable the decision learning process. The meta-learning phase of this research uses more general knowledge about supervised and unsupervised machine learning algorithms to create some hypo that is then applied in an experiment and based on the performance results of the experiments a set of decision rules are drawn.

7. AUTOMATED MACHINE LEARNING (AUTOML)

As previously stated, some learning algorithms may not be very effective for handling heterogeneous datasets for which they were not previously trained to handle in an automatic, effective and timely manner. There is the need to know how we can improve the automatic build of models using more general knowledge and information about a given dataset. The field of automated machine learning, also known as AutoML, is a fast-growing machine learning approach, designed to automate tasks of data preparation, pre-processing, and model training to ease the tasks of both intermediate and experts in the field. The autoML problem is formally defined for example in (Feurer, et al 2015) as:

Formal Definition: For i=1,…,n+m, let $x_i \in \mathbb{R}^d$ signify a feature vector and y_iÎY the corresponding target value. Given a training dataset $\mathcal{D}_{\text{train}} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and the feature vectors $x_{n+1}, \ldots, x_{n+m}$ of a test dataset $\mathcal{D}_{\text{test}} = \{(x_{n+1}, y_{n+1}), \ldots, (x_{n+m}, y_{n+m})\}$ taken from the corresponding data distribution, given a budget resource $\mathcal{B}$ (for example, computational resources such as the CPU/memory usage and/or the clock time which in practice is equal to the user's time spent) and a loss function $\mathcal{L}(.,.)$, the autoML problem is to automatically produce a set of test dataset predictions $\dot{y}_{n+1},\ldots,\dot{y}_{n+m}$. The loss of a solution $\dot{y}_{n+1},\ldots,\dot{y}_{n+m}$ to the autoML problem is specified by:

$$\frac{1}{m}\sum n_{j=1}^{m}\mathcal{L}\left(\dot{y}_{n+j},y_{n+j}\right) \tag{2.8}$$

According to (Datarobot & Triffacta), the former U.S. chief data scientist says that data cleaning takes up about eighty percent of the tasks in any data science project while Forrester records that "massive machine learning automation is the future in data science". This research focuses on improving the model training aspect of AutoML without having to spend time in the data preparation stage. This in turn will allow for a less time consuming, tedious and a costly process when building highly efficient machine learning models for big data mining.

There are many strategies one can adopt in the field of automated machine learning, two to consider is Starting High and Exhaustive Searching.

7.1. Starting High

Starting High is a machine learning method that is sophisticated and known to perform well on a range of predictive model problems, such as when random forest or gradient boosting isselected. Then the model is evaluated on the given problem and the results used as an approximate top-end benchmark, then the simplest model that achieves similar performance is found. The "Start High" approach is fast and can help you define the bounds of model skill to expect on the problem and find a simple (e.g. Naïve Bayes or Occam's Razor) model that can achieve similar results. It can also help you find out whether the problem is solvable/predictable fast, which is important because not all problems are predictable.

7.2. Exhaustive Searching

Evaluate all the machine learning methods that you can think of on the problem and select the method that achieves the best performance relative to the baseline. The "Exhaustive Search" is slow and is really intended for long-running projects where model skill is more important than almost any other concern. This is a common approach that current commercial enterprises such as 'Datarobot' and 'Rapid Miner' try to adopt for their AutoML products.

7.3. AutoML Related Works

(Sparks et al., 2015) present a system called TUPAQ designed to automate the process of training predictive models. They address the challenges of using fixed hyper parameter configurations, by achieving high quality model building via a wider search amongst the hyper parameter configuration space of Machine learning algorithms. TUPAQ takes advantage of the logical and physical optimizations for the purpose of large-scale model searching. They focus precisely on the supervised learning setting. They consider a small number of model families (linear Support Vector Machines, Logistic regression trained via gradient descent, and nonlinear SVMs that uses random features) with several hyper parameters, under the assumptions that in reality, only a small proportion of general-purpose classifiers are used in practice. The authors compare a baseline approach with the TUPAQ approach to solving the model search problem. The baseline model search approach compared with TUPAQ is the conventional model search approach using sequential grid search. Where the input is the labelled data, model space and budget, while the output is the best model. The models are trained at grid points generated on the hyper parameter space, resulting in several models being trained on one dataset. The budget which refers to the total number of models to train on a dataset is specified. Distinctively, TUPAQ includes batch size as an input, and allows for the possibility of using training history as an input. The TUPAQ architecture is made up of several components which includes the *driver* (in charge of providing the model search space and budget), the *planner* (passes the driver's information to the tuner and the tuner's configurations to the executor), the *hyper parameter tuner* (generates new model configurations to use) and the *executor* (for the actual training of models on the dataset and gives back the planner an appropriate execution strategy). TUPAQ design space makes use of four optimizations strategy namely: *cost-based execution strategy* (a model search space and budget are considered), *advanced hyper parameter tuning* (using training history as input for the hyper parameter tuning process), *bandit resource allocation* (via runtime inspections to generate on-fly decisions) and *batching* (to train multiple models simultaneously). They evaluate each design space strategy of TUPAQ on five UCI machine learning repository datasets individually, and then evaluated a combination of all the strategies together on two datasets with different learning goals. Significant improvement of the model searching process using the bandit allocation and batching strategies was observed on one of the datasets. Also, significant reduction in the search time and test error is seen with the optimisation strategies used by TUPAQ as compared with the common baseline un-optimized grid search method. The authors of TUPAQ explore in depths the effect of batching in a distributed setting and present an application of this method to the model search problem, while ensuring an optimization of the parallel execution of algorithms. The estimator in their design however needs more input from the developer of an algorithm and focuses on predicting a reasonable cluster size for a given ML model.

In (Kotthoff, et al, 2017), Auto-Weka has been designed to help users of Weka to search through all available learning algorithms and hyper parameter settings in Weka that reduces the loss due to cross validation. They achieve this by using a Bayesian optimization (highly parametric) approach to find a strong instance for the dataset given. How Auto-Weka identifies the classifier that performs best on a given dataset is by using SMAC (Sequential Model-based Algorithm Configuration). The user is asked to provide only one dataset at a time to process, a memory bound (there is a default of 1GB) as well as an overall learning time budget (the default is 15 minutes). Auto-Weka as it stands can only run the auto search on one dataset at a time and the authors advice that for auto-Weka to select the best learning scheme the user should set a minimum of 24hours. This means that to find the best learning scheme automatically for 5 different data sets, one will spend 1hour 15mins (using the default) or 120 hours/ 5 days (going by the advice of the authors) just to search for the algorithm suggestion to use. Which is still a very time-consuming process. The decision-making layer of hybrid system designed in this paper employs the use of more general characteristics of the dataset and more general knowledge learnt/known about the different learning schemes to choose faster the most ideal learning scheme, without needing to set any time budget or initial parameters (i.e. it is not a highly parametric system because it relies on less parameter space searching). This is a first step to making sure that parameter optimizations (which might improve performance), is done using the parameter set of only the selected scheme (as opposed to having a set containing all possible parameter settings of the various schemes available in Weka). Which means that learning time will be greatly improved overall.

(Feurer et al., 2015) describe an autoML system 'auto-sklearn' which uses the same type of optimizer (i.e. Bayesian optimization) as auto-Weka, includes however a smaller model and hyper parameter space than auto-Weka (they consider classifiers and pre-processors implemented in scikit-learn ML framework that are of high performance). Auto-sklearn uses additional meta-learning methods and ensemble building in its design. The results of the meta-learning method are used as a kick starter for the complex optimisation challenge of searching the hyper parameter space of a complete ML system. While their ensemble building acts as a post optimization method, where models trained during the Bayesian optimization search are built into an ensemble. However, promising auto-sklearn appears to be over autoWeka, and like the approach of meta-learning in this paper to aid auto machine learning, auto-sklearn is quite a complex system because of its use of Bayesian optimization, pre-processors, meta-learning and ensemble building. It also does not tackle semi-supervised or unsupervised problems. While in this paper, we design a non-complex system that searches for the best learning method tackling classification, regression, semi supervised and unsupervised problem areas.

Table 2. A summary of current state-of-the-art autoML systems

System	Reference	Aim	Method	A	B	C	D
Auto-SkLearn	4	Extend Auto-Weka	Highly Parametric ML Framework with Bayesian Optimization, meta-learning step, auto ensemble construction	No	No	Yes	No
AutoWeka	2	Automatic ML algorithm selection & Hyperparameter optimization	Bayesian Optimization & SMAC	No	No	Yes	No
DataRobot	6	Automatic data processing, model selction and Scoring algorithm	Supervised ML model or ensemble selection, Model building transparency, Exhaustive search of model space	No	No	Yes (Shows rankings)	No
RapidMiner	5	Automated Modelling for advanced analytical use cases.	Human friendly user interface, Several different supervised algorithms, Exhaustive search of model space, Model Transparency	No	No	Yes (shows several suggestions)	No
TuPAQ	3	Automatic ML at Scale/Supervised model search	Batching, Advanced Hyperparameter tuning, sequential grid search, Bandit resource allocation	No	No	Yes	No

Rapid Miner, a commercial data science platform introduced an additional auto model function to enhance automatic modelling which is completely transparent to the user (ROY, 2018). Their auto model function supports several learning algorithms and trains models using several learning algorithms, then ranks and mentions to the user the most suitable models they can choose from. However, there is a lot of user engagement involved to achieve the process of selecting the best model, and the user can only supply one dataset at a time as compared to the design in this research.

Table 2 below provides a summary of the current state of the art autoML systems from the literatures discussed above.

The following is referred to in Table 2,

A: Supports input and automatic processing of multiple datasets at the same time.
B: Selects Learning setting automatically.
C: Selects appropriate model.
D: Use Fixed hyper parameter Configurations.

SUMMARY

From the literatures we have been able to understand and discuss big data ML, then we build an understanding of the most widely used ML methods (supervised learning/classification and unsupervised learning/clustering), performance evaluators and statistics commonly used in testing the performance of ML algorithms. We also discuss and compare some well-known classification tools in the ML research community. From building an understanding of the most widely used ML schemes and algorithms, it has been observed that there are a lot of ML algorithms out there and a model trained on one dataset might not be useful on another dataset, that data scientists spend an awful amount of time searching and selecting the best ML algorithm to use for a given data problem, which in turn brings about the need and growth in the autoML research field. We observed from the state of the arts and literature study in the field that the algorithm selection problem and reduction of the time data scientist spend in building ML models can be greatly reduced by engaging the sub fields of autoML and meta-learning. Lastly, we have carried out some background study and comparison into some of the autoML systems out there in the research community and commercially. The next chapter discusses the methodology and pre design experimentations used in the design of an hybrid-autoML system.

REFERENCES

Achcar, F., Camadro, J.-M., & Mestivier, D. (2009). AutoClass@ IJM: A powerful tool for Bayesian classification of heterogeneous data in biology. *Nucleic Acids Research*, *37*(suppl_2), gkp430. doi:10.1093/nar/gkp430 PMID:19474346

Aggarwal, C. C. (2014). *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC. doi:10.1201/b17320

Aggarwal, C. C. (2014c). A Survey of Stream Classification Algorithms. *Data Classification: Algorithms and Applications, 245*.

Aggarwal, C. C., & Reddy, C. K. (2013). *Data Clustering: Algorithms and Applications* (Vol. 31). Chapman and Hall/CRC. doi:10.1201/b15410

Aha, D. W. (1992). Generalizing from case studies: A case study. *Proc. of the 9th International Conference on Machine Learning*. 10.1016/B978-1-55860-247-2.50006-1

Akerkar, R. (2013). *Big Data Computing*. CRC Press. doi:10.1201/b16014

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2010). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, *17*(2-3), 255–287.

Alelyani, S., Tang, J., & Liu, H. (2013). Feature Selection for Clustering: A Review. *Data Clustering: Algorithms and Applications, 29*.

Alsallakh, B., Hanbury, A., Hauser, H., Miksch, S., & Rauber, A. (2014). Visual Methods for Analyzing Probabilistic Classification Data. *IEEE Transactions on Visualization and Computer Graphics*, *20*(12), 1703–1712. doi:10.1109/TVCG.2014.2346660 PMID:26356884

Angelov, P., & Yager, R. (2012). A new type of simplified fuzzy rule-based system. *International Journal of General Systems*, *41*(2), 163–185. doi:10.1080/03081079.2011.634807

Azar, A. T., & El-Said, S. A. (2013). Probabilistic neural network for breast cancer classification. *Neural Computing & Applications*, *23*(6), 1737–1751. doi:10.100700521-012-1134-8

Bankert, R. L. (1994). Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network. *Journal of Applied Meteorology*, *33*(8), 909–918. doi:10.1175/1520-0450(1994)033<0909:CCOAII>2.0.CO;2

Ben-David, A. (2008). Comparison of classification accuracy using Cohen's Weighted Kappa. *Expert Systems with Applications*, *34*(2), 825–832. doi:10.1016/j.eswa.2006.10.022

Bertini, J. R., & Zhao, L. (2013). *A Comparison of Two Purity-Based Algorithms When Applied to Semi-supervised Streaming Data Classification*. Paper presented at the BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence. 10.1109/BRICS-CCI-CBIC.2013.15

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive Online Analysis. *Journal of Machine Learning Research*, *11*, 1601–1604.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blunsom, P. (2004). Hidden Markov models. *Lecture Notes, 15*, 18-19.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC Press.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 27. doi:10.1145/1961189.1961199

Cheeseman, P., Self, M., Kelly, J., & Stutz, J. (1996). *Bayesian Classification*. AutoClass.

Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, *19*(2), 171–209. doi:10.100711036-013-0489-0

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*(4), 261–283. doi:10.1007/BF00116835

Cohen, W. W. (1995). Fast effective rule induction. *Proceedings of the twelfth international conference on machine learning*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. doi:10.1007/BF00994018

Costa, B. S. J., Angelov, P. P., & Guedes, L. A. (2015). Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier. *Neurocomputing*, *150*, 289–303. doi:10.1016/j.neucom.2014.05.086

Cruz, R. M., Sabourin, R., Cavalcanti, G. D., & Ren, T. I. (2015). META-DES: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, *48*(5), 1925–1935. doi:10.1016/j.patcog.2014.12.003

Efron, B., & Tibshirani, R. (1997). Improvements on cross-validation: The 632+ bootstrap method. *Journal of the American Statistical Association*, *92*(438), 548–560.

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC Press. doi:10.1201/9780429246593

Esposito, F., Malerba, D., Semeraro, G., & Kay, J. (1997). A comparative analysis of methods for pruning decision trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19*(5), 476-491.

Fabrico, L. (2014). *Data Mining Classification*. Academic Press.

Fan, W., & Bifet, A. (2013). Mining big data: Current status, and forecast to the future. *SIGKDD Explorations*, *14*(2), 1–5. doi:10.1145/2481244.2481246

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). *Efficient and robust automated machine learning*. Paper presented at the Advances in Neural Information processing systems.

Fischetti, M. (2015). Fast training of Support Vector Machines with Gaussian kernel. *Discrete Optimization*.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289–1305.

Francisci Morales, D. G., & Bifet, A. (2015). SAMOA: Scalable Advanced Massive Online Analysis. *Journal of Machine Learning Research*, *16*(Jan), 149–153.

Gurov, S. I. (2013). Estimation of the reliability of a classification algorithm as based on a new information model. *Computational Mathematics and Mathematical Physics*, *53*(5), 640–646. doi:10.1134/S0965542513050059

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, *11*(1), 10–18. doi:10.1145/1656274.1656278

Haralick, R. M., Shanmugam, K., & Dinstein, I. H. (1973). Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, (6), 610-621.

Hassan, M. R., Ramamohanarao, K., Karmakar, C., Hossain, M. M., & Bailey, J. (2010). *A novel scalable multi-class ROC for effective visualization and computation. In Advances in Knowledge Discovery and Data Mining*. Springer.

Ingersoll, G. (2009). *Introducing apache mahout. Scalable, commercial friendly machine learning for building intelligent applications*. IBM.

Iounousse, J., Er-Raki, S., El Motassadeq, A., & Chehouani, H. (2015). Using an unsupervised approach of Probabilistic Neural Network (PNN) for land use classification from multitemporal satellite images. *Applied Soft Computing*, *30*, 1–13. doi:10.1016/j.asoc.2015.01.037

Jacques, J., & Preda, C. (2014). Functional data clustering: A survey. *Advances in Data Analysis and Classification*, *8*(3), 231–255. doi:10.100711634-013-0158-y

Jain, A., Murty, M., & Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*(3), 264–323. doi:10.1145/331499.331504

Jain, A. K., & Waller, W. G. (1978). On the optimal number of features in the classification of multivariate Gaussian data. *Pattern Recognition*, *10*(5), 365–374. doi:10.1016/0031-3203(78)90008-0

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*.

Jordan, A. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, *14*, 841.

Kohavi, R. (1995). *A study of cross-validation and bootstrap for accuracy estimation and model selection*. Paper presented at the IJCAI.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT Press.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2017). Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, *18*(1), 826–830.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2017). Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, *18*(1), 826–830.

Krawczyk, B., Stefanowski, J., & Wozniak, M. (2015). Data stream classification and big data analytics. *Neurocomputing*, *150*, 238–239. doi:10.1016/j.neucom.2014.10.025

Kwak, N., & Choi, C.-H. (2002). Input feature selection for classification problems. *Neural Networks, IEEE Transactions on, 13*(1), 143-159.

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. Academic Press.

Law, Y.-N., & Zaniolo, C. (2005). An adaptive nearest neighbor classification algorithm for data streams Knowledge Discovery in Databases. *PKDD*, *2005*, 108–120.

Leiva-Murillo, J. M., Gomez-Chova, L., & Camps-Valls, G. (2013). Multitask Remote Sensing Data Classification. *IEEE Transactions on Geoscience and Remote Sensing*, *51*(1), 151–161. doi:10.1109/TGRS.2012.2200043

Li, L. (2015). *Support Vector Machines Selected Applications of Convex Optimization*. Springer.

Li, T., Zhang, C., & Ogihara, M. (2004). A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics (Oxford, England)*, *20*(15), 2429–2437. doi:10.1093/bioinformatics/bth267 PMID:15087314

Lin, Z., Yan, C., Yan, L., & Nan, L. (2008). *Application of Data Mining Classification Algorithms in Customer Membership Card Classification Model.* Paper presented at the Information Management, Innovation Management and Industrial Engineering, 2008. ICIII '08. International Conference on.

Liu, H., Li, J., & Wong, L. (2002). A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics, 13*, 51-60.

Liu, H., & Motoda, H. (1998). *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media. doi:10.1007/978-1-4615-5689-3

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on, 17*(4), 491-502.

Loh, W. Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews. Data Mining and Knowledge Discovery*, *1*(1), 14–23. doi:10.1002/widm.8

Lu, N., Mabu, S., Mabu, S., Li, W., Hirasawa, K., & Hirasawa, K. (2010). *Hybrid rule mining based on fuzzy GNP and probabilistic classification for intrusion detection*. Paper presented at the SICE Annual Conference.

Lukasiewicz, T. (2008). Expressive probabilistic description logics. *Artificial Intelligence*, *172*(6), 852–883. doi:10.1016/j.artint.2007.10.017

Ma, B. L. W. H. Y. (1998). Integrating classification and association rule mining. *Proceedings of the fourth international conference on knowledge discovery and data mining*.

Majnik, M., & Bosnic, Z. (2013). ROC analysis of classifiers in machine learning: A survey. *Intelligent Data Analysis*, *17*(3), 531–558. doi:10.3233/IDA-130592

Maravall, D., De Lope, J., & Fuentes, J. P. (2013). Fusion of probabilistic knowledge-based classification rules and learning automata for automatic recognition of digital images. *Pattern Recognition Letters*, *34*(14), 1719–1724. doi:10.1016/j.patrec.2013.03.019

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, *55*(1), 169–186. doi:10.1016/S0925-2312(03)00431-4

Mladeni'c, D., & Grobelnik, M. (1998). *Feature selection for classification based on text hierarchy*. Paper presented at the Text and the Web, Conference on Automated Learning and Discovery CONALD-98.

Mossman, D. (1999). Three-way rocs. *Medical Decision Making*, *19*(1), 78–89. doi:10.1177/0272989X9901900110 PMID:9917023

Murphy, K. P. (2006). *Naive bayes classifiers*. University of British Columbia.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT Press.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, *2*(4), 345–389. doi:10.1023/A:1009744630224

Nielsen, J. D., Rumí, R., & Salmerón, A. (2009). Supervised classification using probabilistic decision graphs. *Computational Statistics & Data Analysis*, *53*(4), 1299–1311. doi:10.1016/j.csda.2008.11.003

Nosofsky, R. M., & Little, D. R. (2010). Classification response times in probabilistic rule-based category structures: Contrasting exemplar-retrieval and decision-boundary models. *Memory & Cognition*, *38*(7), 916–927. doi:10.3758/MC.38.7.916 PMID:20921104

Owen, S., Anil, R., Dunning, T., & Friedman, E. (2011). *Mahout in action: Manning Shelter Island*. Academic Press.

Pal, M., & Foody, G. M. (2010). Feature selection for classification of hyperspectral data by SVM. *Geoscience and Remote Sensing, IEEE Transactions on, 48*(5), 2297-2307.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27*(8), 1226-1238.

Pizzuti, C., & Talia, D. (2003). P-autoclass: Scalable parallel clustering for mining large data sets. *Knowledge and Data Engineering, IEEE Transactions on, 15*(3), 629-641.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers, 10*(3), 61-74.

Pratama, M., Anavatti, S. G., Joo, M., & Lughofer, E. D. (2015). pClass: An Effective Classifier for Streaming Examples. *IEEE Transactions on Fuzzy Systems*, *23*(2), 369–386. doi:10.1109/TFUZZ.2014.2312983

Punch, W. F., III, Goodman, E. D., Pei, M., Chia-Shun, L., Hovland, P. D., & Enbody, R. J. (1993). *Further Research on Feature Selection and Classification Using Genetic Algorithms*. Paper presented at the ICGA.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. doi:10.1007/BF00116251

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286. doi:10.1109/5.18626

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). *Cross-validation Encyclopedia of database systems*. Springer.

Rice, J. R. (1975). *The algorithm selection problem*. Academic Press.

Rokach, L., & Maimon, O. (2010). Classification Trees. In O. Maimon & L. Rokach (Eds.), *Data Mining and Knowledge Discovery Handbook* (pp. 149–174). Springer US.

Roy, K. (2018). *RapidMiner looks to boost data scientists' productivity with Auto Model*. Retrieved from 451 Research: https://rapidminer.com/resource/451-research-report-auto-model/

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Sheikholesalmi, F., Mardani, M., & Giannakis, G. B. (2014). *Classification of streaming big data with misses*. Paper presented at the 48th Asilomar Conference on Signals, Systems and Computers.

Sinha, K. (2014). Semi-Supervised Learning. In C. C. Aggarwal (Ed.), Data Classification: Algorithms and Applications. Academic Press.

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, *41*(1), 6. doi:10.1145/1456650.1456656

Sokal, R. R. (1974). Classification: Purposes, principles, progress, prospects. *Science*, *185*(4157), 1115–1123. doi:10.1126cience.185.4157.1115 PMID:17835456

Sparks, E. R., Talwalkar, A., Haas, D., Franklin, M. J., Jordan, M. I., & Kraska, T. (2015). Automating model search for large scale machine learning. *Proceedings of the Sixth ACM Symposium on Cloud Computing*. 10.1145/2806777.2806945

Suthaharan, S. (2014). Big data classification: Problems and challenges in network intrusion prediction with machine learning. *Performance Evaluation Review*, *41*(4), 7. doi:10.1145/2627534.2627557

Tang, J., Alelyani, S., & Liu, H. (2014). *Feature selection for classification: A review. Data Classification: Algorithms and Applications* (C. Aggarwal, Ed.). CRC Press.

Tankard, C. (2012). Big data security. *Network Security*, *2012*(7), 5–8. doi:10.1016/S1353-4858(12)70063-6

Tekin, C., & van der Schaar, M. (2013). *Distributed online big data classification using context information.* Paper presented at the Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on. 10.1109/Allerton.2013.6736696

Triguero, I., García, S., & Herrera, F. (2013). Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study. *Knowledge and Information Systems*, *42*(2), 245–284. doi:10.100710115-013-0706-y

Tung, A. K. (2009). *Rule-based Classification Encyclopedia of Database Systems*. Springer.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, *73*(2), 185–214. doi:10.100710994-008-5077-3

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*(2), 77–95. doi:10.1023/A:1019956318069

Wang, X., & Pardalos, P. M. (2015). A Survey of Support Vector Machines with Uncertainties. *Annals of Data Science*, *1*(3-4), 293–309. doi:10.100740745-014-0022-8

Yan, Y., Zhu, Q., Shyu, M.-L., & Chen, S.-C. (2016). *A Classifier Ensemble Framework for Multimedia Big Data Classification*. Academic Press.

Yunck, T. P. (1976). A technique to identify nearest neighbors. *Systems, Man and Cybernetics, IEEE Transactions on*, (10), 678-683. doi:10.1145/2627534.2627557

Zhang, C., & Zhang, S. (2002). *Association rule mining: models and algorithms*. Springer-Verlag. doi:10.1007/3-540-46027-6

Zhao, Z., & Liu, H. (2007). *Semi-supervised Feature Selection via Spectral Analysis*. Paper presented at the SDM. 10.1137/1.9781611972771.75

Chapter 10 Methodology for the Research Conducted

ABSTRACT

This chapter describes and discusses a combination of research methodologies (e.g., experimental, theoretical, and systems design) used in this research, allowing us to eliminate as much as possible every limitation that can be encountered with the individual methods themselves. For example, experimental research methodology has a limitation because the experiments are performed mainly in a controlled environment and might not reflect properly some practices performed 'in the wild'. But combining this with some survey and prototype (system's) design reduced such limitations. The knowledge gained from carrying out preliminary experimentation is used in the next chapter to design and model the Hybrid-AutoML system.

1. INTRODUCTION

This research uses a mixture of several research methods, briefly described as follows:

Pure Research: based on the summary themes of (Baban et al., 2009) in (Hassani, 2017). This methodology aims to enable us to discover new knowledge without expecting an instant mark on the present state of things in the field.

Exploratory research: aims at discovering useful information in the field, which previous information cannot be found in order to develop reflective hypothesis.

Descriptive research: aims at explaining what the situation and characteristics of a problem is, as a benefit for another or other research areas.

Experimental methodology: experiments help us test the accuracy of concepts/theories and hypothesis. In computer science, it is often used to analyse behaviours and performance, in many different fields such as automating theorem proving, machine learning, etc. There is often the need to also use some tools or methods (e.g. statistical analysis) in conjunction with the experimental method. Doing that will help in

DOI: 10.4018/978-1-7998-7316-7.ch010

proving and backing up the legibility of the work developed and whether the hypothesis is supported. It is important that all experiments are reproducible by clearly explaining the steps carried out during the experiments and tools/resources used for the experimentation.

Theoretical Methodology: this is a methodology related based on mathematics and logic. Ideas can be an existence of conceptual and formal models e.g. data models and algorithms. Since this methodology is based on logic and mathematics, some ways in which it deals with problems is through iterations, initiations and recursions. Developing theories is important to build ideas, reason about programs, improve logic and semantics in order to prove accuracy of the concept and formal models. Theoretical methodology through dedicated designs and algorithm analysis help us unravel improved solutions (e.g. improved performance solutions). However beneficial this method is, it still requires other methods that can help prove efficiency of new models/theories designed. For example, in the machine learning field if a new classifier is to be designed, often the developer using mathematical of theoretical methodologies will require a proof of model efficiency by consuming one or more previous techniques. Since this approach is based on mathematics there is a limitation that the mathematical abstractions used in a proof maybe too abstract/generic that it ignores completely some serious issues that need to be considered in the actual system implementation.

Systems Design Methodology: a methodology consisting mainly of five stages namely, 'design of concept', 'system architecture construction', 'prototype building', 'product development' and 'technology transfer'. This research work performs the first three up to prototype building. Prototype building helps us have a proof of concept for feasibility demonstration. However, the aim is to later go further into the product development stage, once this research work has received due evaluation and acceptance in the wider research community.

The reasons/benefits of using this multimethod logical approach is highlighted as follows:

- It helps to tackle the research area properly,
- It reveals in a better manner, the characteristics of the research.
- It allows the research to be conducted in a very effective and orderly manner.

This chapter gives a detailed and logically ordered plan of the approach, techniques, procedures and steps followed to achieve the research aims and objectives.

2. METHODS

A list of the steps carried out in this research is summarised as follows:

1. Theoretical studies.
2. A mini data classification survey.
3. Analysing some key limitations from the state-of-the-art big data classification and auto ML systems.
4. Preliminary Hypothesis and Experimentations.
5. Evaluation and Analysis of the pre-experimental results.
6. Gained knowledge from the pre-experimental results.
7. Design and modelling of the hybrid autoML system using the knowledge gained.

8. Programming of this system using java object-oriented programming.
9. Testing the designed hybrid autoML system model to proof it works.
10. Evaluation and analysis of the results from the test and comparison with other autoML systems discussed in the literature.

2.1 Reviewing Literatures

As a basis of this research, knowledge about big data, machine learning, data classification, clustering algorithms, autoML systems, meta-learning, their applications in the field etc. is developed through the study of journals, articles, books, etc. The aim of which was to build, nurture & improve our knowledge and understanding of what is current (including limitations) in the field of data mining and machine leaning. Information gained from undergoing this is discussed in Chapter 2 and across this research.

2.2. Mini Survey

Using an online survey tool called Qualtrics, in 2015 a mini survey is designed to determine the knowledge, importance and application of big data classification and classification tools amongst data science professionals. It helped to determine further and justify the relevance of big data classification in the field. A link to the survey was distributed to former work colleagues of mine who are data scientist, posted on researchgate and linkedIn (both online platforms for professionals in the field). The raw questions asked can be located in Appendix 1. The results from the survey shows that the majority (85%) of those who hear about big data also hear about data classification. It also showed that about 41% of the participants agreed big data classification has many application areas including improving security measures of a system through advanced prediction of threats. On the use of big data classification tools, the majority (38%) said that they don't use any big data classification tool.

2.3. Hypothesis and Assumptions

Hypothesis 1: if given a large labelled train data set $\mathcal{D}_{train}$ and a corresponding test data set $\mathcal{D}_{test}$ on which some prediction is to be made. The size ratio of the training data to the test data will affect the accuracy of the model built upon any given algorithm.

Hypothesis 2: A supervised learner will be more appropriate than an unsupervised learner. Given a data set $\mathfrak{D}$, with an already existing large set of pre-labelled training data $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$ which is relatively smaller in size than $\mathcal{D}_{train}$, and based on general knowledge gained about supervised learners performing well in the presence of a larger pre-labelled $\mathcal{D}_{train}$.

Hypothesis 3: If Hypothesis 2 is true, and a supervised ML algorithm is more desirable to be selected than an unsupervised ML algorithm, then we assume that general information about the instances and attributes of the dataset such as the size of the training data, the number of attributes, the types of attributes found, the class attribute type, etc. will influence the choice of selecting the best supervised learning algorithm to use on a dataset.

Hypothesis 4: An unsupervised self-evolving learner will be more appropriate than a supervised learner. Given a data set $\mathfrak{D}$, without pre-labelled training data instances and the knowledge that unsupervised learners are best used when no pre-labelled training dataset exists.

3. PRELIMINARY EXPERIMENTS

Pre-experimentation has been done to tests some hypothesis and assumptions made from studying the state-of-the-art literature in the data mining and machine learning field. These experiments were aimed at proving or disproving some knowledge and limitations gained from the background study which this project aimed to design a system model to help overcome some of the limitations identified.

3.1. Experiment Materials

In undergoing research experiments, all essential materials need to be determined and organised. Materials here and in most computer science project refers to the software tools, technologies, programming language and data used for the project. There is usually more than one software tool or programming language that can be used to achieve one's aim when it comes to building software solutions. It is important to highlight the aims and reasons for using the tools and programming language chosen. The following sections under this 'Materials' section aims to highlight and give more details into what was used for this research.

The reasons it was used is because, 1) it an open source data-mining tool designed mainly for research purposes and widely accepted in the data mining community, 2) it is java based and java is a familiar object oriented programming language used for developing scalable commercial or research systems and services, 3) It contains a lot of functionalities for performing data mining tasks easily and can help assist in the development and testing of new machine learning algorithms and systems 4) it has both a simple Graphical User Interface (GUI) and an API that helps to build standard customisable machine learning applications in any way desired. Information comparing Weka with some other tools in the field is discussed in section 2.4.

The Weka GUI is used for all pre-experiments in this research while the Weka API is used for implementation of the model Designed after the pre experiments.

When Weka is downloaded and launched, appendix 3 shows a representation of the GUI and other related tabs of the GUI for Weka.

3.2. Big Data

A variety of datasets collected from the UCI machine learning (Dua & Taniskidou, 2017) & KDnuggets data repository, as well as from the Weka tool data and auto-Weka (Lars et al, 2017) repositories is used. Weka has a special format for its dataset, called the 'Attribute Relation File Format (. ARFF)'. Which is an ASCII text file describing a list of instances that share a set of attributes. Weka however, provides through its GUI the ability to load '.CSV' files and manually select other file loaders. However, using the Weka API allows us easily work with a variety of datasets such as '.CSV', '.TXT', '.XML', '. JSON', etc. Or even by accessing databases directly using JDBC.

For the experimentation and implementation tests of the system modelled in this research, a variety of datasets in different formats, collected from the various sources, were collected and placed in a 'data' directory with sub directories within it. A total of about thirty-five different datasets were used. A full list of the various datasets can be found in Appendix 4. Although for simplicity, we will be discussing the experiments in this section using just a few of the datasets. Doing this, will help drive the clarity of the knowledge learnt from the experiments.

Table 1. A list of datasets used for preliminary experiments, taken as a subset from the full list of datasets used in this research

Dataset	# Instances	#Attributes	Class attribute type	Missing Values
contact-lenses	24	All nominal (5)	Nominal	No
Cpu	209	All numeric (7)	Numeric	No
cpu.with.vendor	209	1 Nominal, 7 Numeric	Numeric	No
credit-g	1000	14 Nominal, 7 Numeric	Nominal	No
Diabetes	768	8 Numeric, 1 Nominal	Nominal	No
Glass	214	9 Numeic, 1 Nominal	Nominal	No
Ionosphere	351	34 numeric, 1 Nominal	Nominal	No
iris.2D	150	2 Numeric, 1 Nominal	Nominal	No
Labor	57	9 nom, 8 numeric	Nominal	Yes (2%)
reutersCorn-train	1554	String	Nominal	No
segment-challenge	1500	19 Numerical, 1 Nominal	Nominal	No
Soybean	683	36 Nominal	Nominal	Yes (<1%)
Supermarket	4627	217 nominal	Nominal	Up tp 77%
Unbalanced	856	32 numerical, 1 Nominal	Nominal	No
Vote	435	17 nominal	Nominal	Yes (3%)
weather.nominal	14	5 nominal	Nominal	No
weather.numeric	14	2 Numeric, 3 Nominal	Nominal	No
Dexter	420	20001 Numeric	Numeric	No

3.3. Experimental Setup

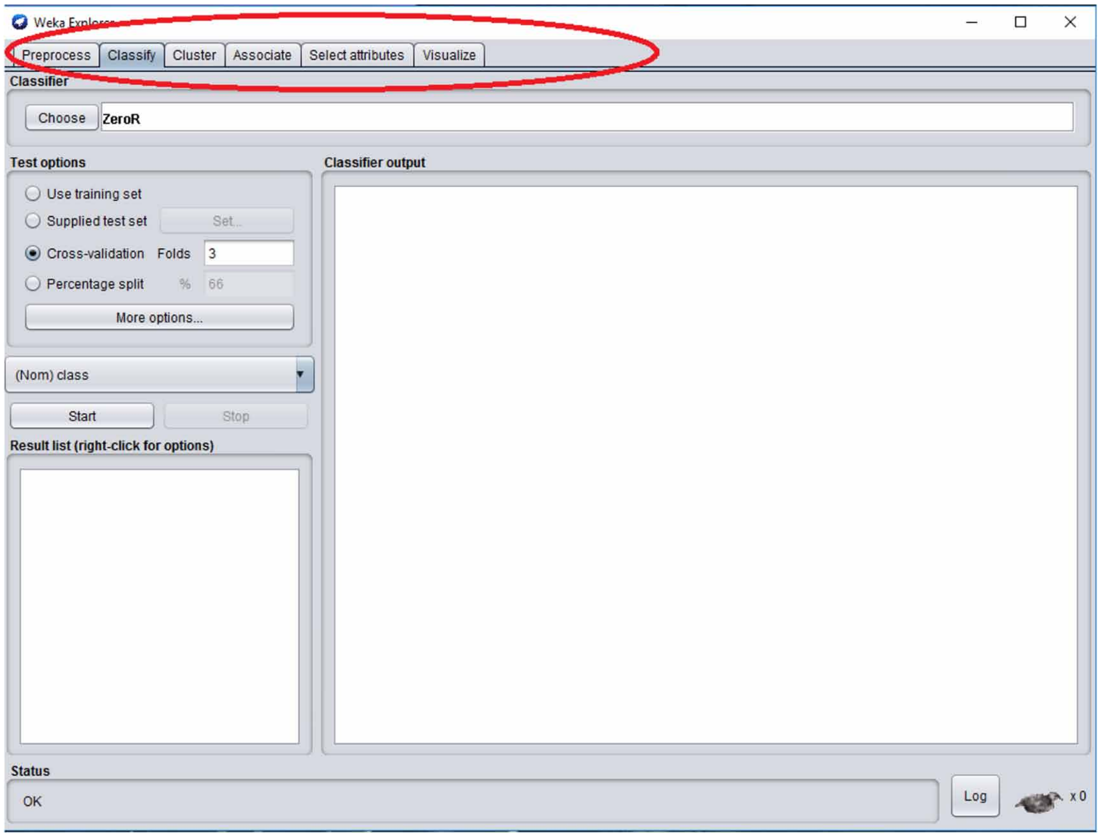
All preliminary experiments were conducted using both the Weka explorer, knowledge flow and experimenter GUIs. Performed to:

1. Prove or disprove the hypothesis made in previous section 3.1.3 to determine what general factors about a dataset will make a particular machine learning algorithm more suitable than another.
2. Determine the resulting performances of supervised and unsupervised algorithms present in Weka and what factors or characteristics of the data influenced their performance.
3. To identify limitations and knowledge in selecting the best algorithm for building a machine learning model.

All the supervised and unsupervised algorithms listed in the experiment materials section (section 3.3) were tested multiple times on the different datasets.

Setup as follows when need be:

Figure 1. Weka explorer 'Classifier' tab



ZeroR

- Use training set
- Supplied test set
- [✓ Cross-validation] Folds 3
- Percentage split % 66

(Nom) class

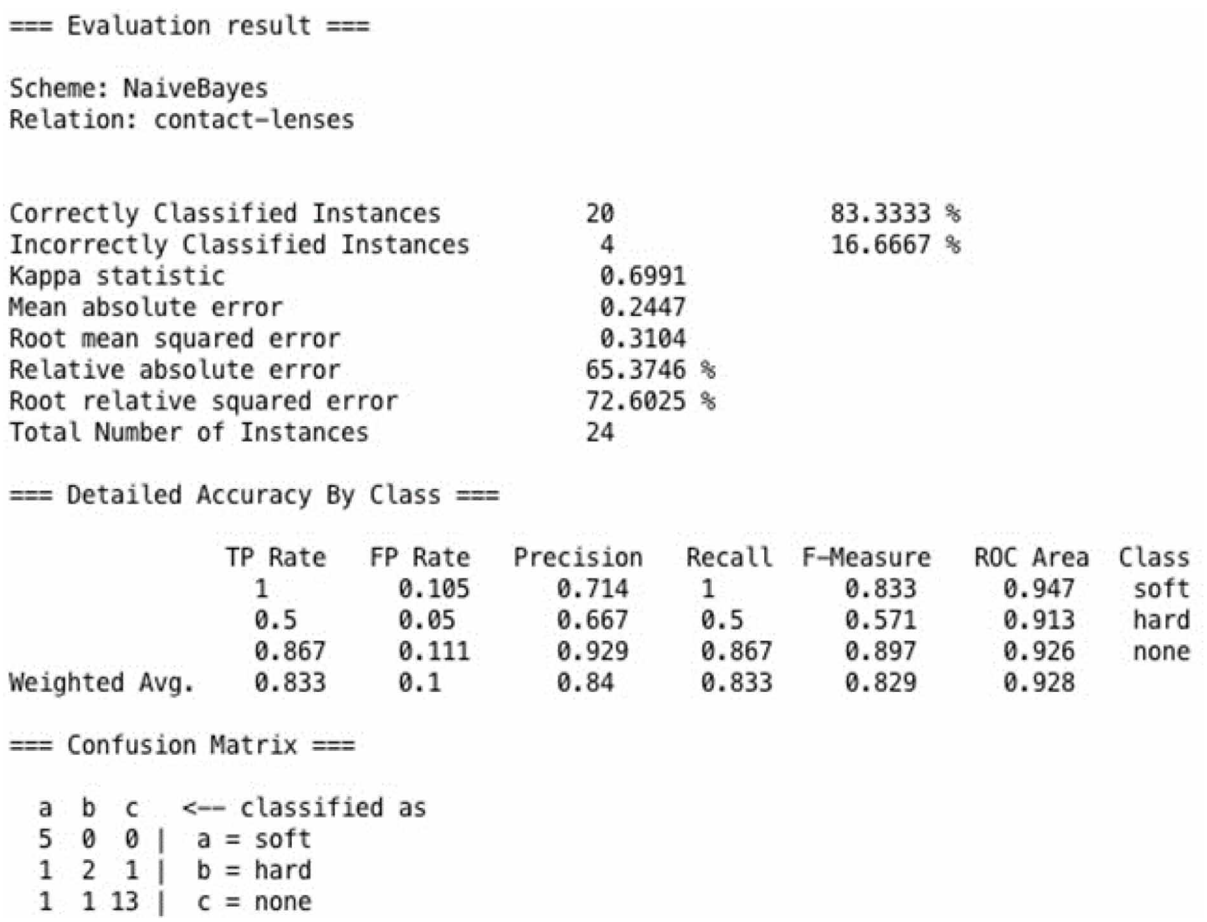
OK

When Weka 'Explorer' is launched, we are presented with its GUI's 'preprocess' tab by default. Which we can then toggle between the other explorer options such as the 'classify' and 'cluster' tab using the tool bar above (as seen in the Figure 3.1 above). There are varying tests options that can and were used during the experiments. For example, the number of folds can be played with by adjusting the 'Cross-validation' option in the test options window. When the model builds and testing has been performed, the results are displayed in the classifier output window as follows:

In the experiment seen in Figure 3 above, the selected dataset is loaded in by configuring the 'ArffLoader'. The 'ClassAssigner' determines what the class label in the dataset is. A 'Cross Validation FoldMaker' and a 'Train Test SplitMaker' where used interchangeably to split the dataset into training and test sets. Several supervised algorithms were used during different runs of the experiment instead of just a 'NaiveBayes' classifier alone.

The 'Cross Validation FoldMaker' allows cross validation evaluation to be carried out on the dataset. Clicking on it in the knowledge flow setup will allow the number of k-folds to be set. The number of folds chosen has been experimentally proven to have an effect on the performance results of the classifier, by varying the number of folds in the experiment. After several fold variations, it was discovered that three and ten folds are more relevant. Hence, only the results for the three and ten folds' experiments are recorded.

Figure 2. Output Result window display for a 'classifier' in Weka



```
=== Evaluation result ===

Scheme: NaiveBayes
Relation: contact-lenses


Correctly Classified Instances          20               83.3333 %
Incorrectly Classified Instances         4               16.6667 %
Kappa statistic                          0.6991
Mean absolute error                      0.2447
Root mean squared error                  0.3104
Relative absolute error                 65.3746 %
Root relative squared error             72.6025 %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.105     0.714     1         0.833      0.947    soft
                 0.5       0.05      0.667     0.5       0.571      0.913    hard
                 0.867     0.111     0.929     0.867     0.897      0.926    none
Weighted Avg.    0.833     0.1       0.84      0.833     0.829      0.928

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  1  2  1 |  b = hard
  1  1 13 |  c = none
```

Figure 3. Example of a single supervised Learning knowledge flow setup in Weka

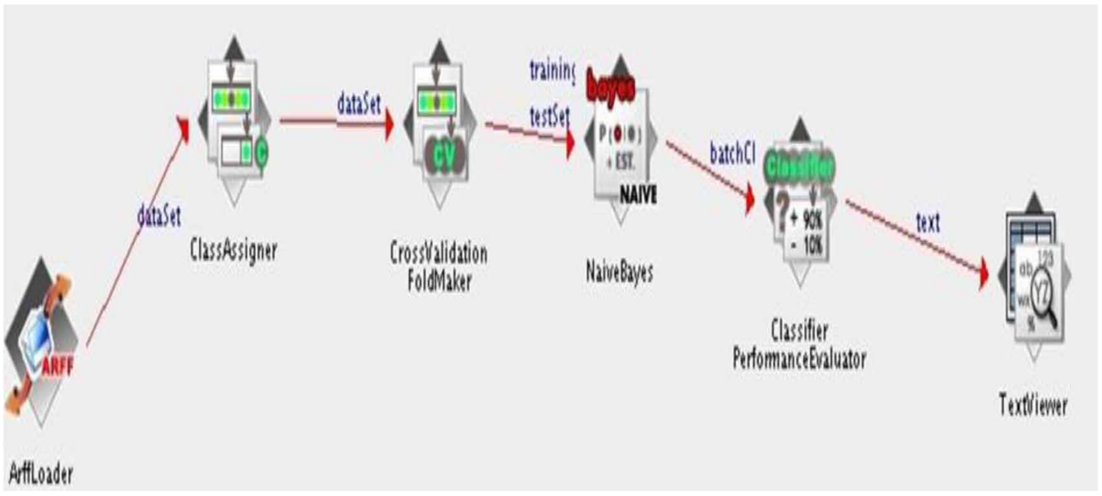


Figure 4. Unsupervised Learning knowledge flow setup in Weka

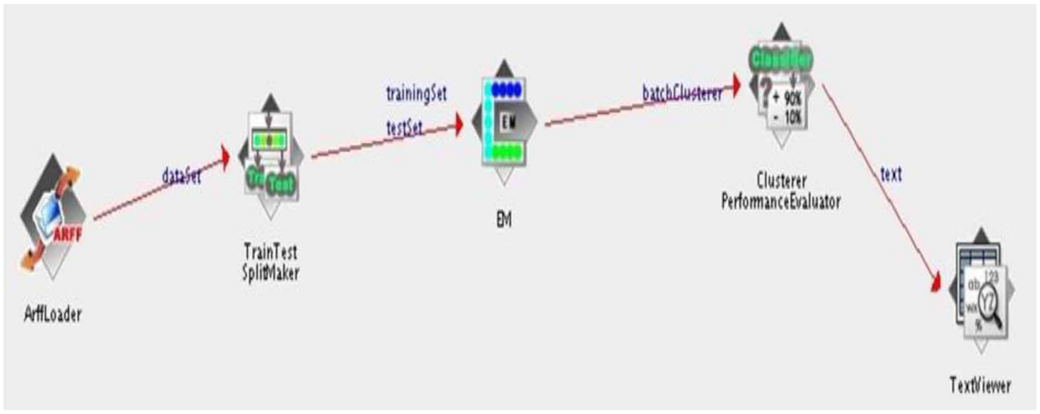
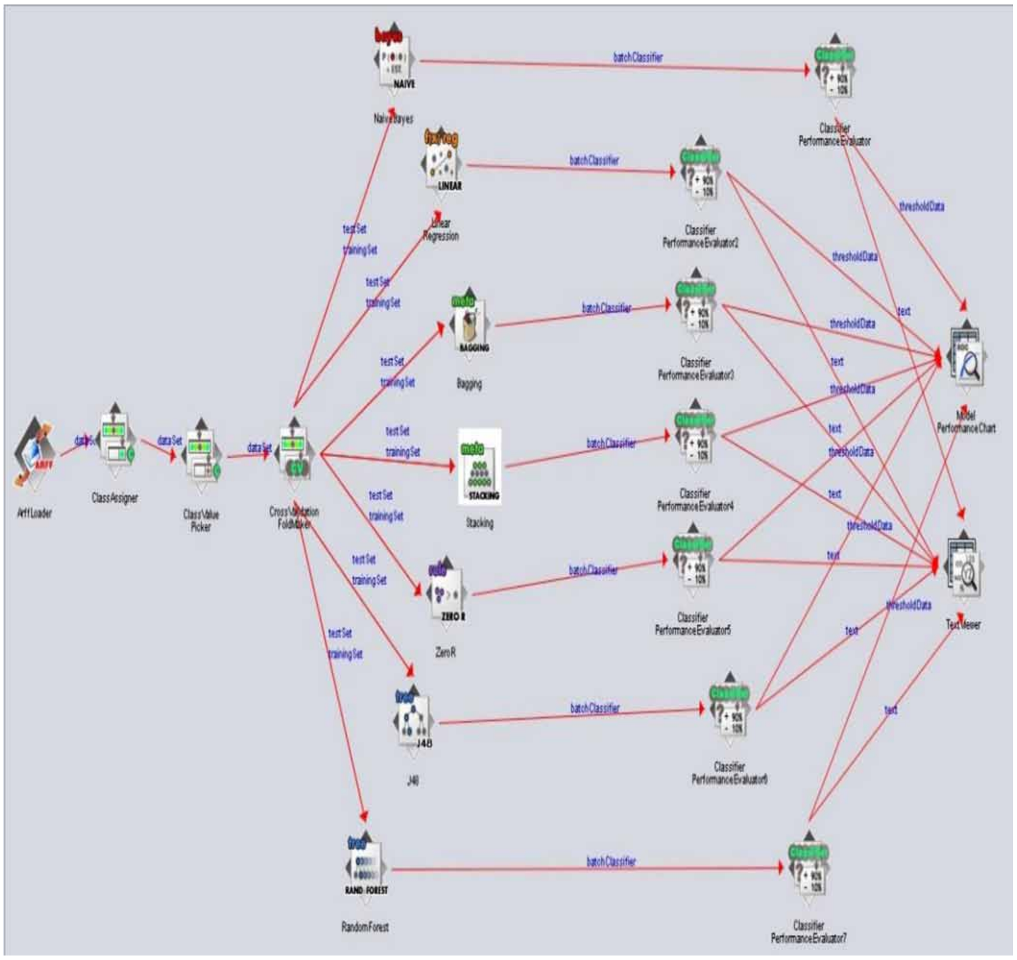


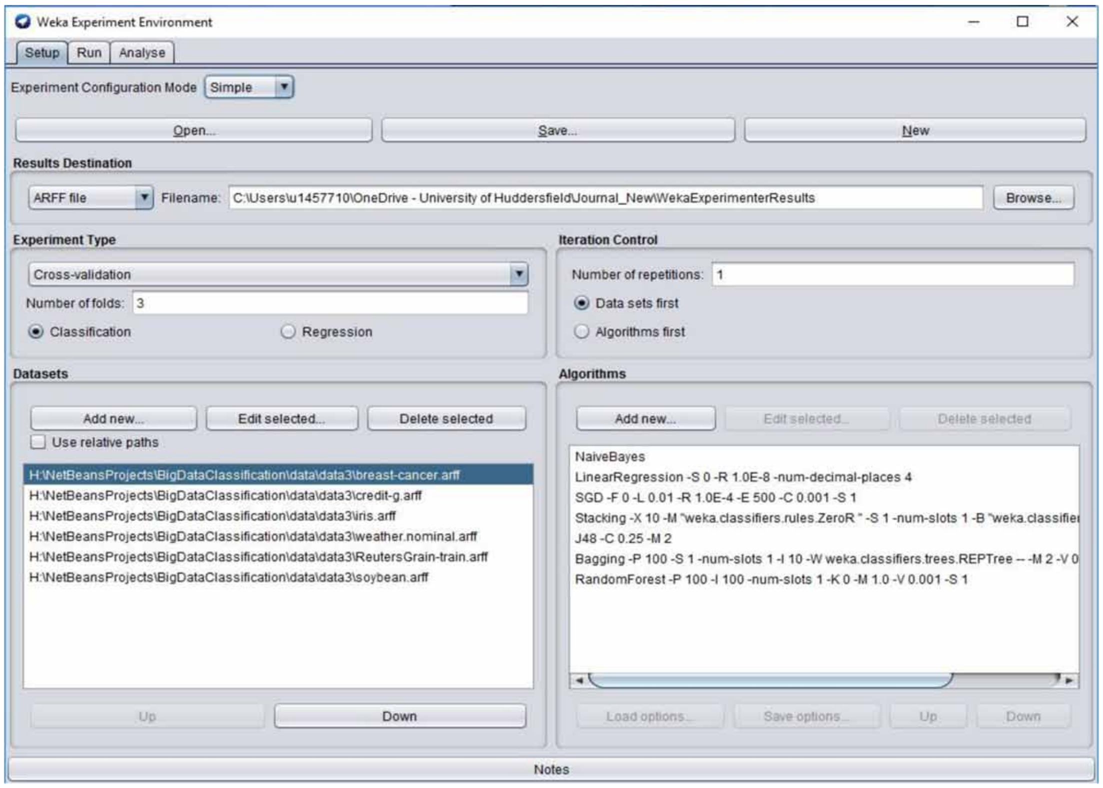
Figure 5. Knowledge flow setup for testing several classification algorithms on a given dataset in parallel


ArffLoader
ClassAssigner
ClassValue Picker
CrossValidation FoldMaker
dataSet
testSet
trainingSet
NaiveBayes
Linear Regression
Bagging
Stacking
ZeroR
J48
RandomForest
batchClassifier
Classifier PerformanceEvaluator
Classifier PerformanceEvaluator2
Classifier PerformanceEvaluator3
Classifier PerformanceEvaluator4
Classifier PerformanceEvaluator5
Classifier PerformanceEvaluator6
Classifier PerformanceEvaluator7
thresholdData
text
Model PerformanceChart
TextViewer

As earlier stated in previous sections, when using Weka's Knowledge flow GUI, we can set multiple classifiers in the process flow. The setup in Figure 5 above shows an example of such a scenario. Right clicking on the 'ARFFLoader' (which is the input) in the flow above enables us load up the dataset under consideration. After which, we can then configure and pass this dataset onto the different classifiers through the various perspectives in the setup, and then run the setup. By default, the last attribute index in the dataset is taken as the class index. But this was varied easily by right clicking and using the configuration settings of the 'ClassAssigner' in the setup. The 'ClassValuePicker' was used to specify what class label needs to be determined. The 'CrossValidationFoldMaker' was used to configure k-Folds Cross-Validation analysis. By default, 10 folds is set, but this was also tweaked during various runs of the experiment using the configuration settings of the cross-validation fold maker. The number of classifiers to evaluate on a given dataset can be increased easily in the setup. When the setup is run, if a classifier in the setup is unsuitable for that particular dataset, an error is logged and the analysis interrupted. An advantage of using this setup during the initial implementation tests, is that we are able to visualize and analyse the performance of the different classifiers by plotting their ROC curves on a single graph. A disadvantage of using just the knowledge flow, is that you can only experiment on one dataset at a time. This is where using the 'Experimenter' setup in the Weka GUI is useful.

The experimenter setup is as follows:

Figure 6. Experimenter setup for testing several classification algorithms on various datasets



Experiment Configuration Mode: Simple

Results Destination: ARFF file Filename: C:\Users\u1457710\OneDrive - University of Huddersfield\Journal_New\WekaExperimenterResults

Experiment Type: Cross-validation
Number of folds: 3
[✓ Classification] Regression

Iteration Control
Number of repetitions: 1
[✓ Data sets first] Algorithms first

Datasets
Use relative paths

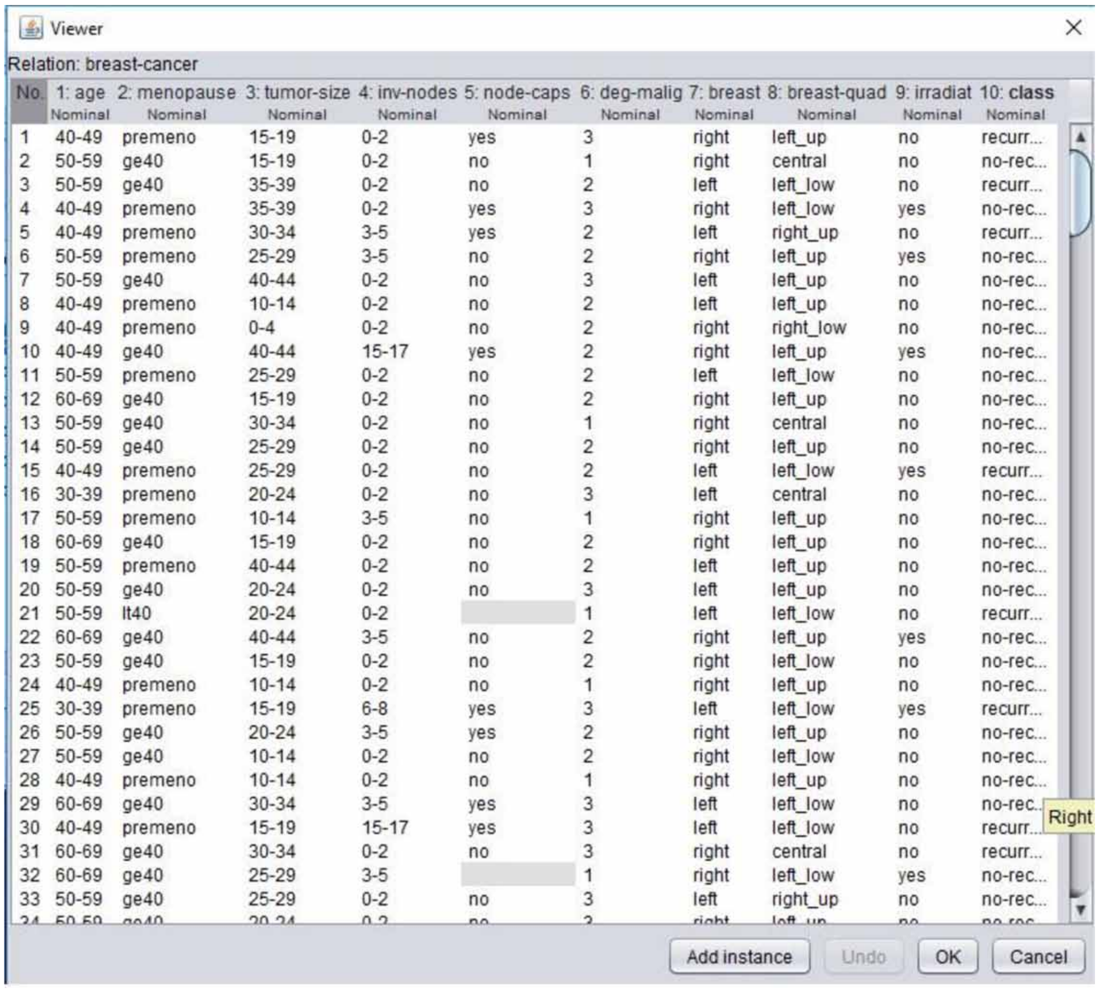
```
H:\NetBeansProjects\BigDataClassification\data\data3\breast-cancer.arff
H:\NetBeansProjects\BigDataClassification\data\data3\credit-g.arff
H:\NetBeansProjects\BigDataClassification\data\data3\iris.arff
H:\NetBeansProjects\BigDataClassification\data\data3\weather.nominal.arff
H:\NetBeansProjects\BigDataClassification\data\data3\ReutersGrain-train.arff
H:\NetBeansProjects\BigDataClassification\data\data3\soybean.arff
```

Algorithms

```
NaiveBayes
LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4
SGD -F 0 -L 0.01 -R 1.0E-4 -E 500 -C 0.001 -S 1
Stacking -X 10 -M "weka.classifiers.rules.ZeroR " -S 1 -num-slots 1 -B "weka.classifie
J48 -C 0.25 -M 2
Bagging -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0
RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
```

In Figure 6 is an example of how the Weka GUI Experimenter was used in part of the preliminary experiments. When using the Experimenter, you can add several datasets and several algorithms all at once, to analyse different performance evaluation metrics such as the accuracy and F-measure. When the Experimenter is launched, using the 'New' button at the top right corner, allows the new datasets and algorithms we intend to analyse to be added. The dataset/datasets to be analysed appears in the 'Datasets' window, while the algorithms to be analysed appear in the 'Algorithms' window. We then use the 'Run' tab at the top of the experimenter window to run the experiments, after which we use the 'Analyse' tab at the top to view different evaluation metrics we desire to use in evaluating our algorithms against the different datasets.

Figure 7. Dataset view in tabular format from the Experimenter

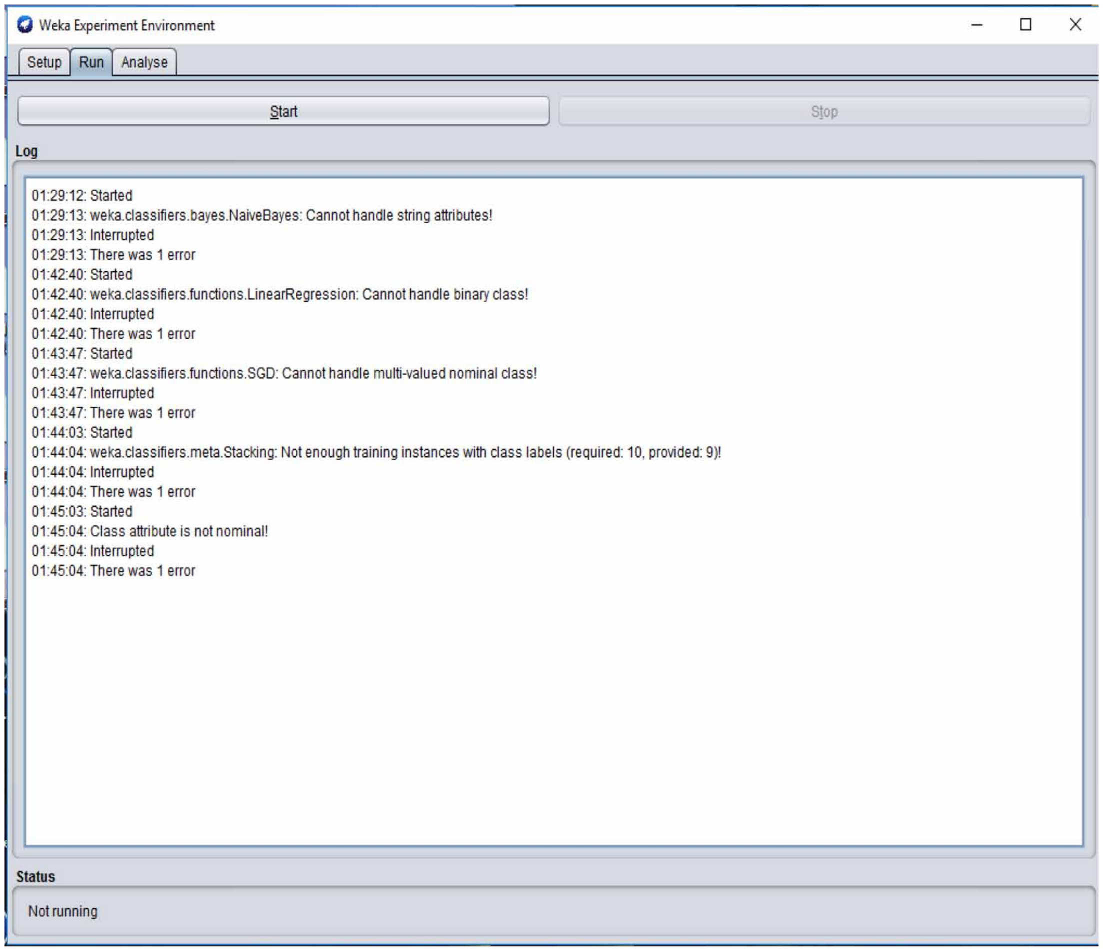


Relation: breast-cancer

No.	1: age Nominal	2: menopause Nominal	3: tumor-size Nominal	4: inv-nodes Nominal	5: node-caps Nominal	6: deg-malig Nominal	7: breast Nominal	8: breast-quad Nominal	9: irradiat Nominal	10: class Nominal
1	40-49	premeno	15-19	0-2	yes	3	right	left_up	no	recurr...
2	50-59	ge40	15-19	0-2	no	1	right	central	no	no-rec...
3	50-59	ge40	35-39	0-2	no	2	left	left_low	no	recurr...
4	40-49	premeno	35-39	0-2	yes	3	right	left_low	yes	no-rec...
5	40-49	premeno	30-34	3-5	yes	2	left	right_up	no	recurr...
6	50-59	premeno	25-29	3-5	no	2	right	left_up	yes	no-rec...
7	50-59	ge40	40-44	0-2	no	3	left	left_up	no	no-rec...
8	40-49	premeno	10-14	0-2	no	2	left	left_up	no	no-rec...
9	40-49	premeno	0-4	0-2	no	2	right	right_low	no	no-rec...
10	40-49	ge40	40-44	15-17	yes	2	right	left_up	yes	no-rec...
11	50-59	premeno	25-29	0-2	no	2	left	left_low	no	no-rec...
12	60-69	ge40	15-19	0-2	no	2	right	left_up	no	no-rec...
13	50-59	ge40	30-34	0-2	no	1	right	central	no	no-rec...
14	50-59	ge40	25-29	0-2	no	2	right	left_up	no	no-rec...
15	40-49	premeno	25-29	0-2	no	2	left	left_low	yes	recurr...
16	30-39	premeno	20-24	0-2	no	3	left	central	no	no-rec...
17	50-59	premeno	10-14	3-5	no	1	right	left_up	no	no-rec...
18	60-69	ge40	15-19	0-2	no	2	right	left_up	no	no-rec...
19	50-59	premeno	40-44	0-2	no	2	left	left_up	no	no-rec...
20	50-59	ge40	20-24	0-2	no	3	left	left_up	no	no-rec...
21	50-59	lt40	20-24	0-2		1	left	left_low	no	recurr...
22	60-69	ge40	40-44	3-5	no	2	right	left_up	yes	no-rec...
23	50-59	ge40	15-19	0-2	no	2	right	left_low	no	no-rec...
24	40-49	premeno	10-14	0-2	no	1	right	left_up	no	no-rec...
25	30-39	premeno	15-19	6-8	yes	3	left	left_low	yes	recurr...
26	50-59	ge40	20-24	3-5	yes	2	right	left_up	no	no-rec...
27	50-59	ge40	10-14	0-2	no	2	right	left_low	no	no-rec...
28	40-49	premeno	10-14	0-2	no	1	right	left_up	no	no-rec...
29	60-69	ge40	30-34	3-5	yes	3	left	left_low	no	no-rec...
30	40-49	premeno	15-19	15-17	yes	3	left	left_low	no	recurr...
31	60-69	ge40	30-34	0-2	no	3	right	central	no	recurr...
32	60-69	ge40	25-29	3-5		1	right	left_low	yes	no-rec...
33	50-59	ge40	25-29	0-2	no	3	left	right_up	no	no-rec...

An advantage of using the Experimenter in this experiment stage is that we are able to get a view of the dataset as seen in Figure 7, via clicking on any of the dataset in the 'Datasets' view as seen in Figure 3.6. From this, we can easily see for example, the attribute type of our class, or how many numeric and how many nominal type attribute we have in the dataset. This way, we can determine the influence of this, when we are analysing the performances of our supervised classifiers. However, using the experimenter to run tests on several datasets at once, can give us several error messages such as is displayed in Figure 8 below.

Figure 8. Possible errors faced when running the experimenter on several datasets and algorithms



```
01:29:12: Started
01:29:13: weka.classifiers.bayes.NaiveBayes: Cannot handle string attributes!
01:29:13: Interrupted
01:29:13: There was 1 error
01:42:40: Started
01:42:40: weka.classifiers.functions.LinearRegression: Cannot handle binary class!
01:42:40: Interrupted
01:42:40: There was 1 error
01:43:47: Started
01:43:47: weka.classifiers.functions.SGD: Cannot handle multi-valued nominal class!
01:43:47: Interrupted
01:43:47: There was 1 error
01:44:03: Started
01:44:04: weka.classifiers.meta.Stacking: Not enough training instances with class labels (required: 10, provided: 9)!
01:44:04: Interrupted
01:44:04: There was 1 error
01:45:03: Started
01:45:04: Class attribute is not nominal!
01:45:04: Interrupted
01:45:04: There was 1 error
```

Not running

From Figure 8 above, the last error that says 'Class attribute is not nominal' is as a result of the experimenter trying to run a classification algorithm that can only work when the class attribute is nominal. Although we can tell there is an error, it is hard to tell which of the data inputs caused this. The user, will then have to go back and spend time working out which dataset must have caused the error (i.e. in a case of multiple datasets to multiple algorithms. This limitation forms a part of the motivation for this research in question. This kind of error was resolved by running the experiments in parts, investigating and harnessing general knowledge about the various datasets and their effects on the choice and performances of the various classification algorithm. Doing this and then afterwards writing Java codes using the Weka API to implement the findings, eliminates such errors as we shall be discussing shortly in the results section of the next chapter. The experimenter also provides us the ability to save the results into a CSV file for further analysis.

3.4. Preliminary Experiment Results

When evaluating the performance of various algorithms, it is assumed that we can combine a number of known measures for success, to correctly help and point us to choosing the best algorithm for any specific dataset. Doing this, helps us to gain more confidence in the choice we make as regarding what algorithm performed better for a particular dataset. Hence, allowing us to easily find out if another dataset with similar general features (e.g. class attribute type, number of nominal to number of numeric attributes, the size, etc.), will also choose the same ML algorithm as its best.

Using 3 Folds Cross Validation, the following evaluation measures where gathered on various datasets listed in this paper. Due to not much significantly improved results of using 10 folds, the results from the 10 folds' cross validation can be found in *Appendix 5*.

Table 2. Area Under the Curve (AUC)

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
conact-lenses	0.93	0.50	-	0.8	-	0.8	0.50	0.50	0.82	0.84
Cpu	-	-	-	-	-	-	-	-	-	-
cpuwith.vendor	-	-	-	-	-	-	-	-	-	-
creit-g	0.78	0.49	0.66	0.67	-	0.7	0.50	0.50	0.66	0.76
Diabetes	0.82	0.50	0.72	0.74	-	0.8	0.50	0.50	0.74	0.82
Glass	0.70	0.74	-	0.8	-	0.8	0.49	0.49	0.75	0.92
Ionosphere	0.93	0.83	0.82	0.88	-	0.9	0.50	0.50	0.88	0.97
iris.2D	1.00	1.00	-	1.0	-	1.0	0.49	0.49	0.98	1.00
Labor	0.98	0.87	0.86	0.96	-	0.8	0.47	0.47	0.73	0.94
reutersCorn-test	-	-	-	-	-	-	0.5	0.50	-	-
reuersCorn-train	-	-	-	-	-	-	0.5	0.50	-	-
segent-challenge	1.00	0.72	-	1.0	-	1.0	0.50	0.50	1.00	1.00
Soybean	1.00	1.00	-	1.0	-	0.9	0.50	0.50	0.97	1.00
supermarket	0.50	0.50	0.50	0.50	-	0.5	0.50	0.50	0.50	0.50
unbalanced	0.59	0.50	0.50	0.65	-	0.6	0.50	0.50	0.50	0.70
Vote	0.97	0.95	0.95	0.98	-	0.9	0.50	0.50	0.98	0.99
weather.nominal	0.82	0.50	0.63	0.78	-	0.7	-	0.4	0.67	0.84
weather.numeric	0.80	0.50	0.53	0.69	-	0.7	-	0.4	0.63	0.64

Table 2 above, shows the area under the ROC curve, estimated for the various algorithms per dataset. Where a '- 'is observed means that the supervised algorithm, was unsuitable for that dataset. While a 'None' observation means that no ROC curve is produced given when that algorithm was used on that dataset.

Figure 9. Area Under ROC (AUC)

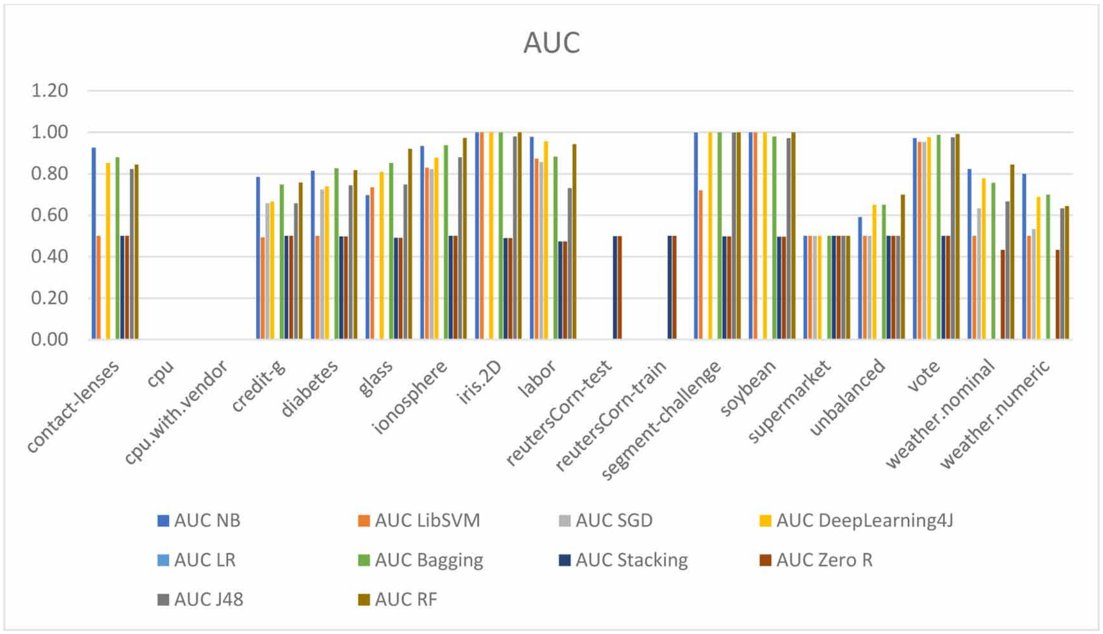


Table 3. F-Measure for datasets per algorithm

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
conact-lenses	0.83	0.77	-	0.8	-	0.5	0.77	1.00	0.89	0.83
Cpu	-	-	-	-	-	-	-	-	-	-
cpuwith.vendor	-	-	-	-	-	-	-	-	-	-
creit-g	0.83	0.82	0.84	0.77	-	0.8	0.82	0.82	0.80	0.84
Diabetes	0.82	0.79	0.83	0.76	-	0.8	0.79	0.79	0.79	0.82
Glass	0.23	0.64	-	0.6	-	0.7	0.52	0.52	0.67	0.78
ionosphere	0.86	0.88	0.90	0.81	-	0.9	0.78	0.78	0.93	0.95
iris.2D	1.00	1.00	-	1.0	-	1.0	0.32	0.32	0.98	1.00
Labor	0.96	0.84	0.89	0.84	-	0.8	0.79	0.79	0.84	0.90
reutersCorn-test	-	-	-	-	-	-	0.9	0.98	-	-
reuersCorn-train	-	-	-	-	-	-	0.9	0.99	-	-
segent-challenge	0.96	0.61	-	0.9	-	0.9	0.27	0.27	0.99	1.00
Soybean	1.00	1.00	-	1.0	-	0.8	0.24	0.24	0.90	1.00
supermarket	0.78	0.53	0.78	0.35	-	0.7	0.78	0.78	0.78	0.78
unbalanced	0.96	0.99	0.99	0.99	-	0.9	0.99	0.99	0.99	0.99
Vote	0.92	0.96	0.96	0.96	-	0.9	0.76	0.76	0.97	0.98
weather.nominal	0.84	0.78	0.71	0.75	-	0.8	-	0.7	0.67	0.78
weather.numeric	0.82	0.78	0.67	0.78	-	0.7	-	0.7	0.76	0.78

From Figure 9 above, it is expected that the AUC for choosing the best performing algorithm given a dataset, should be the figure closest to 1.

Table 3 above, gives us the F-measure estimated for each dataset per algorithm.

From Figure 10 above, it is expected that an F-Measure score closer to 1 is more desirable for any given dataset.

Figure 10. F-Measure for each dataset against several classification algorithms

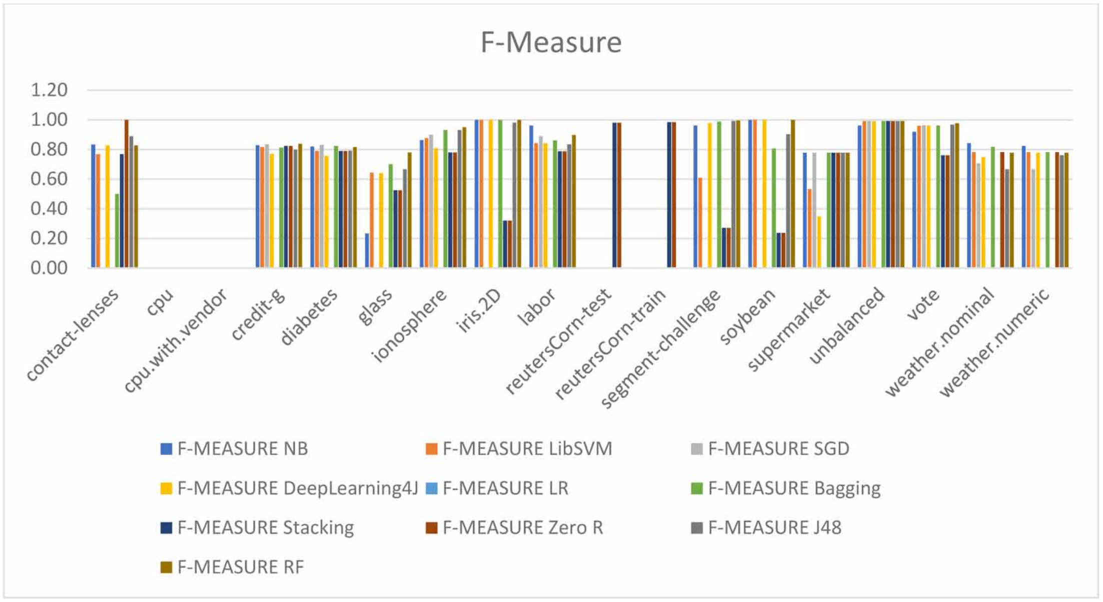


Figure 11 and Figure 12 are graphs derived from using the figures in the above Table 4. Figure 12 shows the plots of the MAE for the Cpu and Cpu.with.vendor datasets. Since the variance of the values in these dataset makes them different from other datasets, their mean absolute error range also differs. Hence, the need to plot this separately from the MAE for the other datasets. It is expected that the MAE closest to 0 is more desirable for any of the given datasets.

In Table 5 above, the values which are displayed on a scale of 0-100 describes the accuracy of the models in terms of the correctly classified data instances of the dataset. While values in the range of 0-1 (i.e. values for Cpu and Cpu.With.Vendor datasets) represents the correlation coefficients. Since the variance of the values in the 'Cpu' and 'Cpu.With.Vendor' datasets make them different from other datasets, they do not return any measure for the percentage of accurately classified instances. However, they return a *correlation coefficient which gives us an estimate of how closely related the estimated value (predicted using the model built from a particular algorithm) is from the real value*. Correlation Coefficient ranges from 0-1 with 1 meaning that the estimated value is the same as the real value. An accuracy of 100% means that the estimated values are 100% correct. Hence, it can be assumed that using the correlation coefficient in the absence of an accuracy measure to compare the accuracy of different algorithms on a dataset is possible. This is the reason why Table 5 displays both. There was however a need to plot this separately on two different plots because of the scale differences.

Table 4. Table of the Mean Absolute Error (MAE) for the various datasets

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
conact-lenses	0.24	0.25	-	0.1	-	0.3	0.37	0.37	0.15	0.22
cpu	-	-	-	20627	43.79	35.73	96.35	96.36	-	26.7
cpu.with.vendor	-	-	-	19648	35.40	29.67	87.64	87.64	-	14.4
credit-g	0.30	0.31	0.25	0.33	-	0.3	0.42	0.42	0.34	0.35
diabetes	0.28	0.35	0.23	0.33	-	0.3	0.45	0.45	0.31	0.31
glass	0.16	0.10	-	0.1	-	0.1	0.21	0.21	0.10	0.10
ionosphere	0.17	0.08	0.14	0.13	-	0.1	0.46	0.46	0.10	0.14
iris.2D	0.03	0.03	-	0.0	-	0.0	0.45	0.49	0.06	0.03
labor	0.08	0.11	0.14	0.12	-	0.2	0.46	0.46	0.31	0.23
reutersCorn-test	-	-	-	-	-	-	0.0	0.08	-	-
reuersCorn-train	-	-	-	-	-	-	0.0	0.06	-	-
segent-challenge	0.06	0.15	-	0.0	-	0.0	0.24	0.24	0.02	0.02
soybean	0.01	0.01	-	0.0	-	0.0	0.10	0.10	0.02	0.03
supermarket	0.46	0.64	0.36	0.48	-	0.4	0.46	0.46	0.46	0.46
unbalanced	0.09	0.01	0.01	0.02	-	0.0	0.03	0.03	0.03	0.03
vote	0.10	0.05	0.04	0.05	-	0.0	0.47	0.47	0.06	0.07
weather.nominal	0.37	0.36	0.36	0.32	-	0.4	-	0.4	0.37	0.33
weather.numeric	0.38	0.36	0.43	0.32	-	0.4	-	0.4	0.40	0.40

Figure 11. Mean Absolute Error (MAE) 0-1

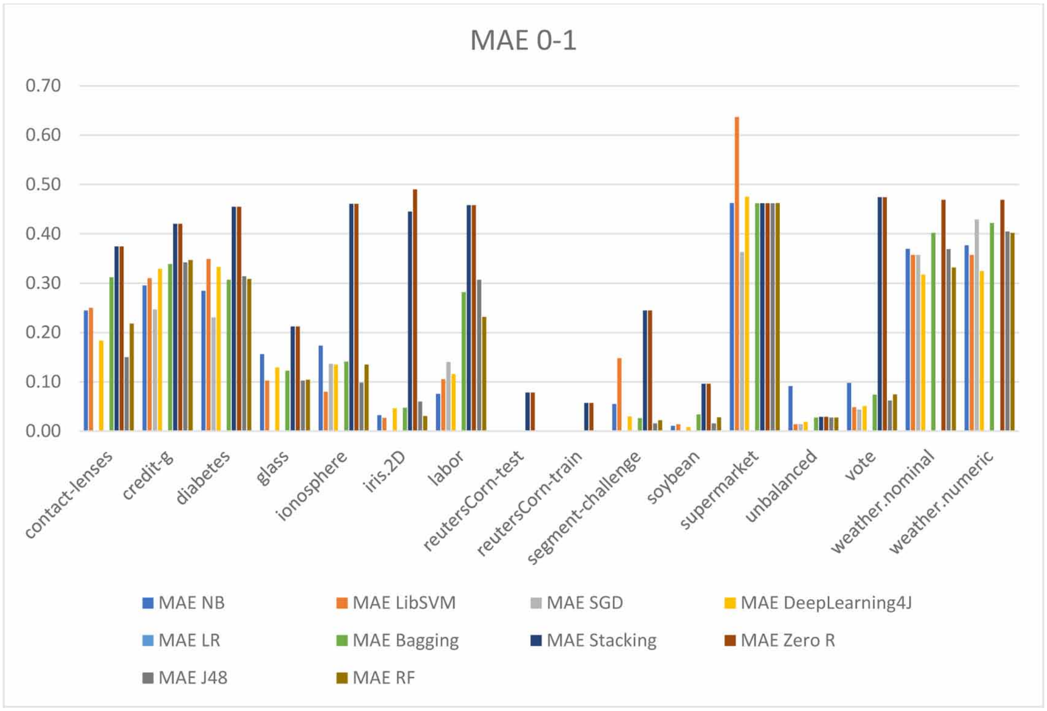


Figure 12. Mean Absolute Error (MAE) for Cpu and Cpu.with.vendor Datasets

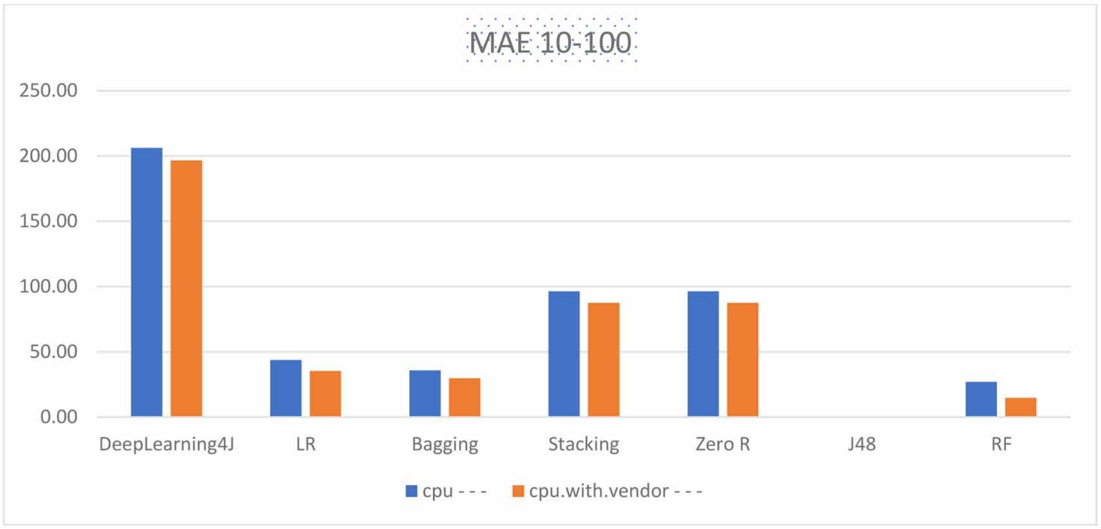


Table 5. Accuracy in % and Correlation Coefficients for Cpu and CPu.With.Vendor datasets

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
conact-lenses	83.33	62.50	-	75.0	-	70.3	62.50	62.50	87.50	79.17
cpu	-	-	-	-0.1	0.89	0.88	-0.11	-0.11	-	0.9
cpu.with.vendor	-	-	-	-0.1	0.93	0.90	-0.11	-0.11	-	0.9
credit-g	75.00	69.00	75.30	67.40	-	72.0	70.00	70.00	70.50	75.50
diabetes	75.91	65.10	76.95	68.36	-	76.7	65.10	65.10	73.31	75.39
glass	44.39	64.02	-	57.1	-	71.0	35.51	35.51	67.76	78.04
ionosphere	83.48	92.02	86.33	86.61	-	91.7	64.10	64.10	90.88	93.45
iris.2D	96.00	96.00	-	94.7	-	96.0	32.00	32.00	93.33	97.33
labor	94.74	89.47	85.96	89.47	-	80.0	64.91	64.91	77.19	85.96
reutersCorn-test	-	-	-	-	-	-	96.3	96.03	-	-
reuersCorn-train	-	-	-	-	-	-	97.0	97.10	-	-
segent-challenge	81.00	48.13	-	90.0	-	95.7	15.73	15.73	95.13	97.33
soybean	92.39	86.82	-	92.9	-	81.0	13.47	13.47	89.75	91.95
supermarket	63.71	36.29	63.71	54.59	-	63.1	63.71	63.71	63.71	63.71
unbalanced	92.52	98.60	98.60	98.13	-	98.0	98.60	98.60	98.60	98.60
vote	90.57	95.17	95.63	95.40	-	95.0	61.38	61.38	96.09	97.01
weather.nominal	78.57	64.29	64.29	71.43	-	71.3	-	64.9	64.29	71.43
weather.numeric	78.57	64.29	57.14	71.43	-	64.9	-	64.9	64.29	71.43

The graphs plotted in Figure 3 and Figure 13 below is used to represent the data from Table 5.

From Figure 13 above, it is expected that the Accuracy closest to 100% is more desirable for any given dataset

Figure 13. Shows the accuracy in percentage (0-100%) of various classification models on a variety of datasets

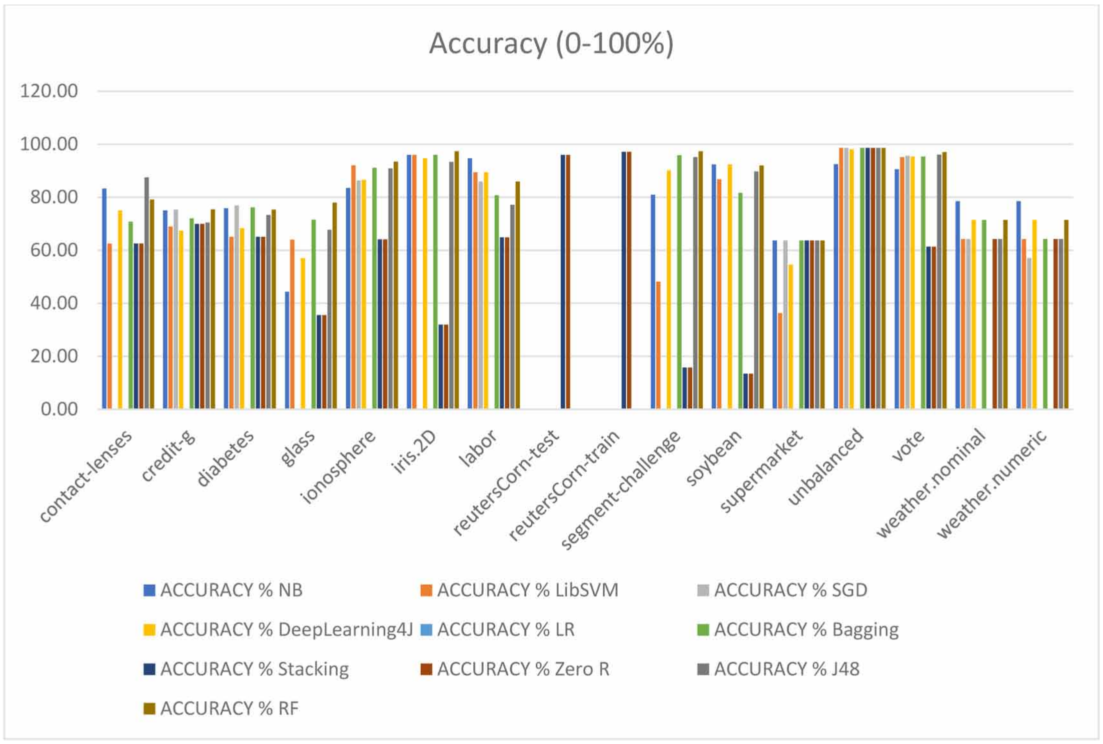
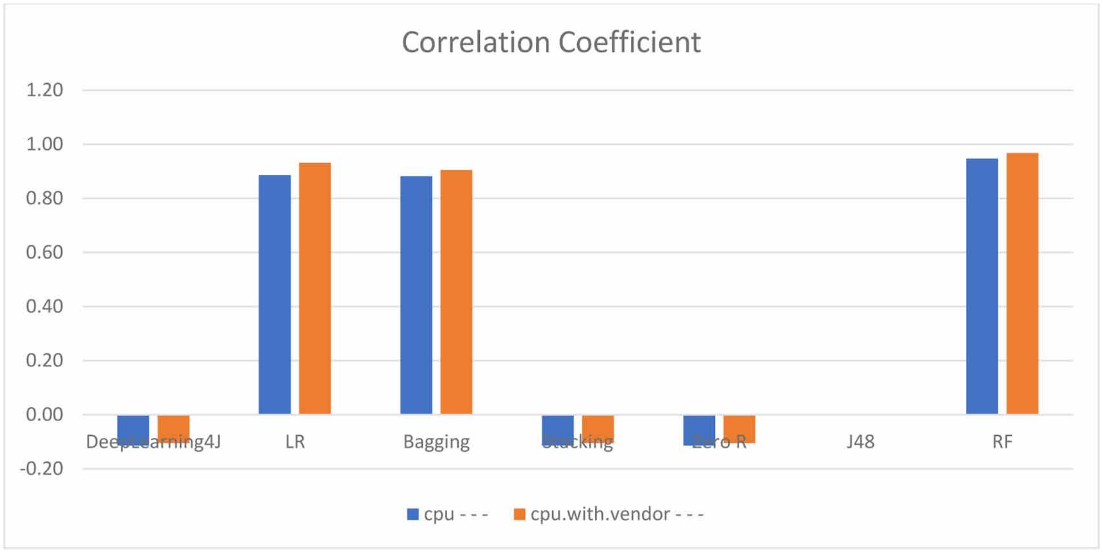


Figure 14. Correlation Coefficient 0-1 (Cpu and Cpu.With.Vendor Datasets)



From Figure 14 above, it is expected that the Correlation Coefficient closest to 1 or -1 (in an inverse correlation), is more desirable for the given datasets.

Table 6. Combination of Evaluation Measures on each dataset to effectively evaluate performance of each algorithm on different algorithms, in order to understand the patterns

Dataset	AUC	F-Measure	MAE	Accuracy	Overall
contact-lenses	NB	ZeroR	J48	J48	J48
cpu	-	-	RF	RF	RF
cpu.with.vendor	-	-	RF	RF	RF
credit-g	NB	RF/SGD	SGD	RF/SGD	SGD
diabetes	Bag	SGD	SGD	SGD	SGD
glass	RF	RF	LibSVM/J48/RF	RF	RF
ionosphere	RF	RF	LibSVM	RF	RF
iris.2D	NB/LibSVM/DL4J/Bag/RF	NB/LibSVM/DL4J/Bag/RF	LibSVM/RF	RF	RF
labor	NB	NB	NB	NB	NB
reutersCorn-test	ZR/Stack	ZR/Stack	ZR/Stack	ZR/Stack	ZR/Stack
reutersCorn-train	ZR/Stack	ZR/Stack	ZR/Stack	ZR/Stack	ZR/Stack
segment-challenge	RF/Bag & NB/J48/DL4J	RF/J48/Bag	J48	RF	RF/J48
soybean	NB/LibSVM/DL4J/RF	NB/LibSVM/DL4J/RF	NB/LibSVM/DL4J	NB/DL4J	NB
supermarket	Any	Any Except libSVM/DL4J	SGD	Any Except libSVM/DL4J	SGD
unbalanced	RF	Any Except NB	LibSVM/SGD	Any Except NB/DL4J	RF/SGD
vote	RF/Bag	RF	SGD	RF	RF
weather.nominal	RF	NB	DL4J	NB	NB
weather.numeric	NB	NB	DL4J	NB	NB

In Table 6, a multiple of evaluation measures are combined to determine the overall most desirable algorithm for each given dataset. The overall most desirable algorithm for each dataset is assumed to be the one that occurred more as the best when the different evaluation measures are considered separately. For example, the AUC analysis showed that the NB model was the best for the ‘contact-lenses’ dataset, while the F-measure, MAE and Accuracy analysis showed the J48 as the best. Combining these, and given the fact that the difference shown in the AUC for the J48 was not so significantly smaller than that of the NB, it is concluded that the J48 algorithm is overall the most desirable amongst the others for the ’contact-lenses’ dataset.

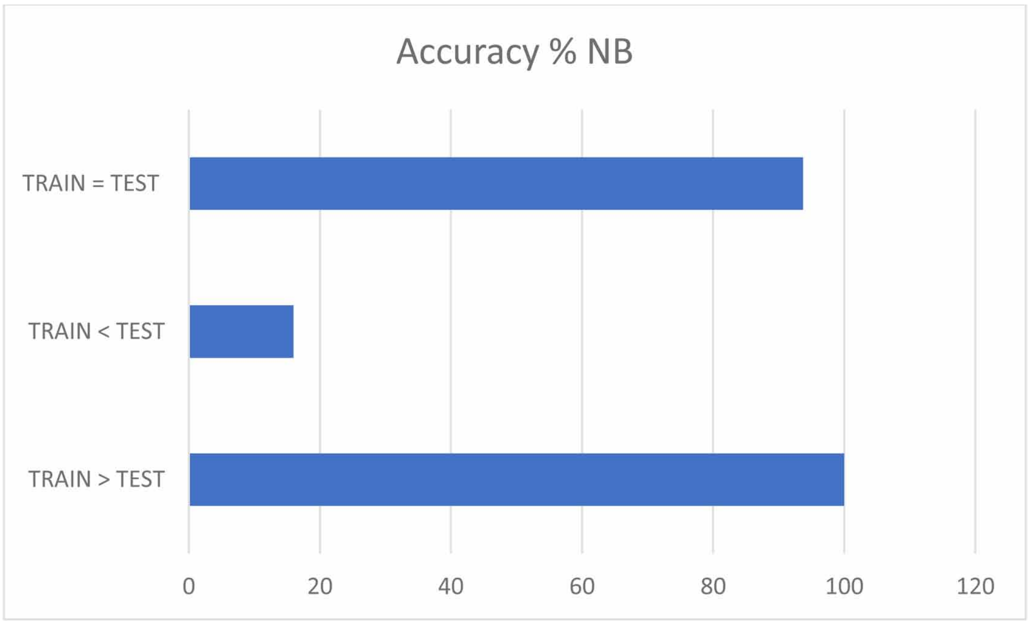
3.5. Size Effect Experiment on an Example Classification Problem

In a given scenario, where a training set and a separate test set are provided independently of each other. We can determine what the size influence of both datasets will have on the performance of a ML algorithm. For example, considering the Soybean and Soy test datasets, we check to see what changing the size of each will have on the performance (accuracy) of a Naïve Bayes classification algorithm. From doing this, the following results were obtained.

Table 7. The effect of the Train and Test Sizes on a Naïve Bayes Classifier (% Accuracy)

Size Comparison	NB (% Accuracy)
TRAIN > TEST	100
TRAIN < TEST	15.959
TRAIN = TEST	93.7042

Figure 15. Size Effect on Accuracy (%)



From Figure 15 above, we can clearly observe that when the training dataset supplied is relatively larger (at least in a ratio of 1:25 for example) than the test dataset supplied, the Naïve Bayes ML algorithm gave us a 100% performance as opposed to if it was smaller than the test dataset. On the other hand, using a train dataset that is equal to a test dataset gives a high performance, but this may not be the best performance that the algorithm can achieve. This simple experiment performed several times with varying dataset and varying algorithms one after the other, gave the same observations which showed that the size of the training data supplied matters a lot when building supervised machine learning models. Thus proving *hypothesis 1* from section 3.1.3 to be true.

4. MACHINE LEARNING ALGORITHMS CONSIDERED

The reason for using and considering these algorithms in the initial experiments, assumes that they are very popular in the data mining research community. It was decided that at least a minimum of two algorithms from the very common categories of data classification algorithms from the literature review section in this research (Section 2.2.3) is taken into consideration.

Table 8. The following algorithms from Weka where used in the experiments carried out

Algorithms considered	Category in Weka
AdaBoostM1 *	Meta
AttributeSelectedClassifier * (With BestFirst & J48)	Meta
AttributeSelectedClassifier * (With BestFirst & NB)	Meta
AttributeSelectedClassifier * (With BestFirst & RF)	Meta
AttributeSelectedClassifier * (With BestFirst & ZeroR)	Meta
AttributeSelectedClassifier * (With GreedyStepWise & J48)	Meta
Bagging *	Meta
DeepLearning4J	Deep Learner based on NN
J48 (c4.5 Decision tree)	learners (trees)
Kstar	learner (Lazy)
LibSVM	learners
Linear Regression	learners (functions)
Locally weigthed learning (LWL) *	Meta
MultiClassClassifier*(With J48)	Meta
MultiClassClassifier*(With NB)	Meta
MultiClassClassifier*(With RF)	Meta
MultiClassClassifier*(With SGD)	Meta
MultiLayerPerception	learner (functions)
NaiveBayes	learners (bayes)
RandomForest	learners (trees)
RandomSubspace *	Meta
REPTree	learner (trees)
SGD	learners (functions)
Stacking +	Ensemble
ZeroR	learners (rules)
Canopy	Clusterer
Cobweb	Clusterer
Expectation Maximization (EM)	Clusterer
FarthestFirst	Clusterer
MakeDensityBasedClusterer	Clusterer (wrapping a simpleKMeans by default)
SimpleKMeans	Clusterer
BestFirst	AttributeSelectionMethods
Greedy Stepwise	AttributeSelectionMethods
Remove UseLess	Filtering algorithm

4.1. Feature Selection and Filtering

- Best First: It is a search method used for selecting features by examining the feature subsets space by greedy hill climbing amplified with a backtracking ability. Setting the number of consecutive non-improving nodes permitted controls the level of backtracking done. Best first (Kohavi & Sommerfield, 1995) may start with the empty set of features and search forward or start with the full set of features and search backward, or begin at any point and search in both directions (by seeing all possible single feature additions and deletions at a specified point). It is chosen and experimentally used with the attribute selected classifier in Weka to create a variation of the classifier.
- Greedy Stepwise: Greedy stepwise (Caruana & Freitag, 1994)makes a greedy forward or backward search through the feature subsets space. Might begin with no/all features or from a random point in the space. Stops when the addition/deletion of any remaining features results in a decrease in evaluation. Can also produce a ranked list of features by traversing the space from one side to the other and recording the order that features are selected. It is chosen and experimentally used with the attribute selected classifier in Weka to create a variation of the classifier.
- Remove Useless: a method in Weka for filtering out attributes that vary too much or do not vary at all (Hall et al., 2009).

4.2. Supervised Classifiers

- AdaBoostM1: It is a classification algorithm (Freund & Schapire, 1996) for boosting a nominal class classifier using the Adaboost M1 method. A 'nominal class' classifier simply refers to a feature label that is a non-quantitative value lacking any numerical significance e.g. 'male', 'female' etc. Only nominal class problems can be tackled. Often dramatically improves performance, but sometimes over fits. Over fitting in machine learning is when the details of a training dataset are overly learnt by a model that it affects its performance on new test data.
- Attribute Selected Classifier: The dimensionality of training and test data is minimized by feature selection before being passed on to a classifier (Shafi, et al, 2008).
- Bagging/Bootstrap Aggregation: It is a technique of applying bootstrap replicates method to a machine learning algorithm of high variance such as classification and regression trees (Breiman, 1996). It helps to reduce such variance in the base learning algorithm. It is an ensemble meta algorithm, that generates multiple versions of a predictor and uses that to obtain an aggregated predictor. There is a random partitioning of the data into subsets to minimize the variance when building the various sub models in parallel, it then uses a weighted average function to combine the single models.
- Deep Learning for Java (Dl4j): It can be downloaded and installed in Weka for classification through Weka's 'package manager' found in the 'Tools' tab of the Weka GUI. Dl4j is a current state of the art in the artificial intelligence (AI) field in which machine learning plays an important part. It is designed based on Neural networks, and allows you create deep neural nets from various shallow nets (e.g. recurrent nets, convolutional nets, etc.) when needed in a distributed environment that uses Spark and Hadoop in addition to distributed CPUs or GPUs.

- J48: This is a decision tree supervised ML algorithm. Decision tree methods have a tree like separation of the data. There are usually internal/decision nodes (labelled with the attributes of the dataset) and leaf node/class labels. Separation at each level is done using a *split criterion*. The split criterion is usually applied on each internal node to determine what the output node is (which could be another internal node or a leaf node (which is usually a class label). Decision tree methods are popular and provide human readable rules (Murthy, 1998). Two very popular decision tree algorithms are the classification and regression trees (CART) (Breiman et al., 1984; Loh, 2011) & the C4.5 algorithm (Quinlan, 1986; Quinlan, 2014). In Weka, the J48 is used to generate a pruned or unpruned $C_{4.5}$ decision tree.
- KStar: KStar (Cleary & Trigg, 1995) is an instance-based classifier. The class of a test instance is based upon the class of those training instances like it, as determined by some similarity function. It differs from other instance-based learners in that it uses an entropy-based distance function.
- Lib Support Vector Machine (LibSVM): LibSVM is an integrated tool for support vector machine classifications, regression and distribution estimation. It can be downloaded and installed in Weka for classification through Weka's 'package manager' found in the 'Tools' tab of the Weka GUI.
- Linear Regression (LR): LR is an algorithm that models the relationship between the variables of the dataset using a linear prediction function (Weisberg, 2005). It is the first type of regression analysis that has been studied and used widely in practice (Yan & Su, 2009).
- Locally Weighted Learning (LWL): LWL makes use of an instance-based algorithm to allocate instance weights which are then used by a specified weighted instances Handler. It can perform classification e.g. using naive Bayes (Hall & Pfahringer, 2002) or regression (e.g. using linear regression).
- Multi Class Classifier: This is a meta classifier for handling multi class datasets with two class classifiers. It can also apply error modifying output codes for improved accuracy.
- MultiLayer Perception: A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances. The network can be made by hand or fixed by means of a simple heuristic. The network parameters can also be supervised and changed during training time. The nodes in this network are all sigmoid (except for when the class is numeric, in which case the output nodes become non-threshold linear units).
- Naïve Bayes (NB): This is a generative probabilistic classification algorithm. It uses the Naïve Bayes hypothesis by (John & Langley, 1995), which is a simplification of the Bayes model. They are very simple, fast and commonly used amongst data classification methods (Murphy, 2006). They make use of statistical interpretation to find the best class for a given sample. Probabilistic classification algorithms will often output an equivalent posterior probability $p(C \mid x)$ for each of the possible classes a test instance may belong to (C. (Aggarwal, 2014).
  - Posterior probability = conditional probability obtained after taking into account precise features of the test case.
  - Prior probability = probability distribution of training records that belongs to each specific class.

The two basic ways that the posterior class probability is estimated:

- Through defining the class conditional probabilities $p(x \mid C)$ for each class (C), after which the prior class probability p(C) is then inferred and Bayes theorem used to determine $p(C \mid x)$.

- By modelling the joint distribution p(x,C) directly and then normalizing it to obtain the p(C | x).
    - Random forest (RF): It is a combination of various decision trees that uses the bagging method. Each tree in the forest depends on the values of a random vector with similar distribution sampled independently (Breiman, 2001).
    - Random Subspace: (Barandiaran, 1998) A decision tree based classifier that maintains highest accuracy on training dataset and improves on generalization accuracy as it develops in difficulty. The classifier contains multiple trees constructed steadily by pseudo randomly choosing subsets of components of the feature vector (i.e. trees constructed in randomly chosen subspaces).
    - REPTree: A fast decision tree learner creates a decision or regression tree with information gain or variance and trims it using reduced-error pruning with back fitting. It only sorts values for numeric features once. Omitted values are handled by splitting the resulting instances into fragments.
    - Stochastic Gradient Descent (SGD): In Weka, this is an implementation of the stochastic gradient descent function, used to learn different linear models e.g. binary class SVM, binary class logistic regression, squared loss, Huber loss and epsilon-insensitive loss linear regression. It replaces globally every missing value and does a transformation of nominal attributes to binary ones. It also normalizes all attributes, so the output coefficients are based on the normalized data. For numeric class attributes, the squared, Huber or epsilon-incentive loss function must be used (Hall et al., 2009).
    - Stacking: This is also a meta algorithm where the original training data is partitioned into various subsets to build average performing models on each subset, then combine the models using a blending technique and a logistic regression function, to minimize both the variance as well as increase the accuracy of predictions (Wolpert, 1992).
    - Zero Rules (ZeroR): This is a rule-based classification algorithm. It relies on the target variable and ignores the other features/predictors. It predicts the majority of class in the train dataset (Aher & Lobo, 2012). Although it does not have any predictability power, ZeroR is useful as a baseline performance benchmark for other classification methods. It works by building a frequency table for the target class variable and select its most frequent value.

4.3. Unsupervised Classifiers

- Expectation Maximization (EM) clustering algorithm: This is a simple expectation maximization algorithm (Moon, 1996), for determining the maximum likelihood estimates through iterations. There is an alternation between two steps (the step where the expectation of the log likelihood is computed, and the step for computing parameters that maximizes the expected log-likelihood found in the first step (Sharma, et al, 2012). Using this algorithm will group the dataset instances into various clusters. EM assigns a probability distribution to each instance, which indicates the probability of it belonging to each of the clusters. In Weka, EM can decide how many clusters to create by cross validation or you may specify beforehand how many clusters to generate. The cross validation for determining the number of clusters is performed by first setting the number of clusters to 1, then the training set is split randomly into 10 folds, then EM is performed 10 times using the 10 folds the usual cross validation way, then the log-likelihood is averaged over all the different results, finally if the log-likelihood has increased the number of clusters is increased by 1 and a new iteration of the steps is repeated again.

- Canopy: A clustering algorithm in Weka that requires just one pass over the dataset. It can be run in either batch or incremental mode. However, the results are not as good when its used incrementally because the minimum or maximum of each numeric feature is not determined in advance (McCallum, et al, 2000).
- Cobweb: Algorithm that implements the cobweb and classit clustering algorithms. It mostly compares the best host, new leaf adding, merge of the two best hosts and splitting of then a split of the best host when deciding where to cluster a new instance (Fisher, 1987; Gennari, et al, 1989).
- Farthest First: It is used as a fast simple approximate clusterer that enables the dataset to learn of discover something for itself. Based usually on the Farthest First algorithm which is first discussed in (Hochbaum & Shmoys, 1985).

4.4. Evaluation Measures

Cross Validation: This is one of the model evaluation techniques used in this research. Hold-Out Validation method is a statistical method that requires the dataset to be split into two segments (one for training the classifier and one for testing the classifier). The training data set is usually larger than the test data set. A disadvantage of this method is that the test is performed on a smaller portion of the data, thus increasing the tendency for false accuracy measurements (Aggarwal, 2014). To address the problems of the hold out method, a more logical approach to the hold out method was developed. It is known as the cross validation method (Refaeilzadeh et al., 2009). It involves the data being split equally and the hold-out evaluation method is performed two times by using the training data set from the first iteration as the test data set in the second iteration and vice versa. The simple form of the cross validation is the k-fold cross validation.

Supplying a test set: Another model evaluation method used in this research. As opposed to using the k-fold cross validation method to analyse the models built, the method of supplying a separate test dataset is provided as an option to the user of the system. Also carried out some performance evaluation using: accuracy of the correctly classified instances as discussed in equation (2.1), recall from equation (2.3), precision from equation (2.4), specificity from equation (2.5), fall-out from equation (2.6) and the f-score (or f-measure) expressed in equation (2.7).

Correlation Coefficient: This tells us how much the true value of interest and the predicted value are related. Its value is usually between -1 and 1, with 0 meaning there is no relationship at all. This Statistical function is only displayed and used as an evaluation measure when reporting numeric class predictions.

Mean absolute error from equation (2.2): As the average distance the model predictions are from the actual data points. The predictions below data points are not treated as negative distances. This evaluation method is reported for both nominal or numeric predictions. The Root Mean Square Error, Relative Absolute Error and Root Relative squared error, are also general estimates that are displayed and can be used to compare the true values to their predicted values.

5. PROBLEM IDENTIFICATION THROUGH EXPERIMENTS

When running the experiments on the different datasets using Weka, the following problems were encountered and identified:

1. A classifier trained using a labelled dataset was not necessarily suitable for the next dataset. Which means that it is therefore important, that one of the aims of this research which is 'to help us automatically select the best machine learning method and algorithm to use on a particular dataset by implementing and transferring knowledge' will help us resolve this problem.
2. Despite the advantages of the experimenter and knowledge flow in Weka. During the pre experimentations carried out in section 3.2 issues/errors were often encountered when automatically trying to apply several algorithms to multiple datasets from different sources while will cause the model building process to fail. Which we do not want to happen when we have data from various sources requiring classification or clustering.
3. To avoid the problem in 2 above, the user has to manually spend a lot of time analysing the dataset and available algorithms, then perform multiple trial and error experiments on one dataset at a time. This problem is resolved by this research, through the building of a hybrid automatic machine learning system that does not require any time wastage on trial and error but can assist the user to pass in multiple datasets and then automatically determine which algorithm is best to use on that dataset.
4. Traditional tools such as Weka are not suitable for present day multiple learning tasks. The experimenter which was the closest to use for running multiple algorithms on multiple datasets at the same time, did not provide a way to use a clustering algorithm. So, assuming one of our datasets is an unlabelled dataset, then the process also automatically fails. The system modelled in this research aims to eliminate this problem by providing an automatic decision on what learning method to adopt depending on meta information learned e.g. by answering the question 'is the data labelled or unlabelled?' at the decision node.

6. KNOWLEDGE GAINED FROM EXPERIMENTS

Some observations made from the results of performing these preliminary experiments include:

- If a set of class labels exists already and can be specified for all training instances, then supervised learning is preferred.
- For any supervised classification algorithm to perform their best, it is important to first and foremost ensure that the size of the labelled train dataset is larger than the test dataset (it is assumed in this paper, based on the experiments performed that this should be around a ratio of 1:25).
- When the number of test instances to be classified is small, Increasing the number of folds increases the accuracy of random forest with nominal data (only by a non-significant difference though if the train data set is large).
- Increasing the number of folds from 3 to 10 increases accuracy of random forest with numeric data (only by a little due to a larger train dataset size used).

- Increasing the number of folds from 3 to 10 increase accuracy of random forest with mixed data (only by a little due to a larger train dataset size used).
- For random forest, when the total number of instances is really small e.g. 24 or 30, its best to use 3 folds. Increasing its number of folds only reduces its performance in such cases.
- We cannot use Naïve Bayes for numeric dataset, and it is very important to train the autoML system designed with these limitations by default.
- Increasing folds from 3 to 10 for NB will improve the accuracy (only a little but the time taken to build the model is much faster than RF) for a large train dataset.
- For Naive Bayes, it is best to use 3 folds if the dataset for training is really small.
- Unsupervised learning is preferable if no pre-existing class label exists,
- Unsupervised learning is preferable if the training set is way smaller than the sample set to be tested.
- When the class attribute type is 'numeric', use the RF algorithm.
- When the class attribute is 'nominal', and all other attributes are nominal and the total number of attributes are less than 10, and the number of instances are less than 50 with missing values <1% in total, then use the J48.
- When the class attribute type is 'nominal', but the other attributes contain 'String' type attributes, then use the ZeroR or Stacking algorithm.
- When the class attribute type is 'nominal', but we have at least half as many numeric attributes as there are nominal (i.e. the ratio of numeric to nominal is close to the scale of 1:2), then use the RF algorithm.
- When the class attribute is 'nominal', and the total number of attributes are less than 10 with all other attributes as 'numeric', and there are no missing values, and the total number of instances are greater than 500, then using the SGD algorithm is favourable.
- When the class attribute type is 'nominal', and the total number of instances are less than 500, and we have more or all other attributes as 'numeric', then use the RF.
- When the class attribute type is 'nominal', and the number of numeric attributes to nominal attributes are not any close to a ratio of 1 to 2, then use the NB algorithm.
- When the class attribute type is 'nominal', and the total number of instances is greater than 500, and the total number of attributes is greater than 10, and we have more numeric attributes than nominal, then use RF.
- When the class attribute type is 'nominal', and the total number of instances is greater than 500, and the total number of attributes is greater than 10, and all other attribute types are nominal, and the missing values are not up to 1% (i.e. they are <1%), then we can use NB.
- When the class attribute type is 'nominal', and the total number of attributes is greater than 100, and the total number of instances are greater than 1000, and the number of missing values are > 50%, then we can choose to use the SGD.
- Last but not the least, when the class attribute type is 'nominal', and the total number of attributes are greater than 10, and all nominal, with missing values > 1% present in the dataset, then we use the RF.

The conclusions derived from these experiments allows us to easily describe the decision learning (learning to learn) process of the auto ML system proposed as a set of Rules. Below in the following subsection, we will be discussing the Meta learning algorithm designed to this effect. As well as provide us with more details about the auto Machine Learning (autoML) system modelled in this research and from the observations listed above.

7. SUMMARY

This chapter describes and discusses a combination of research methodologies e.g. experimental, theoretical and systems design used in this research. Therefore, allowing us to eliminate as much as possible every limitation that can be encountered with the individual methods themselves. For example, experimental research methodology has a limitation because the experiments are performed mainly in a controlled environment and might not reflect properly some practices performed ‘in the wild’. But combining this with some survey and prototype (system’s) design, reduced such limitations. The knowledge gained from carrying out preliminary experimentation is used in the next following chapter to design and model the hybrid-autoML system.

REFERENCES

Aggarwal, C. C. (2014). *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC. doi:10.1201/b17320

Aher, S. B., & Lobo, L. (2012). Comparative study of classification algorithms. *International Journal of Information Technology*, *5*(2), 239–243.

Baban, S. M., Mohammed, P., Baberstock, P., Sankat, C., Boyd, W., Laukner, B., & Baban, S. M. (2009). *The Journey from Pondering to Publishing*. University of the West Indies Press.

Barandiaran, I. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8).

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. doi:10.1007/BF00058655

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. doi:10.1023/A:1010933404324

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Caruana & Freitag. (1994). Greedy attribute selection. *ICML’94: Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, 28–36.

Cleary, J. G., & Trigg, L. E. (1995). *K*: An instance-based learner using an entropic distance measure Machine Learning Proceedings 1995*. Elsevier.

Dua, D., & Karra Taniskidou, E. (2017). *UCI Machine Learning Repository*. https://archive.ics.uci.edu/ml/datasets.html

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*(2), 139–172.

Frank, E., Hall, M., & Pfahringer, B. (2002). Locally weighted naive bayes. *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*.

Freund, Y., & Schapire, R. E. (1996). *Experiments with a new boosting algorithm*. Paper presented at the ICML.

Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, *40*(1-3), 11–61.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, *11*(1), 10–18.

Hassani, H. (2017). *Research methods in computer science: The challenges and issues*. arXiv preprint arXiv:1703.04080.

Hochbaum, D. S., & Shmoys, D. B. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, *10*(2), 180–184.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*.

Kohavi, R., & Sommerfield, D. (1995). Feature subset selection using the wrapper model: Overfitting and dynamic search space topology. *The First International Conference on Knowledge Discovery and Data Mining*, 192–197.

Lars, K., Chris, T., Frank, H., Holger, H., & Kevin, L.-B. (2017). *Auto-WEKA Sample Datasets*. http://www.cs.ubc.ca/labs/beta/Projects/autoWeka/datasets/

Loh, W. Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews. Data Mining and Knowledge Discovery*, *1*(1), 14–23.

McCallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, *13*(6), 47–60.

Murphy, K. P. (2006). *Naive bayes classifiers*. University of British Columbia.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, *2*(4), 345–389.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). *Cross-validation Encyclopedia of database systems*. Springer.

Shafi, S., Hassan, S. M., Arshaq, A., Khan, M. J., & Shamail, S. (2008). Software quality prediction techniques: A comparative analysis. *4th International Conference on Emerging Technologies*, 242-246.

Sharma, N., Bajpai, A., & Litoriya, M. R. (2012). Comparison the various clustering algorithms of Weka tools. *Facilities, 4*(7).

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.

Yan, X., & Su, X. (2009). *Linear regression analysis: theory and computing*. World Scientific.

APPENDIX 1

Mini Survey Questions on the importance of Big Data Classification in practice. The questions can be found using this link https://newqtrial2015az1.az1.qualtrics.com/jfe/form/SV_6nZx2JVfovETMBT?Q_JFE=qdg However, the actual survey itself has since been closed and results analysed.

Q1. Big Data is often defined based on three properties: Volume, Variety and Velocity (known as the 3 Vs).
Have you heard of the term Big Data before now?
Yes
No

Q2. Data classification is the process of allocating data into one or more categories (see an example in the image below.
Have you heard of Data Classification before now?
Yes
No

Q3. Is classification of big data in real time a good management technique?
Yes
No
Not Sure

Q4. Do you think classifying big data will help improve data security measures in place?
Definitely yes
Probably yes
Probably not
Definitely not

Q5. Have you or the organization you work for used Big Data Classification tools before?
Yes
No
Not Sure

Q6. How do you utilize big data? (you can select more than one option)

- Manage big data
- Analyse big data
- Query big data
- Create big data
- Optimize big data
- Financial trading
- Understanding and targeting customers
- Optimizing business processes
- Personal quantification and Performance optimization
- Other

Q7. What is your job role?

Q8. Comments

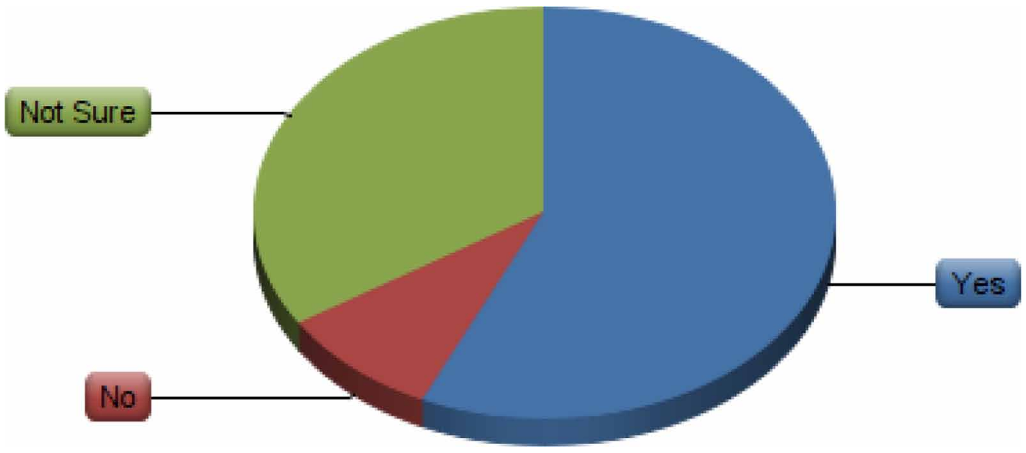
APPENDIX 2

Results from the mini survey

Out of 85% of the participants, who had previously heard about big data and 85% who had heard about data classification before?

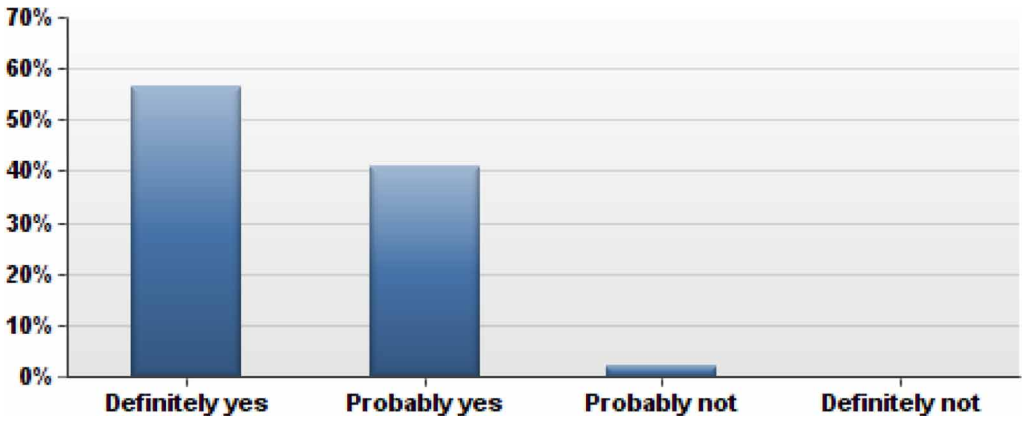
The majority thought classification of big data is a good management technique.

Figure 16. shows data from the survey carried out, that data science professionals are well aware of data classification as a good management technique



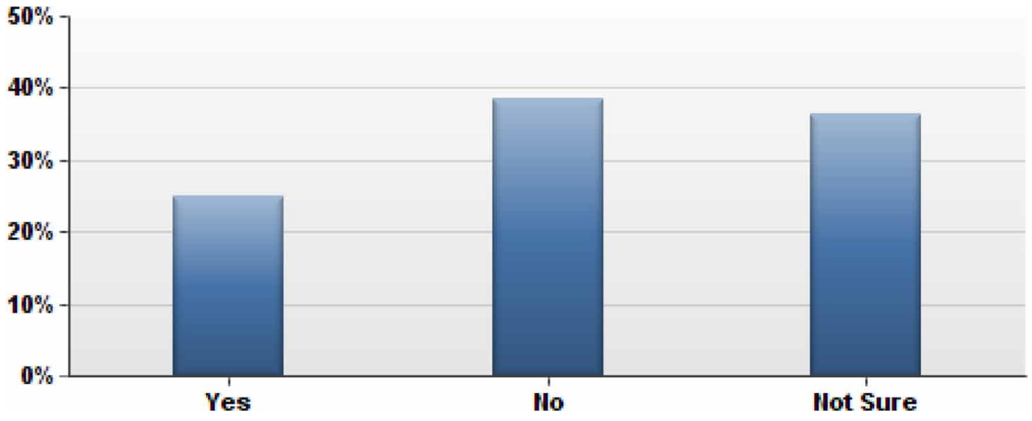
In terms of whether they think big data classification will help improve security measures in place, the majority agreed that it definitely will while about 41% said it ‘probably will’. Meaning for them, there was a high level of uncertainty.

Figure 17. Survey results, showing data science professionals thoughts on whether big data classification measures in place, effectively improves security



On the use of big data classification tools and if it has been used by them or the organization they work for, the majority said no, while many were not sure and just a few actually had.

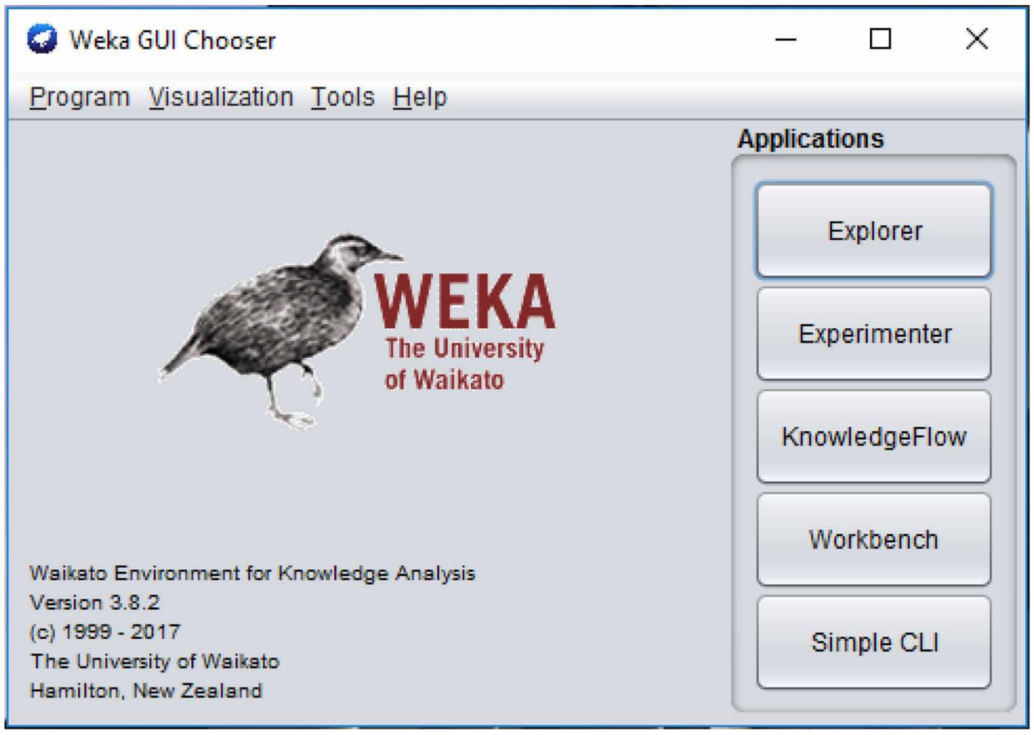
Figure 18. Survey results on the use of big data classification tools by several data science professionals



APPENDIX 3

Weka GUI

Figure 19. Weka GUI when initially launched

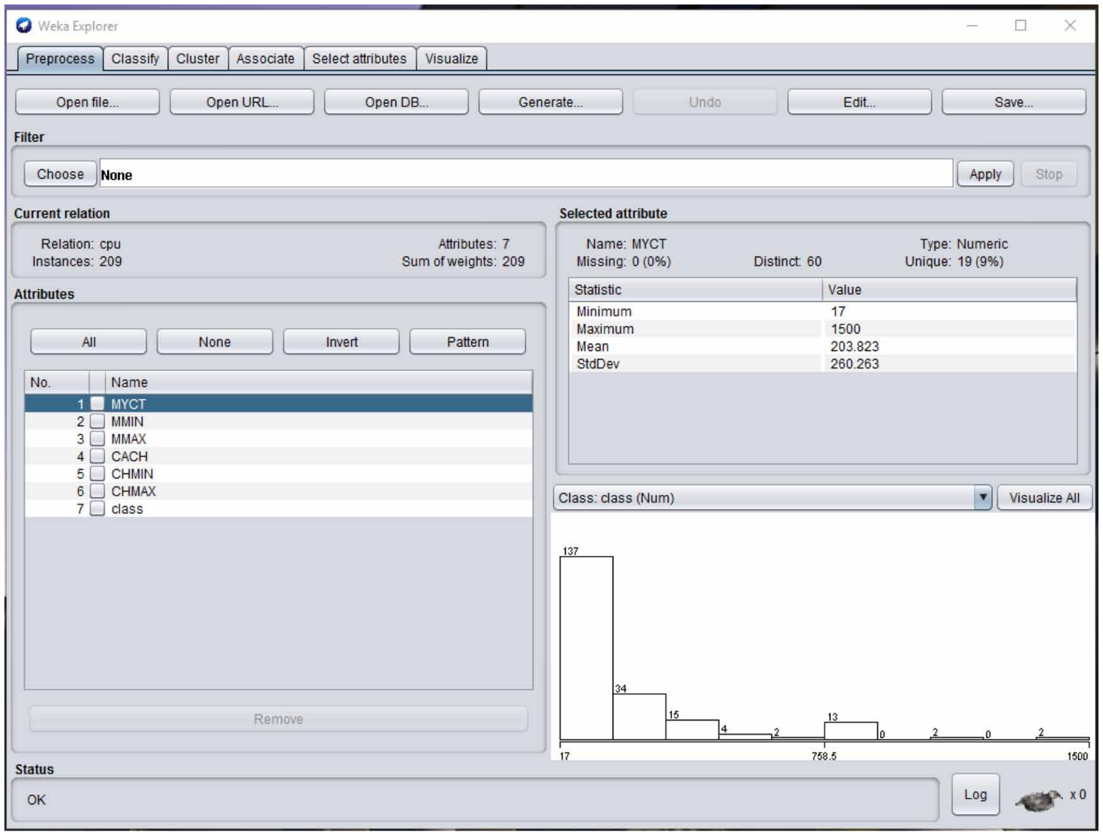


Waikato Environment for Knowledge Analysis
Version 3.8.2
(c) 1999 - 2017
The University of Waikato
Hamilton, New Zealand

The Explorer

Clicking on the ‘Explorer’ tab after launching the Weka GUI, launches the Weka explorer.

Figure 20. The Weka Explorer GUI



Filter: None

Current relation
Relation: cpu Attributes: 7
Instances: 209 Sum of weights: 209

Attributes

No.	Name
1	MYCT
2	MMIN
3	MMAX
4	CACH
5	CHMIN
6	CHMAX
7	class

Selected attribute
Name: MYCT Type: Numeric
Missing: 0 (0%) Distinct: 60 Unique: 19 (9%)

Statistic	Value
Minimum	17
Maximum	1500
Mean	203.823
StdDev	260.263

Class: class (Num)

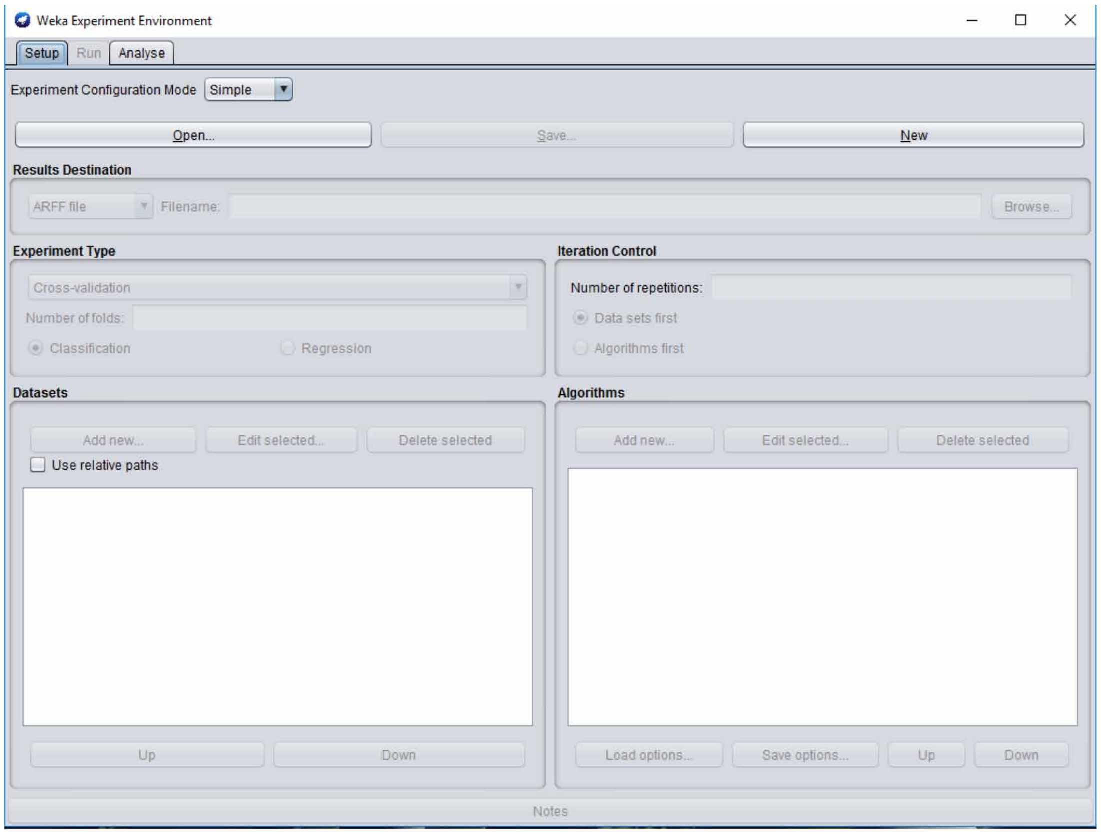
Status
OK

The Explorer lets you pre-process, visualize, classifier and cluster a dataset. It provides the options to load the dataset from a file, url, database or generate data. Once the data is loaded, the explorer will give a brief summary and visualization of the data such as the attributes listed, the name, number of attributes, etc. The pre-processing of the data using the Weka explorer can be achieved by applying one of the many filters it provides and applying this to the data. For more visualization tasks, the ‘Visualize’ tab of the explorer can be used. After pre-processing of the loaded data, the ‘Classify’ and ‘Cluster’ tabs of the explorer will supply a varying list of classification and clustering algorithms that the user can choose from for their given problem. One limitation of using the Weka explorer is the fact that the user has to process and experiment on one dataset & one algorithm at a time.

The Experimenter

Clicking on the ‘Experimenter’ tab after launching the Weka GUI, launches the Weka experimenter.

Figure 21. The Weka Experimenter GUI


Experiment Configuration Mode: Simple

Results Destination: ARFF file; Filename:

Experiment Type: Cross-validation; Number of folds:; [✓ Classification] Regression

Iteration Control: Number of repetitions:; [✓ Data sets first] Algorithms first

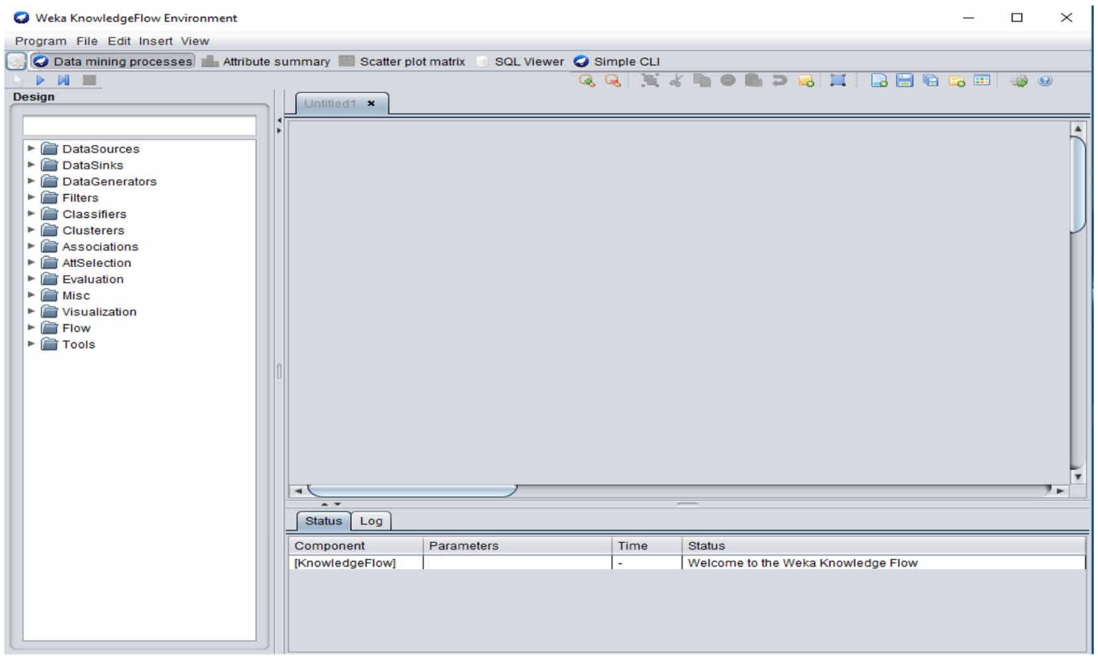
Datasets: Use relative paths

The Weka experimenter enables us to test on a trial and error basis several techniques and parameters, analyse the results to determine the most suitable technique and parameters to use. It automates this trial and error experiments for the user by allowing the user queue up multiple machine learning algorithms to run on multiple data sets, and allows for the collection of the statistical comparison of their performance against each other. Although, the experimenter eliminates to a great degree the limitations of using the explorer it is limited by the fact that if one of the algorithms in the queue is unsuitable for one of the datasets in the queue (because of the meta-features of the dataset for example), then the experiment will fail without the user knowing of identifying why it failed. This limitation can be overcome by an automated machine learning system that takes into account the meta information of the dataset and knowledge of the algorithm to automatically choose and use the suitable ones for the experiment while skipping over the unsuitable ones. This way the user gets the experiments completed successfully to the end.

The Knowledge Flow GUI

Clicking on the 'Knowledge Flow' tab after launching the Weka GUI, launches the Weka knowledge flow.

Figure 22. The Weka Knowledge Flow GUI



Component	Parameters	Time	Status
[KnowledgeFlow]		-	Welcome to the Weka Knowledge Flow

The Weka Knowledge flow gives an alternative way for using Weka in a work flow type way. It allows you build and visualise the data as flowing through from input to output phases. Just like the 'explorer', it allows you perform data mining tasks on one dataset at a time and like the 'experimenter' it can allow you run multiple algorithms on the dataset at the same time. It is sometimes more efficient than the experimenter because, it allows performing tasks on the dataset an instance at a time without the need to load the whole set in memory. Although, this is not advisable under normal circumstances because it can bring about new problems such as more time used in building a model, due to the fact that the dataset will be read one instance at a time. Also, if the experiment is interrupted because of one of the algorithms in the flow, then it gives a proper log to the user of which algorithm failed exactly with reasons for failure. The user can easily adjust the flow by simply removing that algorithm from the flow and run the experiments again. The limitation however of the 'knowledge flow' is the same limitation with the 'explorer', whereby the user can only experiment on one dataset at a time from one data source.

APPENDIX 4

Lists of most datasets used throughout this project. Subsets of this list, are referred to at different points within the main content area.

Table 9. A table summary of datasets used in this research

Dataset	# Instances	#Attributes	Class attribute type	Missing Values
contact-lenses	24	All nominal (5)	Nominal	No
cpu	209	All numeric (7)	Numeric	No
cpu.with.vendor	209	1 Nominal, 7 Numeric	Numeric	No
credit-g	1000	14 Nominal, 7 Numeric	Nominal	No
diabetes	768	8 Numeric, 1 Nominal	Nominal	No
glass	214	9 Numeic, 1 Nominal	Nominal	No
ionosphere	351	34 numeric, 1 Nominal	Nominal	No
iris.2D	150	2 Numeric, 1 Nominal	Nominal	No
labor	57	9 nom, 8 numeric	Nominal	Yes (2%)
reutersCorn-train	1554	String	Nominal	No
segment-challenge	1500	19 Numerical, 1 Nominal	Nominal	No
soybean	683	36 Nominal	Nominal	Yes (<1%)
soytest	26	36 Nominal	Nominal	No
supermarket	4627	217 nominal	Nominal	Up tp 77%
unbalanced	856	32 numerical, 1 Nominal	Nominal	No
vote	435	17 nominal	Nominal	Yes (3%)
weather.nominal	14	5 nominal	Nominal	No
weather.numeric	14	2 Numeric, 3 Nominal	Nominal	No
Dexter	420	20001 Numeric	Numeric	No
Dorothea	805	100000 Numeric	Numeric	No
Yeast	1039	8 Numeric, 1 Nominal	nominal	No
Amazon	1050	10001 numeric	Nominal	No
Secom	1097	591 nominal	Nominal	Yes (5%)
Semeion	1116	256 numeric 1 nominal	Nominal	No
Car	1209	7 nominal	Nominal	No
Madelon	1820	500 numeric 1 nominal	Nominal	No
KR-VS-KP	2238	37 nominal	nominal	No
Abalone	2923	2 nominal, 7 numeric	nominal	No
Wine Quality	3429	11 numeric, 1 nominal	Nominal	No
Waveform	3500	40 numeric, 1 nominal	Nominal	No
Gisette	4900	5000 numeric 1 nominal	Nominal	No
Convex	8000	784 numeric, 1 nominal	Nominal	No
Cifar-10-small	10000	3072 numeric, 1 nominal	Nominal	No
Mnist Basic	12000	784 numeric, 1 nominal	Nominal	No
Shuttle	43500	9 numeric, 1 nominal	Nominal	No
KDD09-Appentency	35000	192 numeric 39 nominal	Nominal	Yes(99%)
Cifar-10	50000	3072 numeric 1 nominal	Nominal	No

APPENDIX 5

Table 10. Area Under Curve using 10-folds cross validation

AUC										
Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
conact-lenses	0.95	0.50	-	0.8	-	0.9	0.22	0.22	0.95	0.86
cpu	-	-	-	-	-	-	-	-	-	-
cpuwith.vendor	-	-	-	-	-	-	-	-	-	-
creit-g	0.79	0.49	0.69	0.69	-	0.7	0.50	0.50	0.64	0.79
diabetes	0.82	0.50	0.73	0.77	-	0.8	0.50	0.50	0.75	0.89
glass	0.72	0.80	-	0.8	-	0.8	0.47	0.47	0.77	0.93
ionosphere	0.94	0.91	0.84	0.87	-	0.9	0.49	0.49	0.89	0.98
iris.2D	1.00	1.00	-	1.0	-	1.0	0.50	0.50	0.99	1.00
labor	0.97	0.91	0.90	0.97	-	0.8	0.47	0.47	0.70	0.94
reutersCorn-test	-	-	-	-	-	-	0.4	0.45	-	-
reuersCorn-train	-	-	-	-	-	-	0.4	0.47	-	-
segent-challenge	1.00	0.76	-	1.0	-	1.0	0.49	0.49	1.00	1.00
soybean	0.99	1.00	-	1.0	-	0.9	0.49	0.49	1.00	1.00
supermarket	0.50	0.50	0.50	0.50	-	0.5	0.50	0.50	0.50	0.50
unbalanced	0.56	0.50	0.50	0.70	-	0.5	0.43	0.43	0.43	0.72
vote	0.97	0.96	0.97	0.98	-	0.9	0.49	0.49	0.97	0.99
weather.nominal	0.58	0.50	0.69	0.69	-	0.3	0.18	0.18	0.63	0.53
weather.numeric	0.44	0.50	0.49	0.53	-	0.1	0.48	0.18	0.79	0.44

Chapter 11
Hybrid-AutoML System Development

ABSTRACT

This chapter presents the Hybrid-AutoML system requirements, design materials, model algorithms, and model design, which encompasses the design goals, architecture (a three-layered architecture), components, and characteristics of the Hybrid-AutoML toolkit developed in this research for automatic mode and model selection on single or multi-varying datasets. The mode components, decision learning and AutoProbClass unsupervised algorithms, and application API are described. The testing and evaluation of the model is conducted by two case studies.

1. INTRODUCTION

This research models a hybrid classification system architecture comprising of three different layers. The second layer which is a decision learning level, automates the decision-making process on what learning method to adopt at any point in time, given a heterogeneously large input of data sets. The decision-making process is a Meta-learning (learning to learn) process. This research presents a hybrid decision learning concept that uses more general knowledge about supervised and unsupervised machine learning algorithms and some meta features of the data. Based on the performance results of the preliminary experiments in section 3.2.4, a set of decision rules are drawn to enable the decision learning process, which further helped in achieving automatic classification of big data. Also, a self-evolving auto unsupervised classification algorithm which is suitable to use automatically in the absence of large labelled datasets is designed and developed in this Section.

DOI: 10.4018/978-1-7998-7316-7.ch011

2. SYSTEM REQUIREMENTS

1. The system is a tool for the classification of big data automatically, by invoking either a supervised machine learning algorithm, unsupervised machine learning algorithm or semi-supervised learning algorithm, depending on the existing state of the data set and the scenario.
2. The system accepts as input data of varying types and from different domains.
3. System check is performed to determine if some knowledge about the data set is known.
4. If some pre-labelled training data is present, the system invokes a probabilistic semi-supervised machine learning algorithm.
5. If no pre-labelled data instance is present, the system invokes an unsupervised machine learning algorithm.
6. The system outputs the corresponding class labels and the probability of an instance belonging to its particular class.

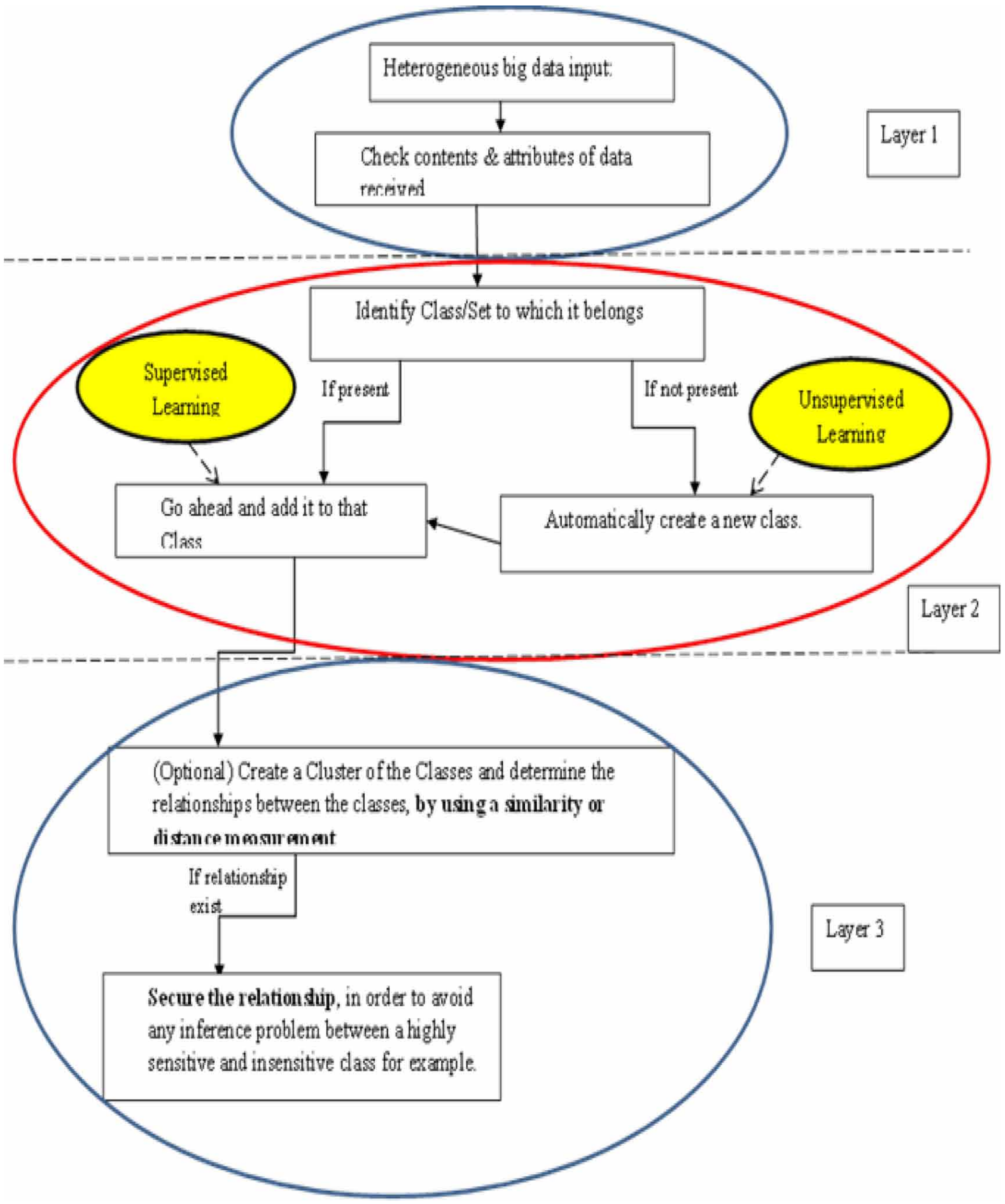
3. THE MODEL DESIGN

3.1. Design Goals and Aims

1. A meta-learning rule-based design that defines a structure for automatically determining whether to invoke a supervised learning algorithm or an unsupervised learning algorithm. One that can be used effectively for achieving automatic pre-processing and model selection of the best machine-learning algorithm for any given dataset.
2. The design of a self-evolving unsupervised clustering algorithm (determining the classes from scratch without any labeled instances). It will allow for effective clustering when required (i.e. depending on what was automatically learned from your data based on the meta-learning phase). Lastly, it should allow for a re-grouping of the classes to avoid having a large dataset of classes.
3. Scalability in terms of the system handling an increasing amount of heterogeneous datasets and data categories. The system input can be datasets from various domains and fields.
4. Achieves classification at a desired speed. It should be able to Achieve classification of the various datasets at a desired speed, making use of some generalization rules and knowledge of supervised and unsupervised algorithms to select automatically the best machine learning algorithm to use in building the model.
5. The model built from automatically selecting the best supervised machine learning algorithm, can be used to make predictions on new instances. While the unsupervised clustering algorithm can be used in identification of anomalies/intrusion if applied or used in an Intrusion Identification System.
6. Flexibility and adaptability. Ensure a high level of flexibility and adaptability of the system to ensure that the learning-to-learn process can improve to enhance an even better performance.

4. MODEL ARCHITECTURE

Figure 1. Three Layered Decision architecture for the hybrid auto machine learning system proposed after experiments


Heterogeneous big data input:
Check contents & attributes of data received
Layer 1
Identify Class/Set to which it belongs
Supervised Learning
If present
If not present
Unsupervised Learning
Go ahead and add it to that Class
Automatically create a new class.
Layer 2
(Optional) Create a Cluster of the Classes and determine the relationships between the classes, by using a similarity or distance measurement
If relationship exist
Layer 3
Secure the relationship, in order to avoid any inference problem between a highly sensitive and insensitive class for example.

4.1. Model Components

Layer 1 (Input / Pre-processing Layer): Since big data is a collection of heterogeneous data which makes it difficult to analyse (Doug, 2001), this layer ensures that an inflow of such a data set is pre-processed appropriately. The pre-processing phase will involve dividing the vast source of data into domain specific sources of knowledge, next a check through the contents and attributes of the data is done to determine if any knowledge or information about its content is present. Having this layer will assist in the process of preventing vagueness in the heterogeneous data. This layer provides layer 2 the reasoning about classifying data using either a supervised classifier or and unsupervised classifier.

Layer 2 (Strategic Learning Decision Layer): At this layer, the decision on which learning method to invoke is made. The main aim of this layer is automatic classification using the most effective learning method to achieve a high level of accuracy at a fast speed. The hypothesis used in this layer for making a decision is based on some general characteristics and knowledge about supervised and unsupervised machine learning. For example, characteristics such as the existence of pre-existing labelled set for training or not, the size of the pre-labelled training set (under the assumption that the size is relative to the number of instances in a particular dataset), existence of a test set which is a subset of the training set, etc.

Layer 3 (Output / Optional Cluster Formation Layer): This is the output layer. In this layer, an evaluation of the different models built for the different dataset is made. This layer also acts as an optional layer for scalability through a technique of clustering the class labels using a similarity estimate. It is also a layer where the relationships of class labels can be properly secured. Activities like securing the relationships between class labels can be performed in this layer. For example, imagine a scenario in which the amount of resulting class labels becomes very large. The question now becomes: 'how can we effectively manage a large and increasing set of class labels?' At this layer, a good technique to effectively manage a large and self-evolving set of class labels is considered. This technique considers the formation of clusters/groups for the class labels by making use of a similarity or distance measurement. The resulting output from this layer will be a set of cluster labels (similar to the class labels, but for representing some knowledge about the clusters).

4.2. Model Characteristics

Meta-Learning / Automatic Learning Architecture: Where supervised and unsupervised classification algorithms will be combined together and depending on certain characteristics knowledge of the data set under consideration, one of the algorithm is invoked automatically to give more accurate classes. This reduces significantly the time spent in deciding the best classification algorithm to use for a particular data set and the high cost of learning realistically accurate classifiers is overcome.

Multi Class-label type classification: A new unsupervised algorithm is developed in this research, which can be used successfully in second layer of the classification system. The algorithm allows an instance of a dataset to have multiple class labels based on sensitivity levels (e.g. sensitive level l1, l2, etc.) assigned to each attribute per instance, rather than assigning one class label to the data instance as a whole (see illustration of this in Table 1).

From Table 1, there are 4 attribute features and 3 instances of the dataset. Every bank ID and D.o.B. is given a sensitivity class label l1, (where l1 is assumed to be the most sensitive class), every instance of the Fname is given the label l2 and the Lname is given a label l3. From this, it is observed that each instance in the data set may have one or more class labels.

Table 1. Hypothetical example case study of a multi-class labels unsupervised algorithm

#	Bank ID	LName	FName	D.o.B
1	10a	Flora	Catch	29.09.83
2	20s	Robin	Thomson	05.10.75
3	3b	Martha	Woods	04.7.60
Class	L1	L3	L2	L1

1. Meta-Classification: this simply means a process of classifying the classes.
2. Multilevel type structure classification.
3. Auto-Class functionality: the beneficial features of Auto-Class includes: 1) its ability to determine the number of classes automatically, 2) it permits the blend of discrete and real valued data, 3) it can handle missing values effectively.
4. Classification Methods to be used: Probabilistic and Rule based methods will be employed.
5. Output: the intended output per instance will be a numerical score that can be converted to a discrete label.

4.3. The Model Algorithms

4.3.1. Decision (Meta) Learning Algorithm

Input: An inflow collection of either labeled (D_l) datasets or unlabeled (D_u) datasets or both from heterogeneous data sources and a collection of fully unlabeled heterogeneous dataset (D). Also, a set of IF $\rightarrow$ THEN rules defined from experimental knowledge obtained about supervised and unsupervised learning, that helps in the decision-making process.

Output: A decision that invokes either a supervised classification algorithm or an unsupervised classification algorithm.

1. IF training labeled set exists then check the size of the labeled set.
2. IF size of the training set > than the test set, THEN invoke a supervised learning method.
3. IF no training set exists, THEN use an unsupervised algorithm.
4. IF the size of the training set < or = test set, use an unsupervised algorithm.
5. IF no labeled instances exist, use an unsupervised algorithm.
6. Output new decision by automatically invoking a learning algorithm that is the best fit for that dataset.

4.3.2. AutoProbClass Unsupervised Algorithm

An autoProbClass unsupervised algorithm: A self-evolving multi-label fuzzy unsupervised algorithm called the ‘autoProbClass’ is designed in this layer. The autoProbClass algorithm combines two similarity/distance measurements. The first similarity measurement is an instance identifier (based on its attribute weighted value) similarity fraction measurement and the second is the Euclidean distance measurement. Euclidean distance measurement is a very popular distance (or similarity) function in the field, were one object describes not one distance but also the data model in which the distances between objects of that model can be calculated.

Input: Unlabeled or partly labeled datasets.

- IF the first instance in the dataset is read,
    - An instance identifier Ĩ is created.

$$\tilde{I} += V_i \text{ where } i < n_a \tag{4.1}$$

- The instance identifier Ĩ is a string.
- V_i = the value of a data instance i
- n_a = the number of attributes for the given instance.
- Instance.value(i) is a method via the WekaAPI that will return an instance's attribute value in internal format.
- For example if we have an instance [young,myope,no,reduced,none] from our contact lenses dataset which has the following attributes:
- @attribute age {young, pre-presbyopic, presbyopic}
- @attribute spectacle-prescrip {myope, hypermetrope}
- @attribute astigmatism {no, yes}
- @attribute tear-prod-rate {reduced, normal}
- @attribute contact-lenses {soft, hard, none}
- Then the identifier $\tilde{I}$ for that instance will be '00002' and another instance [young,hypermetrope ,yes,normal,soft] will have an identifier of '01110'. It uses index points per instance, per attribute value.
    - A new class is created and is added to a Dense Instance list called 'cloud'.
    - Then a label 'ClassK' is created and the label is added to a list of all Class labels. Where K is a counter set for keeping track of the number of class labels created.
- IF it is not the first instance been read, then
    - An instance identifier is created for that new instance.
    - The new instance is then compared with the previous instance/instances in the 'cloud' list, using their instance identifiers. The method to compare the Instances does the following:
        - IF the instanceOldIdentifier.value(i) is the same as the instanceNewIdentifier.value(i), then a true score sum is accumulated.
        - ELSE IF the instanceOldIdentifier.value(i) is NOT the same as the instanceNewIdentifier.value(i), then a false score accumulated.
        - Then a dissimilarity measure is calculated as follows:

$$D = 100\left({}^{F}\!/_{n}\right)\% \tag{4.2}$$

- Where D = dissimilarity, F= false score and n = total number of attributes.
    - While the similarity measure is denoted as: $S=1-D$.

    - It is assumed that for a new instance to be like an old instance, then the dissimilarity measure should be small (for example, we have assumed a score of less than 20%). This assumption can be changed to an even smaller value, to further ensure that the dissimilarity between the two instances is small enough to help in deciding whether they will belong to the same cluster or not.
    - IF the dissimilarity measure is high, then 'false' is added to a 'howCloseList' (which is a list containing the closeness comparison of the instances), ELSE 'true' is added to the list. When a 'true' is recorded in the 'howCloseList', then the percentage of similarity measure is also recorded in a 'simPercent' list at that same index point a 'true' was recorded in the 'howCloseList'. Where a 'false' was recorded, we record a floating-point value of 0.0 in the 'simPercent' list (this just means we are not interested in the similarity measure if the instances are not in the first place at all similar).
    - The 'Euclidean Distance' is also estimated between the newly read instance and the old instance/instances in the 'cloud' list.
- After the compareInstancesTest() has been performed, we get the class label value for the previous instance that is the closest to the new instance, by using the index of the maximum value in the 'simPercent' list.
- IF the maximum value is '0.0' in the 'simPercent' list, then it is assumed that the new instance was in no way like the previous instances. Hence, we create a new class for it and a corresponding new Class label. ELSE we assign the new instance into the same cluster as the closest previous instance to it, as well as assign the corresponding class label to it.

4.4. Design Materials

4.4.1. Weka API

As stated in previous chapters, the Wekatool when downloaded comes with an application programming Interface (API), this API which could be a '.jar' file source packaged with Weka is added in as a library path of the project's implementation in my development environment. The API provides several methods and functions of the Weka tool which is used in a flexible manner to implement the system model. Some functions provided via the API includes: a function for calculating the 'Euclidean distance' between two data points, a function for performing cross validation tests, another for plotting and visualising results via ROC curves, etc.

4.4.2. NetBeans IDE

Netbeans Integrated Development Environment (IDE) was used to implement a Java based application of the model designed. IDEs provide a controlled environment for developing or implementing software designs. The choice was made to use Netbeans IDE because it is a very popular tool when building or implementing java based application and it is easy and friendly to use.

4.4.3. Program

The programming language of choice for the implementation of this research is the Java Programming Language. Some reasons for using Java is because, it is a very familiar programming language, it has a very big user support community, it is efficient in building scalable, flexible software solutions and lastly, Weka is java based and came readily with an API to help aide customisable implementations.

4.5. Testing and Evaluation of System Model

Several case studies and scenarios have been created to guide in the testing and evaluation of the implemented prototype of the system. They are as follows:

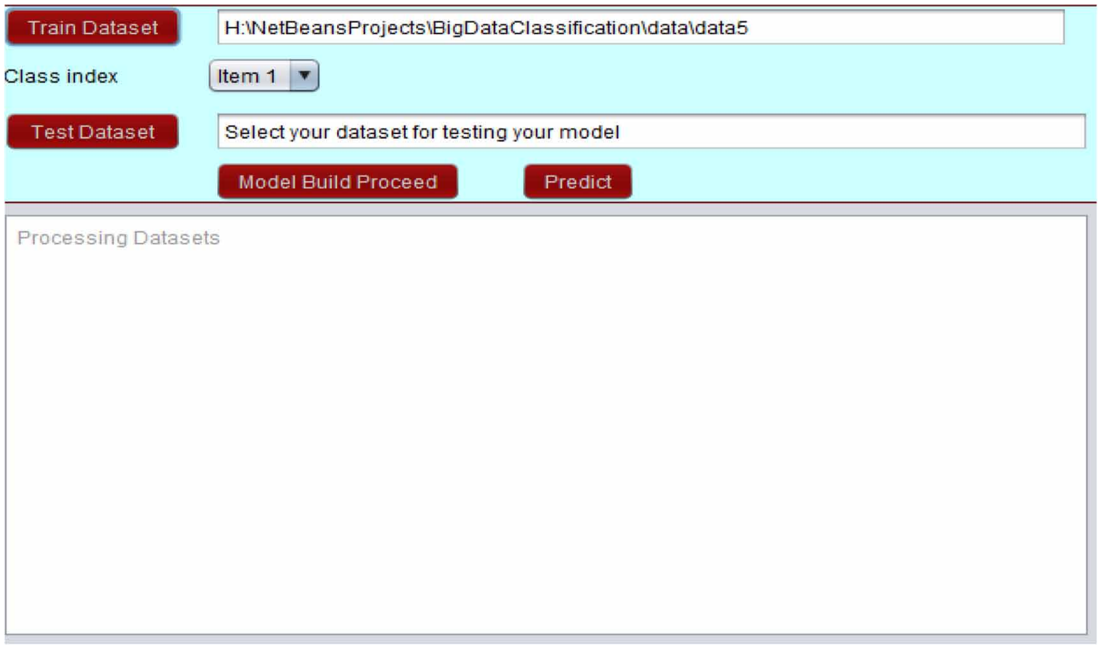
4.5.1. Case Study 1

The hybrid automatic classification rule-based algorithm is implemented in this stage, validated and tested using some datasets not used initially in the preliminary experiments. The rule-based algorithm was written as a result of the fact that, from the preliminary experiments, it was determined that general knowledge about the data set e.g. the size of the training set, the class attribute type, the number of nominals versus numeric attribute, etc. definitely influences the choice of the algorithm to be selected. Implementation of the algorithm and the knowledge gained from preliminary experiments, were written in Java codes using the Weka API, to determine if the rules remain valid whenever it is applied to any other datasets not initially used. The set of rules implemented are derived using the result observations from the preliminary experiments. The datasets used in this stage are the: 'breast-cancer', 'iris', 'soy-test', 'reuters Grain-train' & 'results'. They were all placed together in the same file path, and using the system designed only the main file path was supplied. Doing this helped us to determine two things as proposed in this paper and in hypothesis 1, which includes:

1. When provided with a heterogeneous multi data source, can we automatically take decisions on what mode or model to build for each dataset by using some more general knowledge about machine learning and each dataset?
2. If a decision to use supervised machine learning is made, how then can we build in a timely manner the best model per dataset using the rule based decision-making algorithm described in this research?

Figure 2 shows the simple GUI implemented for the system model designed in this paper. Using this system, the user can use the 'TrainDataset' button to choose a file directory containing all the datasets to build the individual models for. Or they can supply a single train dataset and a test set using the 'TestDataset' button. If we supply a single training Dataset, we have the option to set what the Class index in the dataset is (if this exists and is known). Without setting the class index for it, it will be assumed that it is an unlabelled dataset and going by the design architecture proposed in this paper, when the 'Model Build Proceed' button is clicked, the 'autoProbClassifier' (Unsupervised ML algorithm) is automatically used in that scenario. If a file directory path (containing several datasets) is selected instead, then the last attribute in the dataset is automatically selected as the class attribute for labelled datasets, and clicking the 'Model Build Proceed' button in that scenario will automatically build the most suitable Supervised classification model for each dataset in the directory, by using general knowledge of that dataset and the set of rules derived after the preliminary experiments had been carried out. This case study shows that hybrid-autoML can allow for the automatic mode and model selection on multiple-varying datasets at the same time. Therefore proving our aims from section 1.3 has been achieved.

Figure 2. Simple GUI interface for the Implementation of the Hybrid Auto Classification System



H:\NetBeansProjects\BigDataClassification\data\data5

Select your dataset for testing your model

4.5.2. Case Study 2

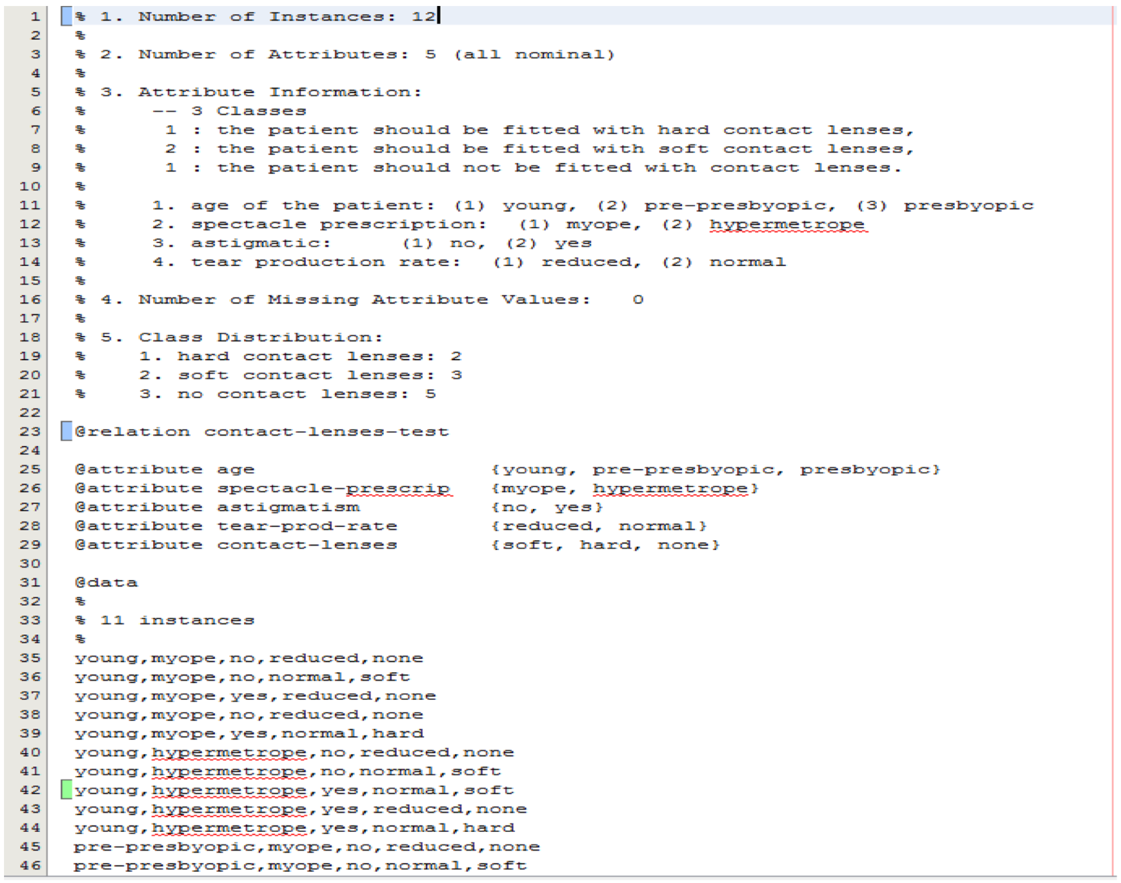
This is the stage of implementing an alternative 'autoProb' function designed in this research. In this given scenario a single dataset file is supplied using the 'Train Dataset' file chooser and the user does not select a class index. If the class Index is not chosen, when a single dataset is supplied, we assume that the training data supplied is unlabelled, hence the decision to use the 'autoProb' self-evolving or any other unsupervised algorithm is made. Results for this scenario is discussed in section 5 and proves if our aims 1 and 2 from section 1.3 have been achieved.

For simplicity, we describe the testing of autoProbClassifier's implementation using the 'contact-lenses-test' dataset (listed in the *full datasets list* table in *Appendix*).

SUMMARY

We presented the system requirements, design materials, model algorithms and model design which encompasses the design goals, architecture (a three-layered architecture), components and characteristics of the 'Hybrid-AutoML' toolkit developed in this research for automatic mode and model selection on single or multi-varying datasets. In the next chapter we evaluate the results obtained from the design implementations.

Figure 3. Details of the contact-lenses-test dataset used



```
% 1. Number of Instances: 12
%
% 2. Number of Attributes: 5 (all nominal)
%
% 3. Attribute Information:
%      -- 3 Classes
%       1 : the patient should be fitted with hard contact lenses,
%       2 : the patient should be fitted with soft contact lenses,
%       1 : the patient should not be fitted with contact lenses.
%
%      1. age of the patient: (1) young, (2) pre-presbyopic, (3) presbyopic
%      2. spectacle prescription:  (1) myope, (2) hypermetrope
%      3. astigmatic:      (1) no, (2) yes
%      4. tear production rate:   (1) reduced, (2) normal
%
% 4. Number of Missing Attribute Values:   0
%
% 5. Class Distribution:
%     1. hard contact lenses: 2
%     2. soft contact lenses: 3
%     3. no contact lenses: 5

@relation contact-lenses-test

@attribute age                     {young, pre-presbyopic, presbyopic}
@attribute spectacle-prescrip      {myope, hypermetrope}
@attribute astigmatism             {no, yes}
@attribute tear-prod-rate          {reduced, normal}
@attribute contact-lenses          {soft, hard, none}

@data
%
% 11 instances
%
young,myope,no,reduced,none
young,myope,no,normal,soft
young,myope,yes,reduced,none
young,myope,no,reduced,none
young,myope,yes,normal,hard
young,hypermetrope,no,reduced,none
young,hypermetrope,no,normal,soft
young,hypermetrope,yes,normal,soft
young,hypermetrope,yes,reduced,none
young,hypermetrope,yes,normal,hard
pre-presbyopic,myope,no,reduced,none
pre-presbyopic,myope,no,normal,soft
```

APPENDIX

Lists of most datasets used throughout this project. Subsets of this list, are referred to at different points within the main content area.

Table 2. A table summary of datasets used in this research

Dataset	# Instances	#Attributes	Class attribute type	Missing Values
contact-lenses	24	All nominal (5)	Nominal	No
Cpu	209	All numeric (7)	Numeric	No
cpu.with.vendor	209	1 Nominal, 7 Numeric	Numeric	No
credit-g	1000	14 Nominal, 7 Numeric	Nominal	No
Diabetes	768	8 Numeric, 1 Nominal	Nominal	No
Glass	214	9 Numeic, 1 Nominal	Nominal	No
Ionosphere	351	34 numeric, 1 Nominal	Nominal	No
iris.2D	150	2 Numeric, 1 Nominal	Nominal	No
Labor	57	9 nom, 8 numeric	Nominal	Yes (2%)
reutersCorn-train	1554	String	Nominal	No
segment-challenge	1500	19 Numerical, 1 Nominal	Nominal	No
Soybean	683	36 Nominal	Nominal	Yes (<1%)
Soytest	26	36 Nominal	Nominal	No
Supermarket	4627	217 nominal	Nominal	Up tp 77%
Unbalanced	856	32 numerical, 1 Nominal	Nominal	No
Vote	435	17 nominal	Nominal	Yes (3%)
weather.nominal	14	5 nominal	Nominal	No
weather.numeric	14	2 Numeric, 3 Nominal	Nominal	No
Dexter	420	20001 Numeric	Numeric	No
Dorothea	805	100000 Numeric	Numeric	No
Yeast	1039	8 Numeric, 1 Nominal	nominal	No
Amazon	1050	10001 numeric	Nominal	No
Secom	1097	591 nominal	Nominal	Yes (5%)
Semeion	1116	256 numeric 1 nominal	Nominal	No
Car	1209	7 nominal	Nominal	No
Madelon	1820	500 numeric 1 nominal	Nominal	No
KR-VS-KP	2238	37 nominal	nominal	No
Abalone	2923	2 nominal, 7 numeric	nominal	No
Wine Quality	3429	11 numeric, 1 nominal	Nominal	No

continues on following page

Table 2. Continued

Dataset	# Instances	#Attributes	Class attribute type	Missing Values
Waveform	3500	40 numeric, 1 nominal	Nominal	No
Gisette	4900	5000 numeric 1 nominal	Nominal	No
Convex	8000	784 numeric, 1 nominal	Nominal	No
Cifar-10-small	10000	3072 numeric, 1 nominal	Nominal	No
Mnist Basic	12000	784 numeric, 1 nominal	Nominal	No
Shuttle	43500	9 numeric, 1 nominal	Nominal	No
KDD09-Appentency	35000	192 numeric 39 nominal	Nominal	Yes(99%)
Cifar-10	50000	3072 numeric 1 nominal	Nominal	No

Chapter 12
Research Output for the Hybrid-AutoML System

ABSTRACT

In this chapter, the authors use a set of use cases to evaluate how the hybrid autoML system is used to achieve the goals set out in the aims and objectives of this research. The authors map each use case to their aims and contributions as outlined in Section 1.3 of this research. A performance comparison is also made between autoWeka and the hybrid autoML system on 33 datasets. The comparison is carried out based on three main evaluation metrics such as the percentage accuracy (or correlation coefficient where applicable), the mean absolute error (MAE), and the time (in seconds) spent building the model on training data. It is observed that the hybrid autoML system fully outperforms autoWeka with regards to the time spent on building models or finding the best algorithms in the first instance.

1. INTRODUCTION

Use cases describe specific situations in which a product or service could potentially be used. They are used mainly during the analysis phase of a project to identify systems functionality. It is made up of a set of possible sequences of interactions between a system and users within an environment and related to a goal or goals of the system. The use case should contain all system activities that have significance to the users within a given system.

In this chapter, we use a set of use cases to evaluate how the hybrid autoML system is used to achieve the goals set out in the aims and objectives of this research. We map each use case to our aims and contributions as outlined in section 1.3 of this research. A performance comparison is also made between autoWeka and the hybrid autoML system on 33 datasets. The comparison is carried out based on three main evaluation metrics such as, the percentage accuracy (or correlation coefficient where applicable), the mean absolute error (MAE) and the time (in seconds) spent building the model on training data. From using the use cases and comparison analysis, it is observed that the aims and objectives of this research has been met fully. Also, the performance comparison shows that the hybrid autoML system performs

DOI: 10.4018/978-1-7998-7316-7.ch012

relatively close to or better than autoWeka on most of the datasets used. Overall, an interesting fact is that the hybrid autoML system fully outperforms autoWeka with regards to the time spent on building models or finding the best algorithms in the first instance.

2. EVALUATION OF USE CASES

The following use cases can be used to replicate some of the scenarios used in evaluating the prototyped implementations for our hybrid autoML system.

2.1. Use Case 1 (Small Unlabeled Dataset)

We supply the hybrid system with a small unlabelled dataset. For this use case, we use the 'soy-test' dataset. This data contains thirty-six attributes and twenty-six unlabelled data instances. The aim of this use case is to show that in such a scenario, the hybrid autoML system will automatically choose an unsupervised clustering algorithm. It is expected that since it is a small unlabelled dataset, the 'AutoProbClass' algorithm described in section 4.3.2 is automatically selected. Hence, proving that the contribution of this research to aid the automatic selection of an unsupervised ML algorithm e.g. the 'AutoProbClass' when supplied with an unlabelled dataset has been achieved. Figure 1 shows the data summary for the 'soy-test' dataset used in this first user scenario.

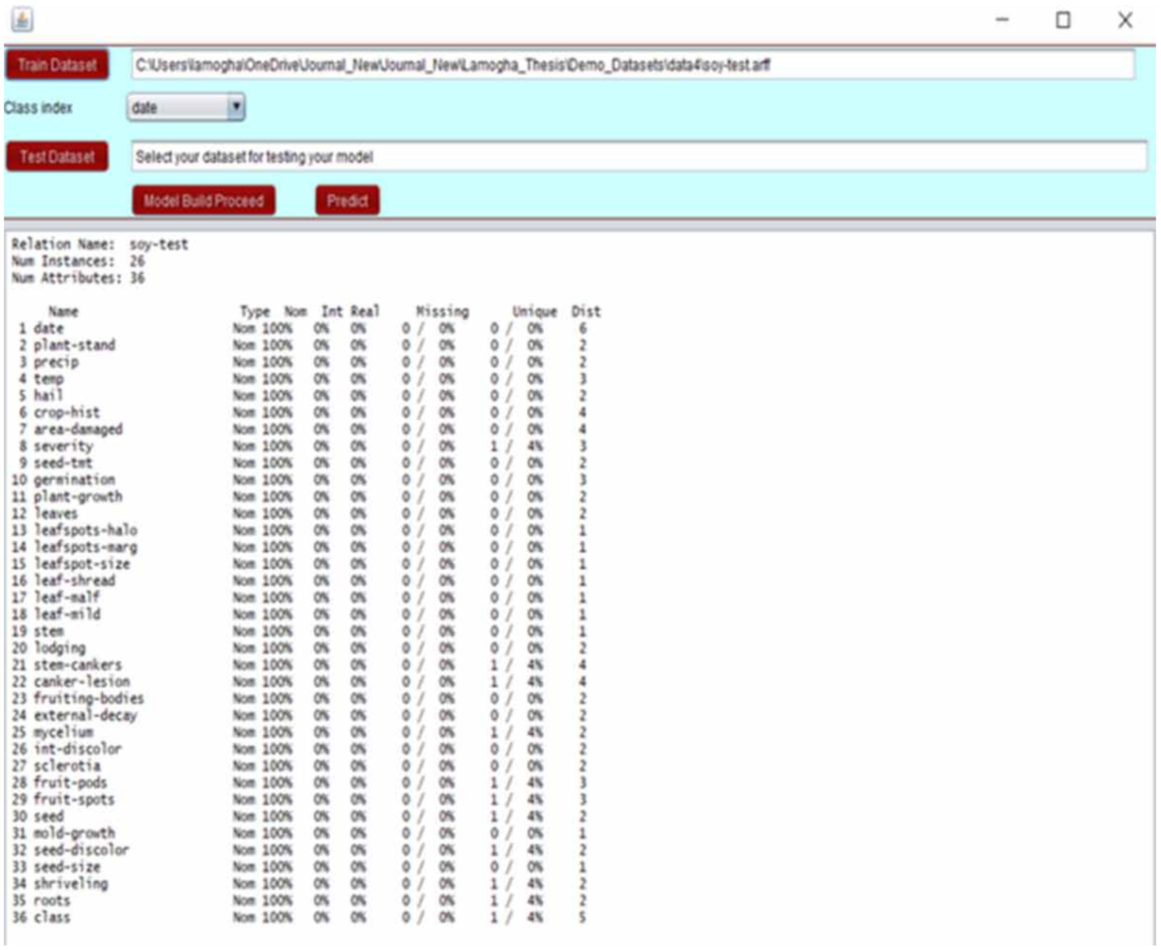
After the upload of the dataset and clicking of the 'Model Build Proceed' button, the system automatically assumes an un-labelled dataset. This occurs when the class index is not set using the 'Class index' dropdown menu and when the target class is an unknown variable for each instance within the dataset.

From Figure 2 above, we can see that in about 0.03 seconds the system automatically chooses a clustering algorithm for the given task. The algorithm modelled uses the 'AutoProbClass' function designed in this research to create six clusters for the given dataset. Finally, we observe from the figure that the aim of this use case showing the contribution of the hybrid autoML system providing a function for automatic selection of a learning scheme, as well as an 'AutoProbClass' function for clustering a small unlabelled dataset has been achieved.

2.2. Use Case 2 (Larger Unlabeled Dataset)

We supply a larger unlabelled dataset. For this use case, an unlabelled 'german-credit' dataset is used. This dataset contains twenty-one attribute variables and seven hundred data instances. The aim of this use case is to further describe and explain how we have implemented and achieved the objective set out in this research to have a function for automatically selecting a learning scheme or model, given a large unlabelled dataset. It is expected from this use case and given the hybrid system's function for model selection, that an unsupervised algorithm e.g. the EM algorithm will be automatically selected. After uploading the dataset and clicking on the 'Model Build Proceed' button, the system automatically assumes an un-labelled dataset, same as in use case 1. The hybrid autoML system, goes further to automatically choose a clustering algorithm for the given task as described below

Figure 1. Shows a data summary on upload of the small unlabelled dataset (soy-test)


Train Dataset
C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\data4\soy-test.arff
Class index
date
Test Dataset
Select your dataset for testing your model
Model Build Proceed
Predict

```
Relation Name:  soy-test
Num Instances:  26
Num Attributes: 36

    Name                       Type  Nom  Int Real     Missing      Unique  Dist
 1 date                        Nom 100%   0%   0%     0 /  0%     0 /  0%     6
 2 plant-stand                 Nom 100%   0%   0%     0 /  0%     0 /  0%     2
 3 precip                      Nom 100%   0%   0%     0 /  0%     0 /  0%     2
 4 temp                        Nom 100%   0%   0%     0 /  0%     0 /  0%     3
 5 hail                        Nom 100%   0%   0%     0 /  0%     0 /  0%     2
 6 crop-hist                   Nom 100%   0%   0%     0 /  0%     0 /  0%     4
 7 area-damaged                Nom 100%   0%   0%     0 /  0%     0 /  0%     4
 8 severity                    Nom 100%   0%   0%     0 /  0%     1 /  4%     3
 9 seed-tmt                    Nom 100%   0%   0%     0 /  0%     0 /  0%     2
10 germination                 Nom 100%   0%   0%     0 /  0%     0 /  0%     3
11 plant-growth                Nom 100%   0%   0%     0 /  0%     0 /  0%     2
12 leaves                      Nom 100%   0%   0%     0 /  0%     0 /  0%     2
13 leafspots-halo              Nom 100%   0%   0%     0 /  0%     0 /  0%     1
14 leafspots-marg              Nom 100%   0%   0%     0 /  0%     0 /  0%     1
15 leafspot-size               Nom 100%   0%   0%     0 /  0%     0 /  0%     1
16 leaf-shread                 Nom 100%   0%   0%     0 /  0%     0 /  0%     1
17 leaf-malf                   Nom 100%   0%   0%     0 /  0%     0 /  0%     1
18 leaf-mild                   Nom 100%   0%   0%     0 /  0%     0 /  0%     1
19 stem                        Nom 100%   0%   0%     0 /  0%     0 /  0%     1
20 lodging                     Nom 100%   0%   0%     0 /  0%     0 /  0%     2
21 stem-cankers                Nom 100%   0%   0%     0 /  0%     1 /  4%     4
22 canker-lesion               Nom 100%   0%   0%     0 /  0%     1 /  4%     4
23 fruiting-bodies             Nom 100%   0%   0%     0 /  0%     0 /  0%     2
24 external-decay              Nom 100%   0%   0%     0 /  0%     0 /  0%     2
25 mycelium                    Nom 100%   0%   0%     0 /  0%     1 /  4%     2
26 int-discolor                Nom 100%   0%   0%     0 /  0%     0 /  0%     2
27 sclerotia                   Nom 100%   0%   0%     0 /  0%     0 /  0%     2
28 fruit-pods                  Nom 100%   0%   0%     0 /  0%     1 /  4%     3
29 fruit-spots                 Nom 100%   0%   0%     0 /  0%     1 /  4%     3
30 seed                        Nom 100%   0%   0%     0 /  0%     1 /  4%     2
31 mold-growth                 Nom 100%   0%   0%     0 /  0%     0 /  0%     1
32 seed-discolor               Nom 100%   0%   0%     0 /  0%     1 /  4%     2
33 seed-size                   Nom 100%   0%   0%     0 /  0%     0 /  0%     1
34 shriveling                  Nom 100%   0%   0%     0 /  0%     1 /  4%     2
35 roots                       Nom 100%   0%   0%     0 /  0%     1 /  4%     2
36 class                       Nom 100%   0%   0%     0 /  0%     1 /  4%     5
```

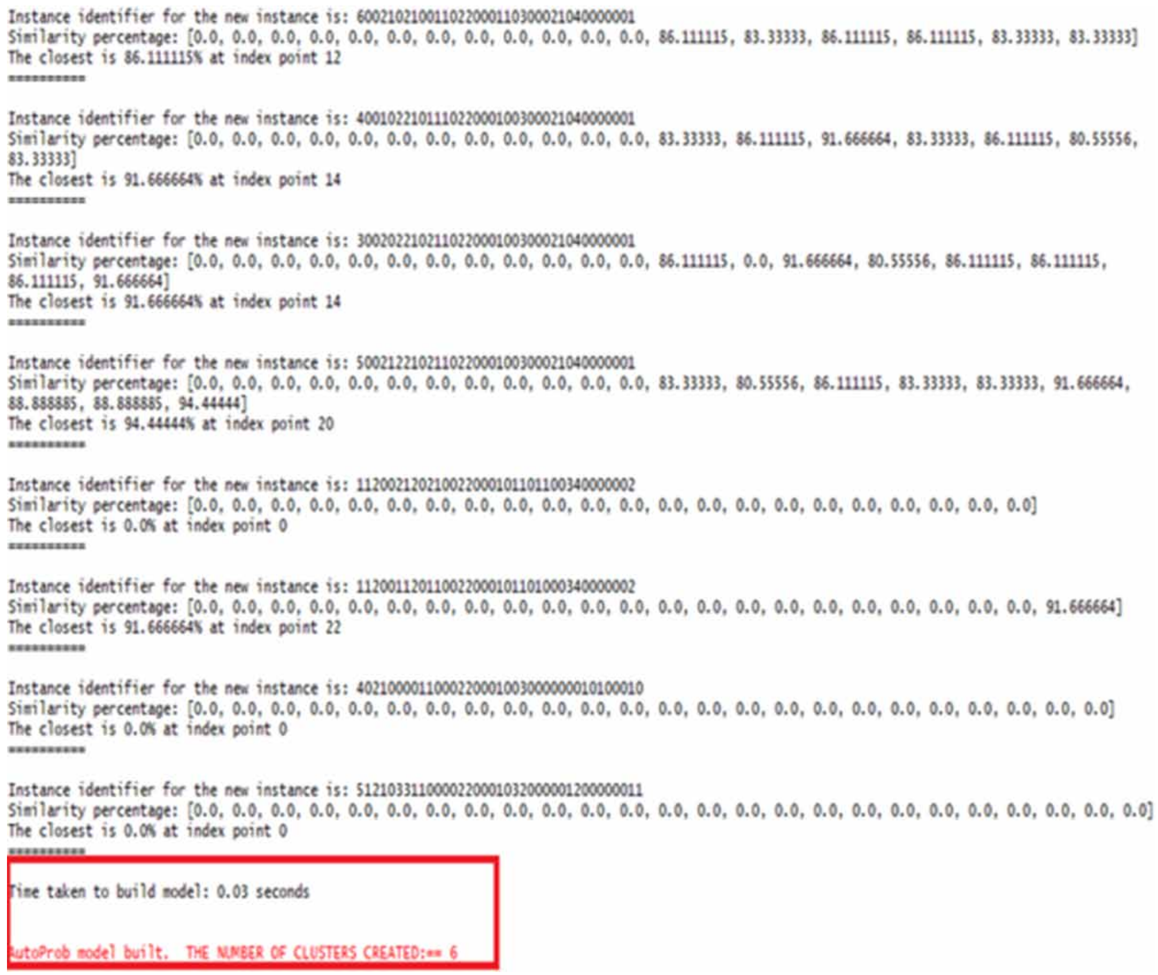
Hence, it can be said that an unsupervised algorithm is more appropriate to use in the absence of a large pre-labelled training set. It has also been observed that using general knowledge about a dataset such as the size of the training set compared with the test set, the class attribute type, the number of numeric in comparison to number of nominal attributes, etc. turned into a rule based algorithm, allows for this automatic mode and model selection.

2.3. Use Case 3 (Large Labelled Train Data With Smaller Test Data)

We have a large labelled dataset and some smaller test dataset. For this use case, we supply the system the ‘gissette’ train and test datasets. The training dataset contains 4900 instances and 5001 attribute variables, while the test dataset contains 2100 instances and 5001 attribute variables. The last attribute in each dataset represents the target class attribute (which is selected using the ‘class index’ selector of the system, after uploading both datasets). The aim of this use case is to show that in a given scenario where a user has a large labelled dataset and some smaller test set, then it is expected that the hybrid autoML system uses it’s rule based algorithm to decide on selecting a supervised learning algorithm. It is also expected that the most appropriate supervised algorithm is selected automatically and in a small amount of time, from a pool of various supervised ML algorithms implemented into the hybrid autoML function for model selection. The following figures and discussions below describes the evaluation of the hybrid autoML system in this user scenario.

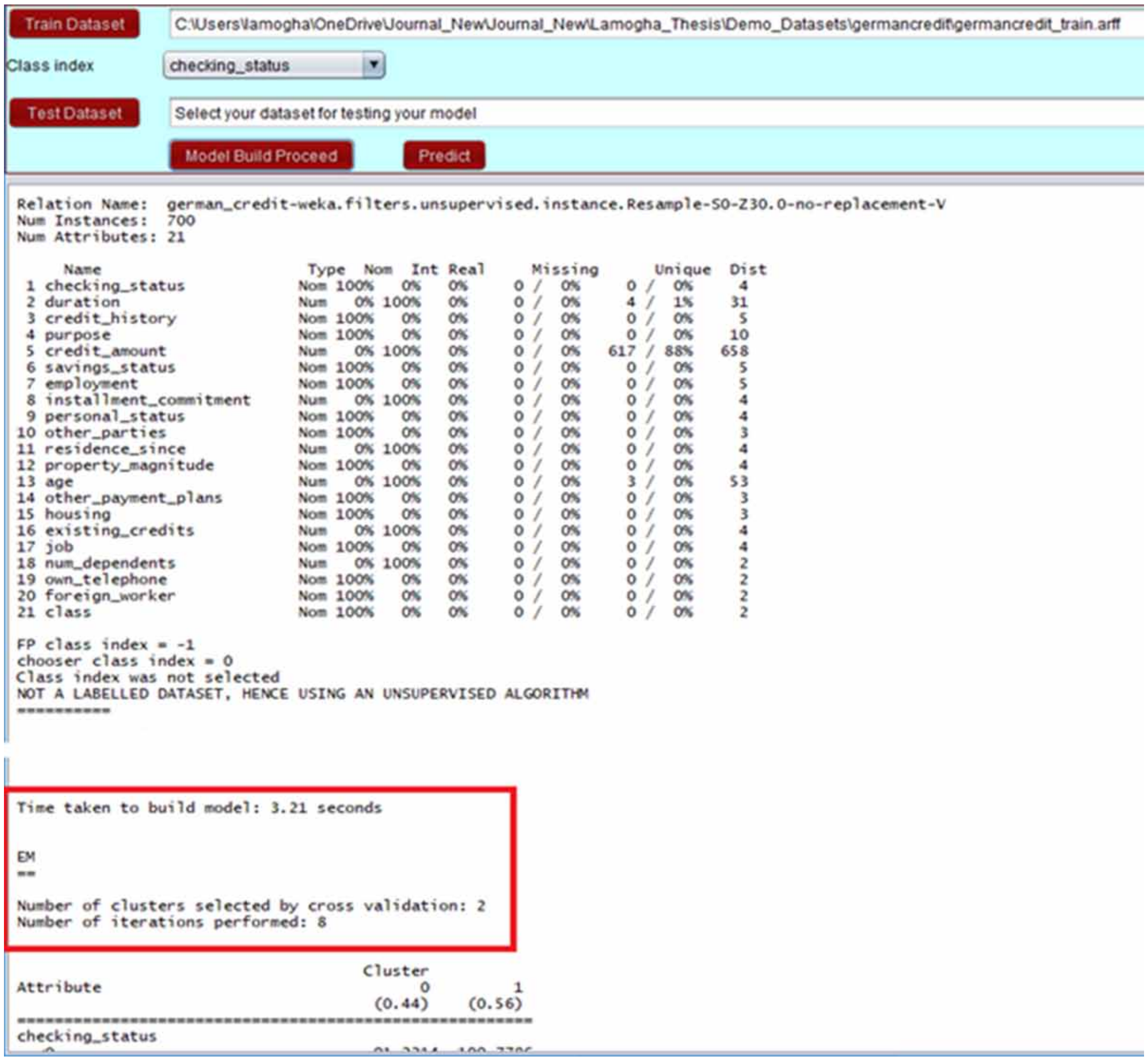
From Figure 5 above, we can see that the hybrid system by following the rule base algorithm designed in this research, automatically uses the random forest to build a model for our given dataset. It is also observed from Figure 5 some evaluation metrics, which measures to what extent the system performs in this instance. For example, the time taken to build the model was 30.8 seconds with an accuracy of 96.52% and MAE of 0.17. The area under the ROC (AUC) as displayed just above the chosen classifier used is 0.99 (as shown in Figure 4). A value closer to 1 for the AUC, represents a high performing classifier, while a value closer to 0 represents a poorly performed classifier. From Figure 5 above, we can see that the classifier automatically used by our hybrid system performed highly in this use case.

Figure 2. shows that an unsupervised ML mode was selected automatically and a clustering model constructed by engaging autoProb clustering function on the soy-test dataset. This model automatically resulted in six cluster been identified in under 0.03 seconds



```
Instance identifier for the new instance is: 600210210011022000110300021040000001
Similarity percentage: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 86.111115, 83.33333, 86.111115, 86.111115, 83.33333, 83.33333]
The closest is 86.111115% at index point 12
==========

Instance identifier for the new instance is: 400102210111022000100300021040000001
Similarity percentage: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 83.33333, 86.111115, 91.666664, 83.33333, 86.111115, 80.55556,
83.33333]
The closest is 91.666664% at index point 14
==========

Instance identifier for the new instance is: 300202210211022000100300021040000001
Similarity percentage: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 86.111115, 0.0, 91.666664, 80.55556, 86.111115, 86.111115,
86.111115, 91.666664]
The closest is 91.666664% at index point 14
==========

Instance identifier for the new instance is: 500212210211022000100300021040000001
Similarity percentage: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 83.33333, 80.55556, 86.111115, 83.33333, 83.33333, 91.666664,
88.888885, 88.888885, 94.44444]
The closest is 94.44444% at index point 20
==========

Instance identifier for the new instance is: 112002120210022000101101100340000002
Similarity percentage: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
The closest is 0.0% at index point 0
==========

Instance identifier for the new instance is: 112001120110022000101101000340000002
Similarity percentage: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 91.666664]
The closest is 91.666664% at index point 22
==========

Instance identifier for the new instance is: 402100001100022000100300000000010100010
Similarity percentage: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
The closest is 0.0% at index point 0
==========

Instance identifier for the new instance is: 512103311000022000103200000120000011
Similarity percentage: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
The closest is 0.0% at index point 0
==========

Time taken to build model: 0.03 seconds

AutoProb model built.  THE NUMBER OF CLUSTERS CREATED:== 6
```

Figure 3. shows an unsupervised ML mode using the EM clustering algorithm was automatically chosen as the best to use for this given task. Two clusters where derived and the EM model built in 3.21 seconds.



```
C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\germancredit\germancredit_train.arff
checking_status
Select your dataset for testing your model
```

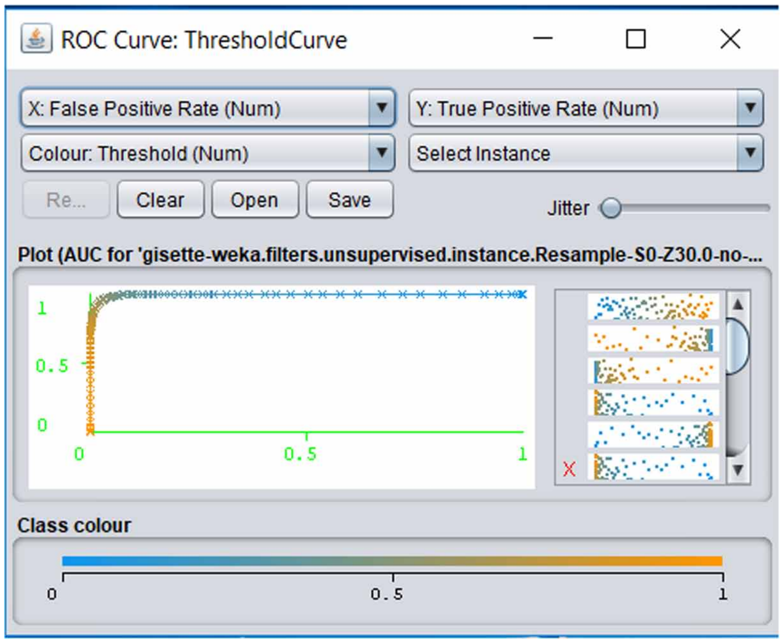
```
Relation Name:  german_credit-weka.filters.unsupervised.instance.Resample-S0-Z30.0-no-replacement-V
Num Instances:  700
Num Attributes: 21

     Name                      Type  Nom  Int Real     Missing      Unique  Dist
  1 checking_status            Nom 100%   0%   0%     0 /  0%     0 /  0%     4
  2 duration                   Num   0% 100%   0%     0 /  0%     4 /  1%    31
  3 credit_history             Nom 100%   0%   0%     0 /  0%     0 /  0%     5
  4 purpose                    Nom 100%   0%   0%     0 /  0%     0 /  0%    10
  5 credit_amount              Num   0% 100%   0%     0 /  0%   617 / 88%   658
  6 savings_status             Nom 100%   0%   0%     0 /  0%     0 /  0%     5
  7 employment                 Nom 100%   0%   0%     0 /  0%     0 /  0%     5
  8 installment_commitment     Num   0% 100%   0%     0 /  0%     0 /  0%     4
  9 personal_status            Nom 100%   0%   0%     0 /  0%     0 /  0%     4
 10 other_parties              Nom 100%   0%   0%     0 /  0%     0 /  0%     3
 11 residence_since            Num   0% 100%   0%     0 /  0%     0 /  0%     4
 12 property_magnitude         Nom 100%   0%   0%     0 /  0%     0 /  0%     4
 13 age                        Num   0% 100%   0%     0 /  0%     3 /  0%    53
 14 other_payment_plans        Nom 100%   0%   0%     0 /  0%     0 /  0%     3
 15 housing                    Nom 100%   0%   0%     0 /  0%     0 /  0%     3
 16 existing_credits           Num   0% 100%   0%     0 /  0%     0 /  0%     4
 17 job                        Nom 100%   0%   0%     0 /  0%     0 /  0%     4
 18 num_dependents             Num   0% 100%   0%     0 /  0%     0 /  0%     2
 19 own_telephone              Nom 100%   0%   0%     0 /  0%     0 /  0%     2
 20 foreign_worker             Nom 100%   0%   0%     0 /  0%     0 /  0%     2
 21 class                      Nom 100%   0%   0%     0 /  0%     0 /  0%     2

FP class index = -1
chooser class index = 0
Class index was not selected
NOT A LABELLED DATASET, HENCE USING AN UNSUPERVISED ALGORITHM
==========

Time taken to build model: 3.21 seconds

EM
==

Number of clusters selected by cross validation: 2
Number of iterations performed: 8

                              Cluster
Attribute                           0         1
                               (0.44)    (0.56)
=================================================
checking_status
```

While showing the use of general knowledge about a dataset such as the size of the training set compared with the test set, etc. we proved that a supervised algorithm is more appropriate to use than an unsupervised algorithm in the presence of large pre-labelled training set, and turning this into a rule based algorithm, allows for automatic mode and model selection in such a scenario.

2.4. Use Case 4 (Small Labelled Train Data With Large Test Data)

The user only has a small labelled training dataset and large test dataset which they supply to our hybrid autoML system. In this example, we use a labelled version of the 'soy-test' dataset (from use Case 1) as our training dataset and an unlabelled 'soybean' dataset (containing 683 data instances and 36 attribute variables). The aim of this use case is to prove that the hybrid function for automatic model selection is effective enough to show that using a supervised model in such a scenario is not as effective when a user only has a small labelled training dataset as opposed to a large dataset for training. The ideal is to extend the hybrid autoML system designed in this research to include the automatic model selection of a semi-supervised classifier if faced with these types of conditions. The following figures and discussions below provide an evaluation of the results obtained after carrying out this use case in the hybrid system.

Figure 4. The ROC curve obtained after a model was built and tested using the gisette data set


ROC Curve: ThresholdCurve
X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Re...
Clear
Open
Save
Jitter
Plot (AUC for 'gisette-weka.filters.unsupervised.instance.Resample-S0-Z30.0-no-...
1
0.5
0
0
0.5
1
X
Class colour
0
0.5
1

From the following ROC curve above in Figure 6, we observed that the AUC when building our training model was 0.91 (a value closer to 1 than to 0). This AUC indicates a high performing classifier model, however when combined with other evaluation metrics as shown in Figure 7 above, it is observed that a Naïve Bayes model built on the small training dataset and tested on the larger test dataset had a very low accuracy. The reason for this is that, ideally in this scenario, a semi-supervised algorithm should be the right choice. However, the hybrid autoML rule-based algorithm was designed and constructed based on a variety of supervised and unsupervised algorithms supplied by WEKA. WEKA via it's API currently lacks an easy way of using semi-supervised algorithms.

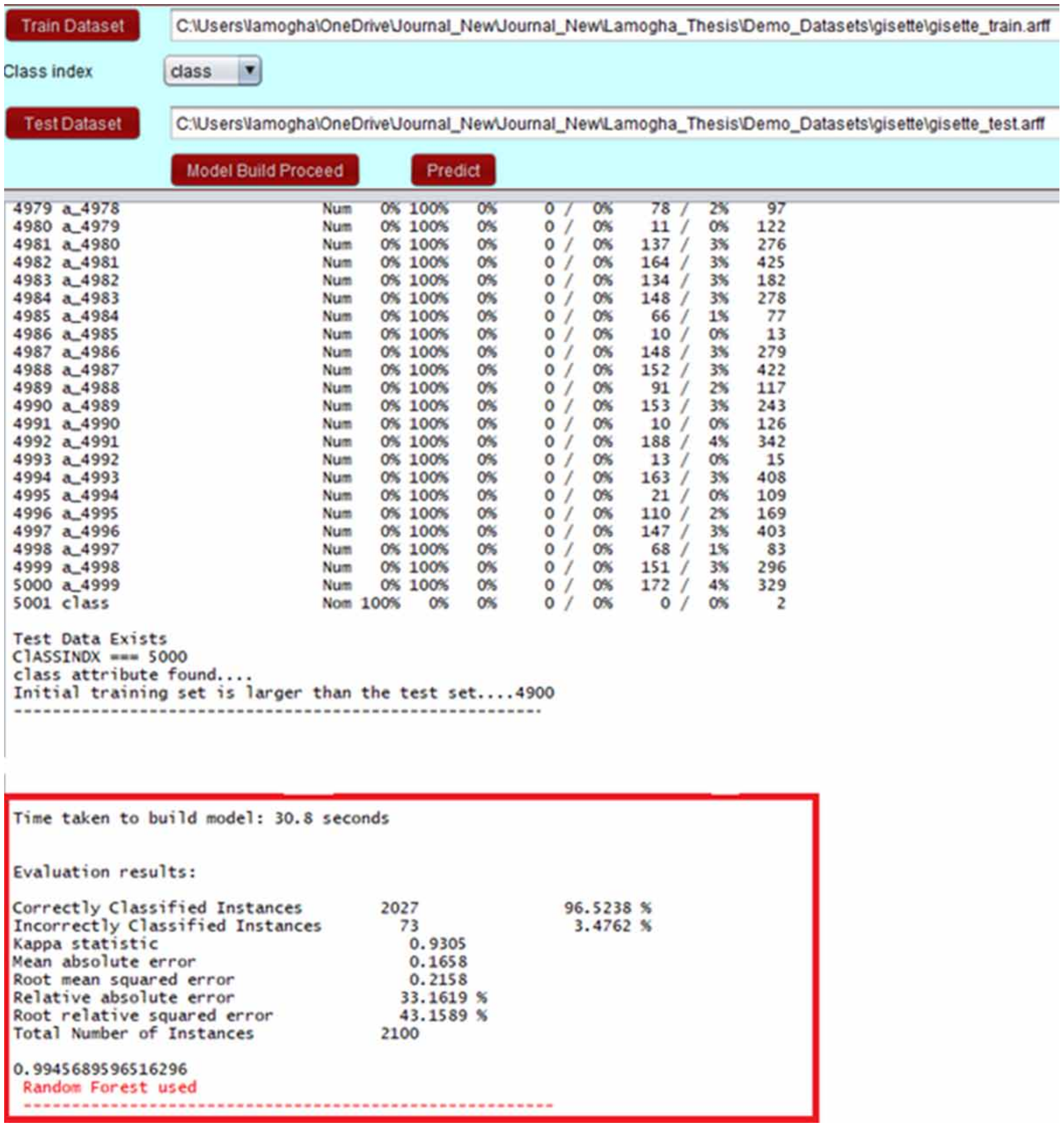
2.5. Use Case 5 (Location With Multi-Varying Data sets)

We supply a location containing multi-varying domain datasets. This use case aims to prove that the hybrid autoML system designed in this research can allow for the automatic model selections for multiple varying datasets in one go as clearly set out to achieve in the aims and objectives. This proves the contribution of the hybrid system been able to handle multiple multi-domain datasets on the fly, while distinctively building and choosing the most appropriate model per dataset. It is expected that this is achieved in a lesser time than if the user was to supply one dataset at a time (which is a common major limitation of other auto ML systems such as autoWeka). The following figures and discussions below in this subsection, describes the datasets and the evaluation results from executing this use case.

Figure 11 above, also shows the evaluation results for the chosen random forest model on this data set.

Figure 12 above, also shows that the Naive Bayes classifier was selected for this data set, and that it had an accuracy performance of up to a 100%.

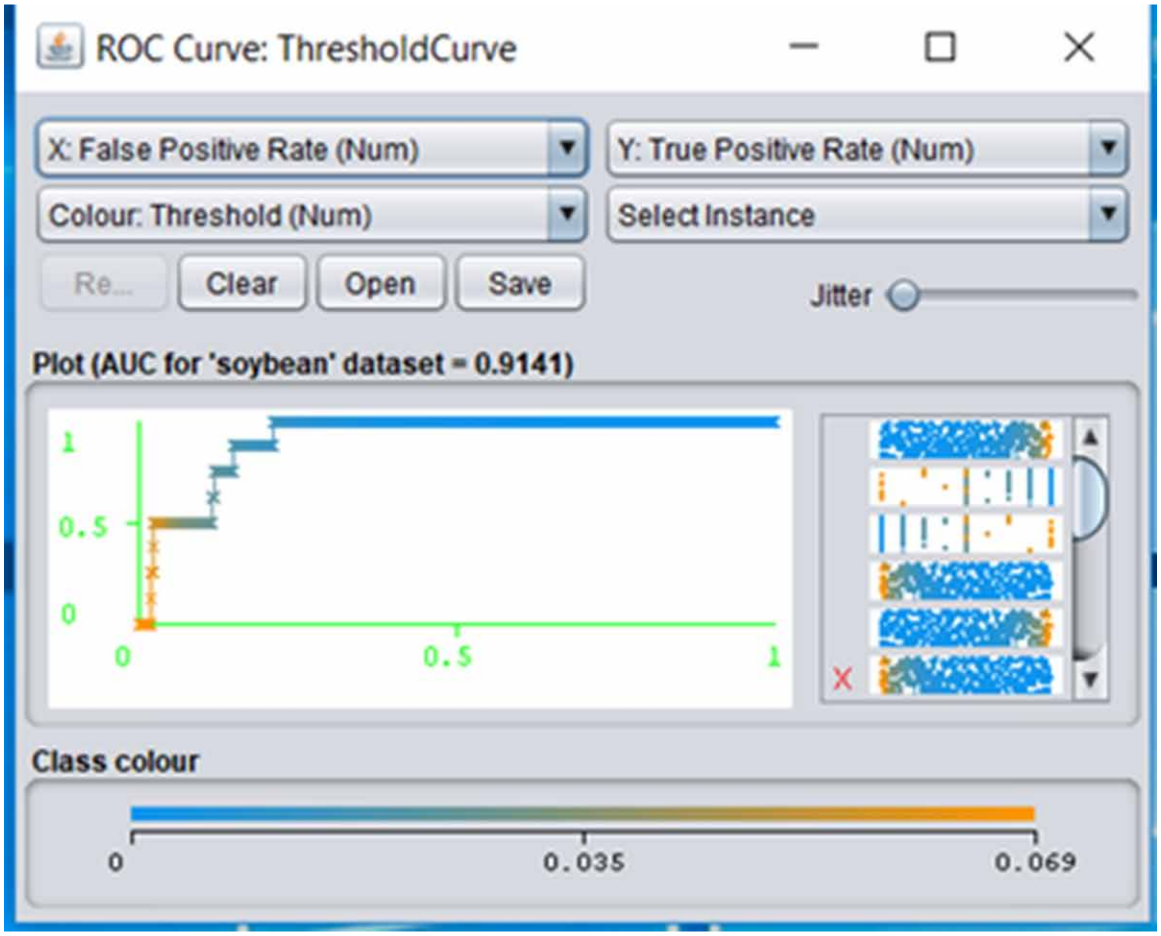
Figure 5. Shows the evaluation result obtained from using the hybrid autoML system on the 'gisette' data set



```
C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\gisette\gisette_train.arff
C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\gisette\gisette_test.arff
```

```
4979 a_4978          Num   0% 100%   0%    0 /  0%   78 /  2%   97
4980 a_4979          Num   0% 100%   0%    0 /  0%   11 /  0%  122
4981 a_4980          Num   0% 100%   0%    0 /  0%  137 /  3%  276
4982 a_4981          Num   0% 100%   0%    0 /  0%  164 /  3%  425
4983 a_4982          Num   0% 100%   0%    0 /  0%  134 /  3%  182
4984 a_4983          Num   0% 100%   0%    0 /  0%  148 /  3%  278
4985 a_4984          Num   0% 100%   0%    0 /  0%   66 /  1%   77
4986 a_4985          Num   0% 100%   0%    0 /  0%   10 /  0%   13
4987 a_4986          Num   0% 100%   0%    0 /  0%  148 /  3%  279
4988 a_4987          Num   0% 100%   0%    0 /  0%  152 /  3%  422
4989 a_4988          Num   0% 100%   0%    0 /  0%   91 /  2%  117
4990 a_4989          Num   0% 100%   0%    0 /  0%  153 /  3%  243
4991 a_4990          Num   0% 100%   0%    0 /  0%   10 /  0%  126
4992 a_4991          Num   0% 100%   0%    0 /  0%  188 /  4%  342
4993 a_4992          Num   0% 100%   0%    0 /  0%   13 /  0%   15
4994 a_4993          Num   0% 100%   0%    0 /  0%  163 /  3%  408
4995 a_4994          Num   0% 100%   0%    0 /  0%   21 /  0%  109
4996 a_4995          Num   0% 100%   0%    0 /  0%  110 /  2%  169
4997 a_4996          Num   0% 100%   0%    0 /  0%  147 /  3%  403
4998 a_4997          Num   0% 100%   0%    0 /  0%   68 /  1%   83
4999 a_4998          Num   0% 100%   0%    0 /  0%  151 /  3%  296
5000 a_4999          Num   0% 100%   0%    0 /  0%  172 /  4%  329
5001 class           Nom 100%   0%   0%    0 /  0%    0 /  0%    2

Test Data Exists
ClASSINDX === 5000
class attribute found....
Initial training set is larger than the test set....4900
-------------------------------------------------------

Time taken to build model: 30.8 seconds


Evaluation results:

Correctly Classified Instances        2027               96.5238 %
Incorrectly Classified Instances        73                3.4762 %
Kappa statistic                          0.9305
Mean absolute error                      0.1658
Root mean squared error                  0.2158
Relative absolute error                 33.1619 %
Root relative squared error             43.1589 %
Total Number of Instances             2100

0.9945689596516296
 Random Forest used
 -------------------------------------------------------
```

Figure 6. ROC curve obtained from training the model on the given train data set

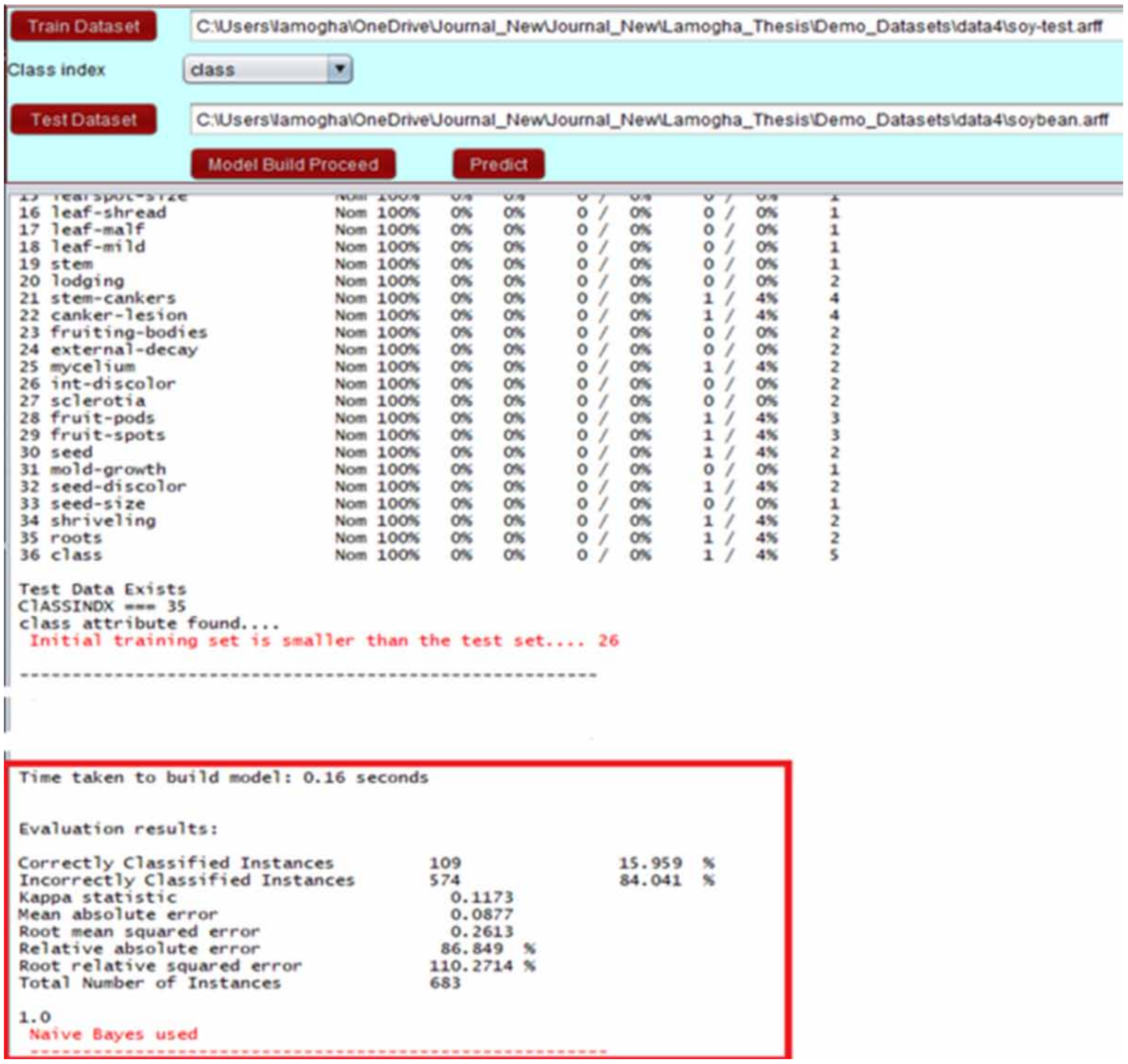

ROC Curve: ThresholdCurve
X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Re...
Clear
Open
Save
Jitter
Plot (AUC for 'soybean' dataset = 0.9141)
1
0.5
0
0
0.5
1
Class colour
0
0.035
0.069

From figure 13 above, it is shown that Random Forest was the classifier of choice used based on the rule-based algorithm designed in this research.

Figure 14 above, shows that the zeroR classifier was automatically chosen on the ’ReutersCorn training’ data set.

Figure 10 to Figure 15 above, shows various ML models been built automatically for the various datasets in our data location. These algorithms were chosen automatically by making use of both the functions for model selection and the function for handling multi datasets in one experimental run designed in this research. The highest time spent on any of the model built is 11.71 seconds. From executing this use case 5, we can conclude that using more general knowledge about a dataset such as the size of the training set compared with the test set, the class attribute type, the number of numeric in comparison to number of nominal attributes, etc. as a rule based functional algorithm, allows for the automatic ML mode and algorithmic model selection on multi-varying dataset as expected.

Figure 7. Evaluation metrics obtained from using a small trained data set and large test set in useCase4



Train Dataset: C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\data4\soy-test.arff

Class index: class

Test Dataset: C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\data4\soybean.arff

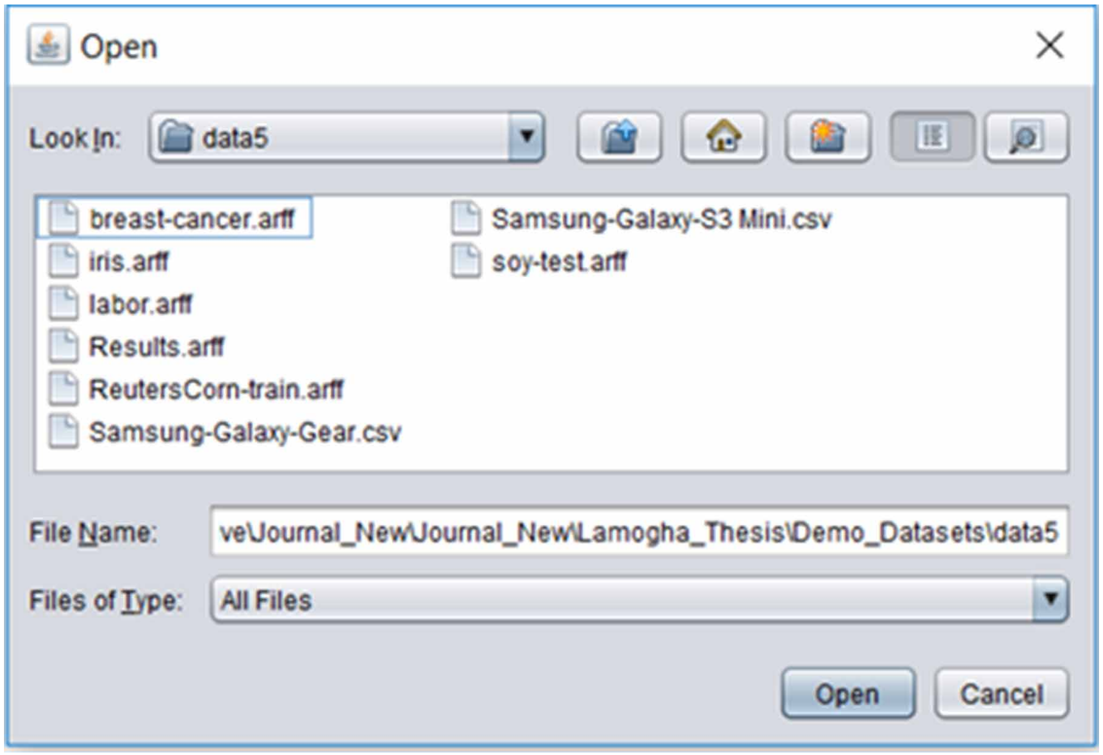
```
15 leafspot-size                 Nom 100%   0%   0%     0 /  0%     0 /  0%    1
16 leaf-shread                   Nom 100%   0%   0%     0 /  0%     0 /  0%    1
17 leaf-malf                     Nom 100%   0%   0%     0 /  0%     0 /  0%    1
18 leaf-mild                     Nom 100%   0%   0%     0 /  0%     0 /  0%    1
19 stem                          Nom 100%   0%   0%     0 /  0%     0 /  0%    1
20 lodging                       Nom 100%   0%   0%     0 /  0%     0 /  0%    2
21 stem-cankers                  Nom 100%   0%   0%     0 /  0%     1 /  4%    4
22 canker-lesion                 Nom 100%   0%   0%     0 /  0%     1 /  4%    4
23 fruiting-bodies               Nom 100%   0%   0%     0 /  0%     0 /  0%    2
24 external-decay                Nom 100%   0%   0%     0 /  0%     0 /  0%    2
25 mycelium                      Nom 100%   0%   0%     0 /  0%     1 /  4%    2
26 int-discolor                  Nom 100%   0%   0%     0 /  0%     0 /  0%    2
27 sclerotia                     Nom 100%   0%   0%     0 /  0%     0 /  0%    2
28 fruit-pods                    Nom 100%   0%   0%     0 /  0%     1 /  4%    3
29 fruit-spots                   Nom 100%   0%   0%     0 /  0%     1 /  4%    3
30 seed                          Nom 100%   0%   0%     0 /  0%     1 /  4%    2
31 mold-growth                   Nom 100%   0%   0%     0 /  0%     0 /  0%    1
32 seed-discolor                 Nom 100%   0%   0%     0 /  0%     1 /  4%    2
33 seed-size                     Nom 100%   0%   0%     0 /  0%     0 /  0%    1
34 shriveling                    Nom 100%   0%   0%     0 /  0%     1 /  4%    2
35 roots                         Nom 100%   0%   0%     0 /  0%     1 /  4%    2
36 class                         Nom 100%   0%   0%     0 /  0%     1 /  4%    5

Test Data Exists
ClASSINDX === 35
class attribute found....
 Initial training set is smaller than the test set.... 26

--------------------------------------------------------


Time taken to build model: 0.16 seconds


Evaluation results:

Correctly Classified Instances         109               15.959  %
Incorrectly Classified Instances       574               84.041  %
Kappa statistic                          0.1173
Mean absolute error                      0.0877
Root mean squared error                  0.2613
Relative absolute error                 86.849  %
Root relative squared error            110.2714 %
Total Number of Instances              683

1.0
 Naive Bayes used
 --------------------------------------------------------
```

2.6. Comparison of the Hybrid autoML with AutoWeka

In the following section, we present and evaluate the performance of the hybrid system with autoWeka (a state-of-the-art auto ML system). We base this comparison on multiple evaluation metrics mainly % accuracy, mean absolute error (MAE) the time in seconds. The aim of which is to prove that the hybrid system performs relatively better than autoWeka.

In Table 1 above, we can see the performance the hybrid autoML system designed in this research has in comparison to autoWeka. Three main metrics of evaluation are used here. They include the accuracy measured in percentage, the mean absolute errors (MAE) and the time in seconds. Data sets which have a numerical class attribute e.g. cpu dataset mainly generated a correlation coefficient on a scale of 0-1, which we then convert into a percentage score value to match up with the scale across other datasets for measuring accuracy (%). The bold numbers in the table shows where the hybrid autoML system performed better or relatively close enough to that of autoWeka. A star beside the classifier chosen by the hybrid model in the column 'chosen (by hybrid)' describes those datasets for which the hybrid model in this research outperformed autoWeka. This involved 14 out of the 33 datasets in the table having a higher performance in the hybrid-autoML system. 10 out of the 33 datasets performed relatively close to how autoWeka performed but with an advantage of been carried out in a lesser time than autoWeka. Which means that, for all target users of the system, the time spent in getting an idea of what algorithms to consider in the first instance is greatly reduced by using the hybrid autoML system designed in this research. Lastly, an important fact to add is that for using the autoWeka, each dataset had to be loaded in one after the other. While with the hybrid-autoML tool, the user only needs to supply a location for all the various datasets in question. Hence, reducing the effort and time of the user.

Figure 8. A file directory supplied as the location containing the varying data sets to be supplied in one run. It shows a total of 8 datasets that we use to test this user scenario



breast-cancer.arff
iris.arff
labor.arff
Results.arff
ReutersCorn-train.arff
Samsung-Galaxy-Gear.csv
Samsung-Galaxy-S3 Mini.csv
soy-test.arff

File Name: ve\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\data5

Files of Type: All Files

SUMMARY

In this chapter, we use a set of five different use cases and comparison analysis, to evaluate the performance unfolding of hybrid-autoML system and how it is used to achieve the goals set out in this research. Use case 1 shows the ability of the hybrid autoML system to select automatically an unsupervised learning strategy i.e. the 'autoProbClass' function given a small unlabelled dataset. While use case 2 shows unsupervised mode with a readily available EM clustering was selected automatically on a large unlabelled dataset. Use cases 3 to 4, shows that the system knows when to automatically use a supervised learning mode to select the most appropriate algorithm in the shortest time possible, on single or multi-varying dataset. All use cases thus proves that the system's function for mode and model selection (whether supervised or unsupervised) is effective and timely. Use case 5 establishes the fact that the system can effectively handle the supply of multiple datasets of varying types and from varying domains at a go. However, maintaining the integrity of using only the most suitable ML model per dataset. All of which means that the aims and objectives set out to be achieved by the modelling and design of the hybrid-autoML system has been effectively met. Lastly, a comparison of the system with autoWeka shows that in 24 out of 33 datasets, the hybrid system performs relatively better than autoWeka and in a way shorter time than autoWeka.

Figure 9. ROC curved obtained for five out of the 8 multi-varying data sets in our data location

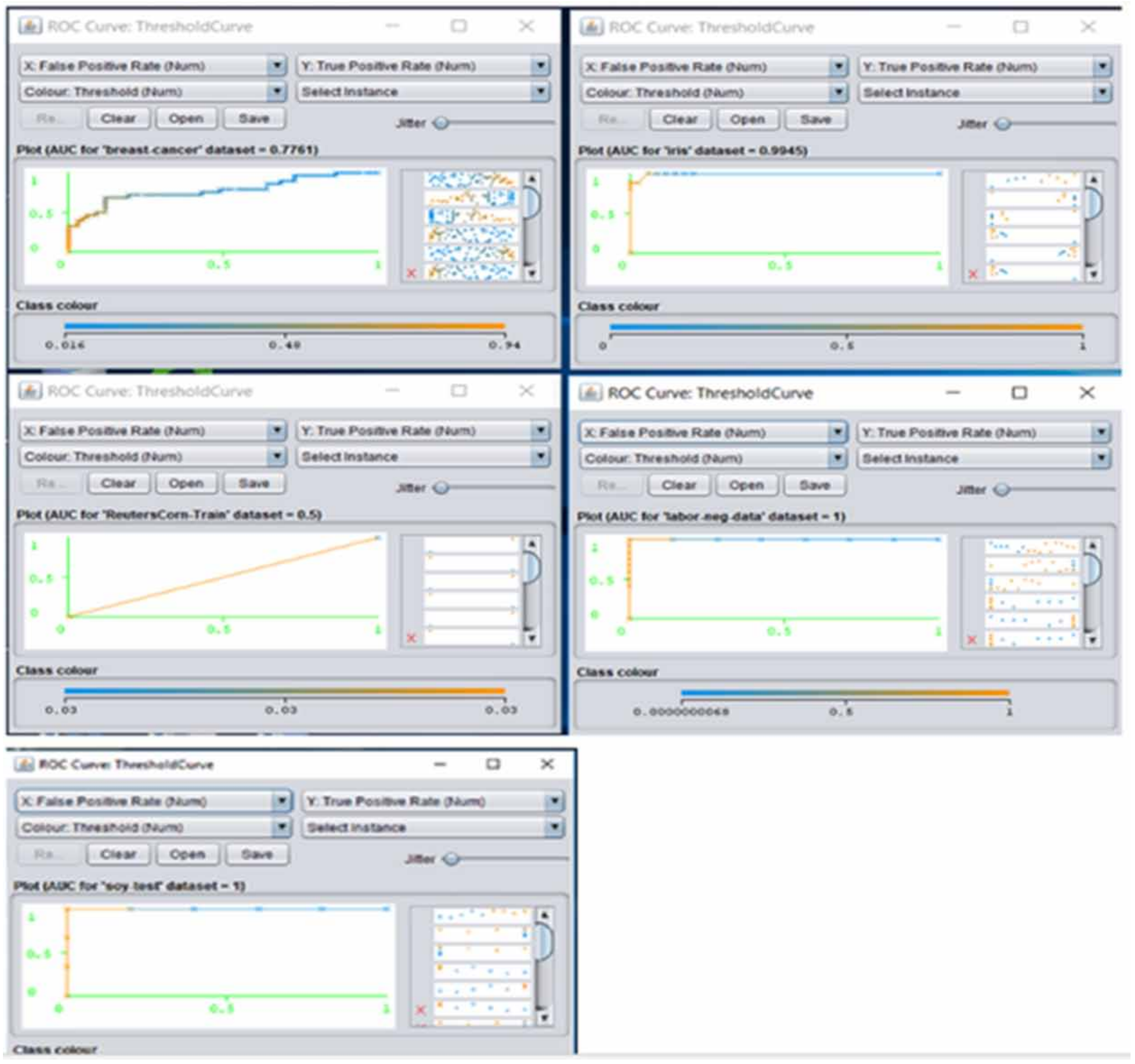
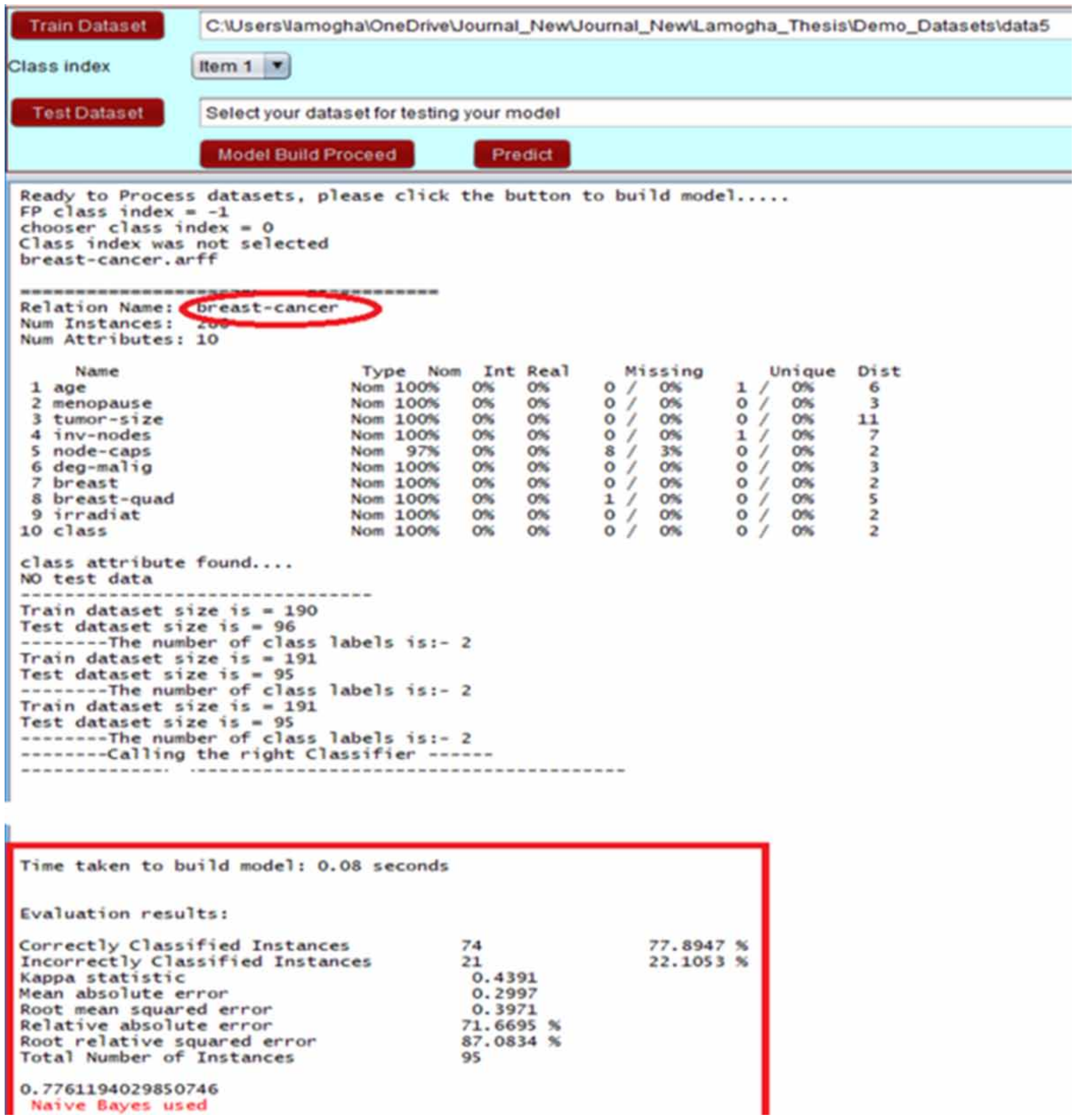
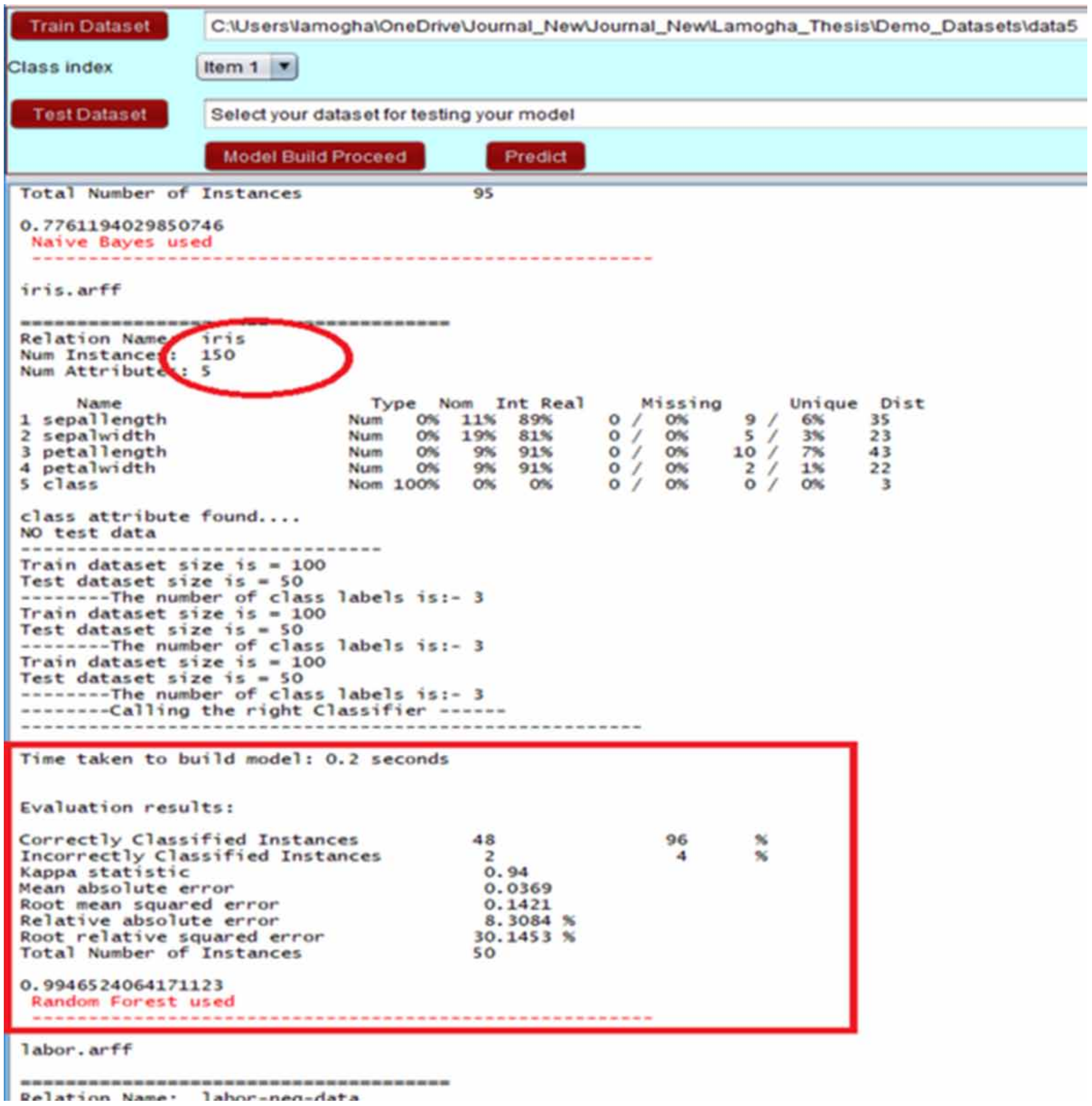

ROC Curve: ThresholdCurve
X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Plot (AUC for 'breast-cancer' dataset = 0.7761)
Plot (AUC for 'iris' dataset = 0.9945)
Plot (AUC for 'ReutersCorn-Train' dataset = 0.5)
Plot (AUC for 'labor.neg-data' dataset = 1)
Plot (AUC for 'soy.test' dataset = 1)

Figure 10. Shows the evaluation for the 'breast cancer' data set and Naive Bayes automatically chosen for it as the classified


Train Dataset
C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\data5
Class index
Item 1
Test Dataset
Select your dataset for testing your model
Model Build Proceed
Predict

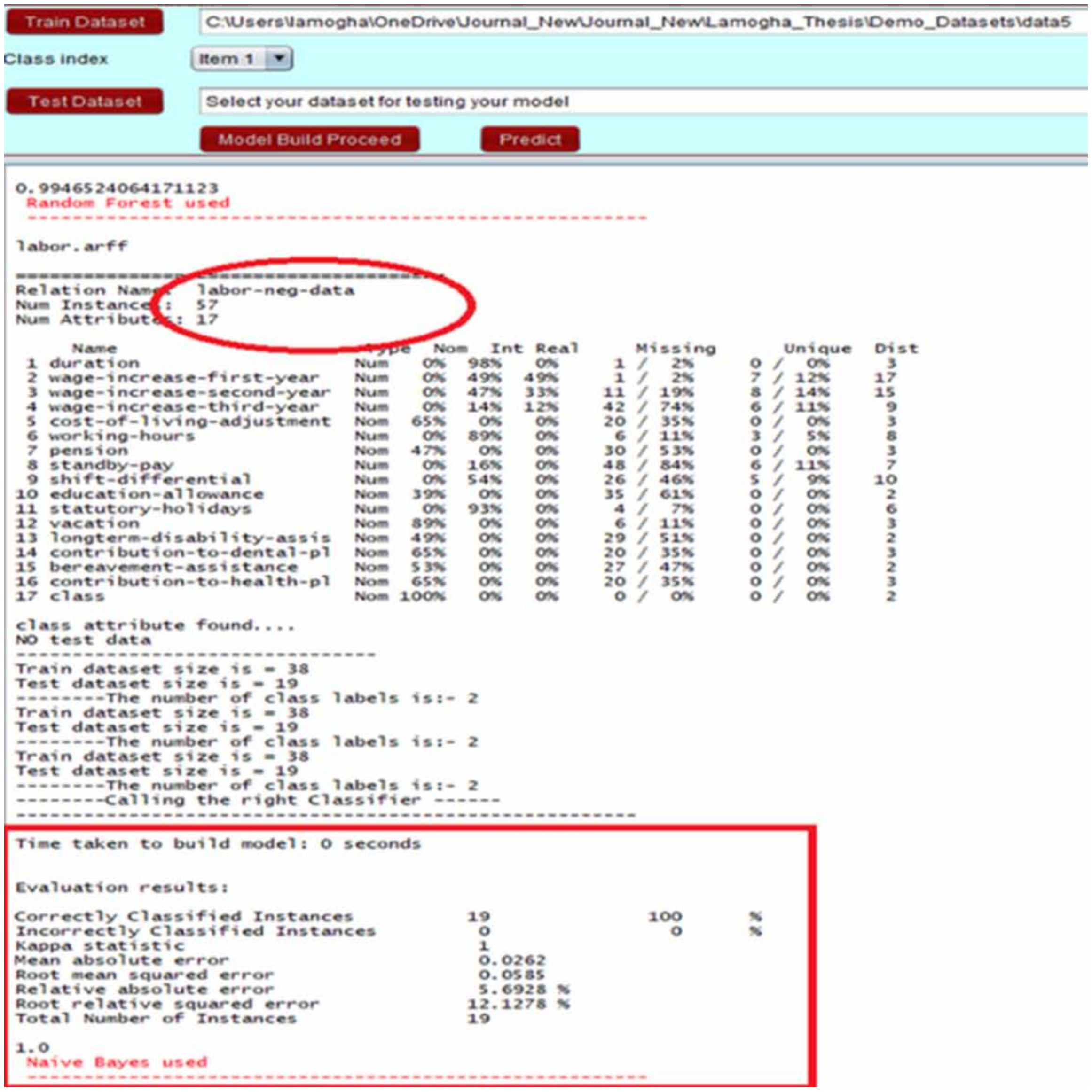
```
Ready to Process datasets, please click the button to build model.....
FP class index = -1
chooser class index = 0
Class index was not selected
breast-cancer.arff

==================================
Relation Name:  breast-cancer
Num Instances:  286
Num Attributes: 10

     Name                          Type  Nom  Int Real     Missing      Unique  Dist
 1 age                             Nom 100%   0%   0%     0 /  0%     1 /  0%     6
 2 menopause                       Nom 100%   0%   0%     0 /  0%     0 /  0%     3
 3 tumor-size                      Nom 100%   0%   0%     0 /  0%     0 /  0%    11
 4 inv-nodes                       Nom 100%   0%   0%     0 /  0%     1 /  0%     7
 5 node-caps                       Nom  97%   0%   0%     8 /  3%     0 /  0%     2
 6 deg-malig                       Nom 100%   0%   0%     0 /  0%     0 /  0%     3
 7 breast                          Nom 100%   0%   0%     0 /  0%     0 /  0%     2
 8 breast-quad                     Nom 100%   0%   0%     1 /  0%     0 /  0%     5
 9 irradiat                        Nom 100%   0%   0%     0 /  0%     0 /  0%     2
10 class                           Nom 100%   0%   0%     0 /  0%     0 /  0%     2

class attribute found....
NO test data
--------------------------------
Train dataset size is = 190
Test dataset size is = 96
--------The number of class labels is:- 2
Train dataset size is = 191
Test dataset size is = 95
--------The number of class labels is:- 2
Train dataset size is = 191
Test dataset size is = 95
--------The number of class labels is:- 2
--------Calling the right Classifier ------
-------------  ---------------------------------------

Time taken to build model: 0.08 seconds


Evaluation results:

Correctly Classified Instances          74               77.8947 %
Incorrectly Classified Instances        21               22.1053 %
Kappa statistic                          0.4391
Mean absolute error                      0.2997
Root mean squared error                  0.3971
Relative absolute error                 71.6695 %
Root relative squared error             87.0834 %
Total Number of Instances               95

0.7761194029850746
 Naive Bayes used
```

Figure 11. Shows that Random forest was chosen for the 'iris' dataset


Train Dataset
C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\data5
Class index
Item 1
Test Dataset
Select your dataset for testing your model
Model Build Proceed
Predict

```
Total Number of Instances              95

0.7761194029850746
 Naive Bayes used
 -------------------------------------------------------

iris.arff

======================================
Relation Name:  iris
Num Instances:  150
Num Attributes: 5

     Name                         Type  Nom  Int Real     Missing      Unique  Dist
1 sepallength                     Num   0%  11%  89%     0 /  0%     9 /  6%    35
2 sepalwidth                      Num   0%  19%  81%     0 /  0%     5 /  3%    23
3 petallength                     Num   0%   9%  91%     0 /  0%    10 /  7%    43
4 petalwidth                      Num   0%   9%  91%     0 /  0%     2 /  1%    22
5 class                           Nom 100%   0%   0%     0 /  0%     0 /  0%     3

class attribute found....
NO test data
--------------------------------
Train dataset size is = 100
Test dataset size is = 50
--------The number of class labels is:- 3
Train dataset size is = 100
Test dataset size is = 50
--------The number of class labels is:- 3
Train dataset size is = 100
Test dataset size is = 50
--------The number of class labels is:- 3
--------Calling the right Classifier ------
-------------------------------------------------------

Time taken to build model: 0.2 seconds


Evaluation results:

Correctly Classified Instances          48               96      %
Incorrectly Classified Instances         2                4      %
Kappa statistic                          0.94
Mean absolute error                      0.0369
Root mean squared error                  0.1421
Relative absolute error                  8.3084 %
Root relative squared error             30.1453 %
Total Number of Instances               50

0.9946524064171123
 Random Forest used
 -------------------------------------------------------

labor.arff

======================================
Relation Name:  labor-neg-data
```

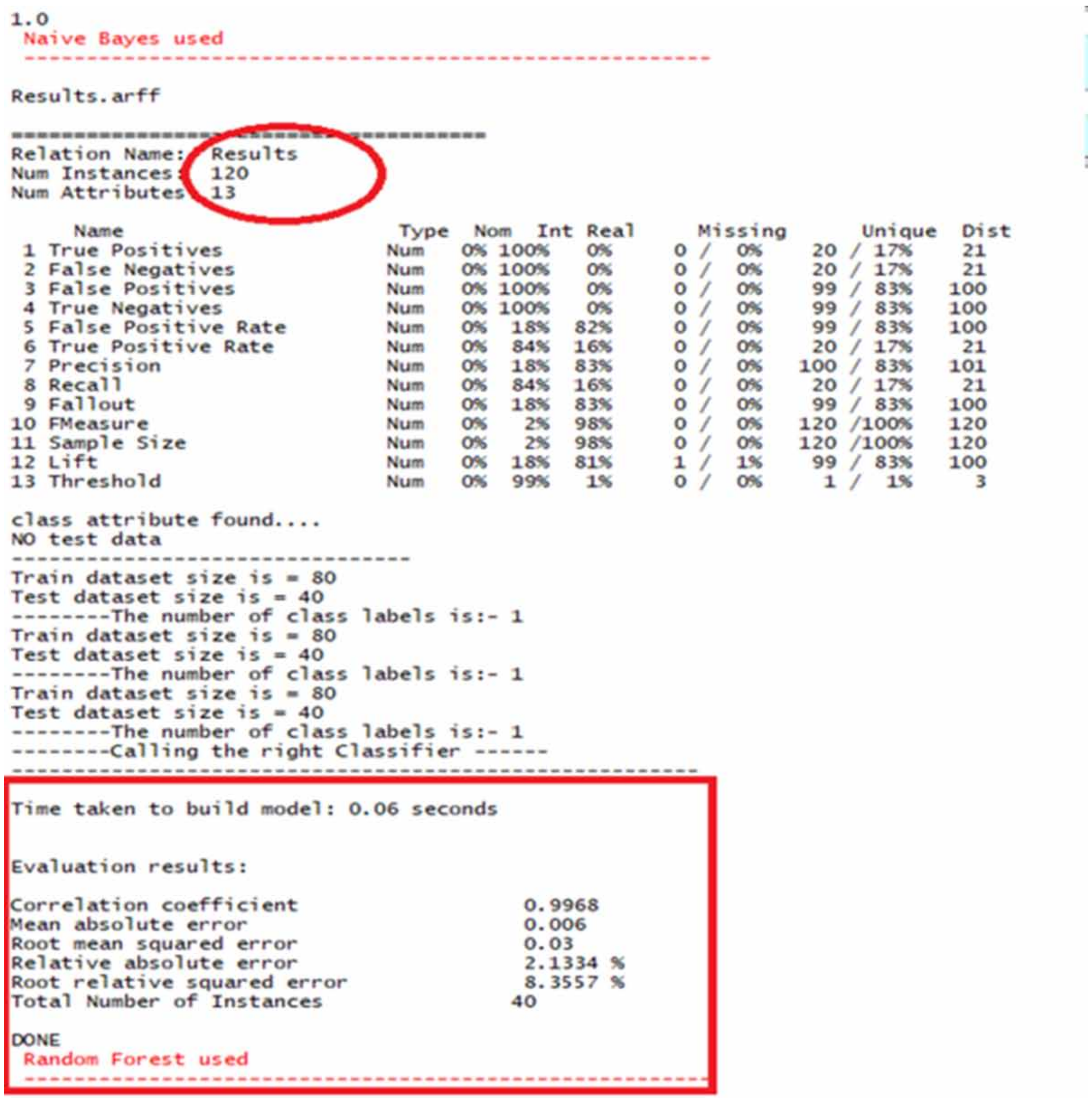
Figure 12. Evaluation results shown for the 'labour' data set



```
C:\Users\lamogha\OneDrive\Journal_New\Journal_New\Lamogha_Thesis\Demo_Datasets\data5
Class index   Item 1
Select your dataset for testing your model
```

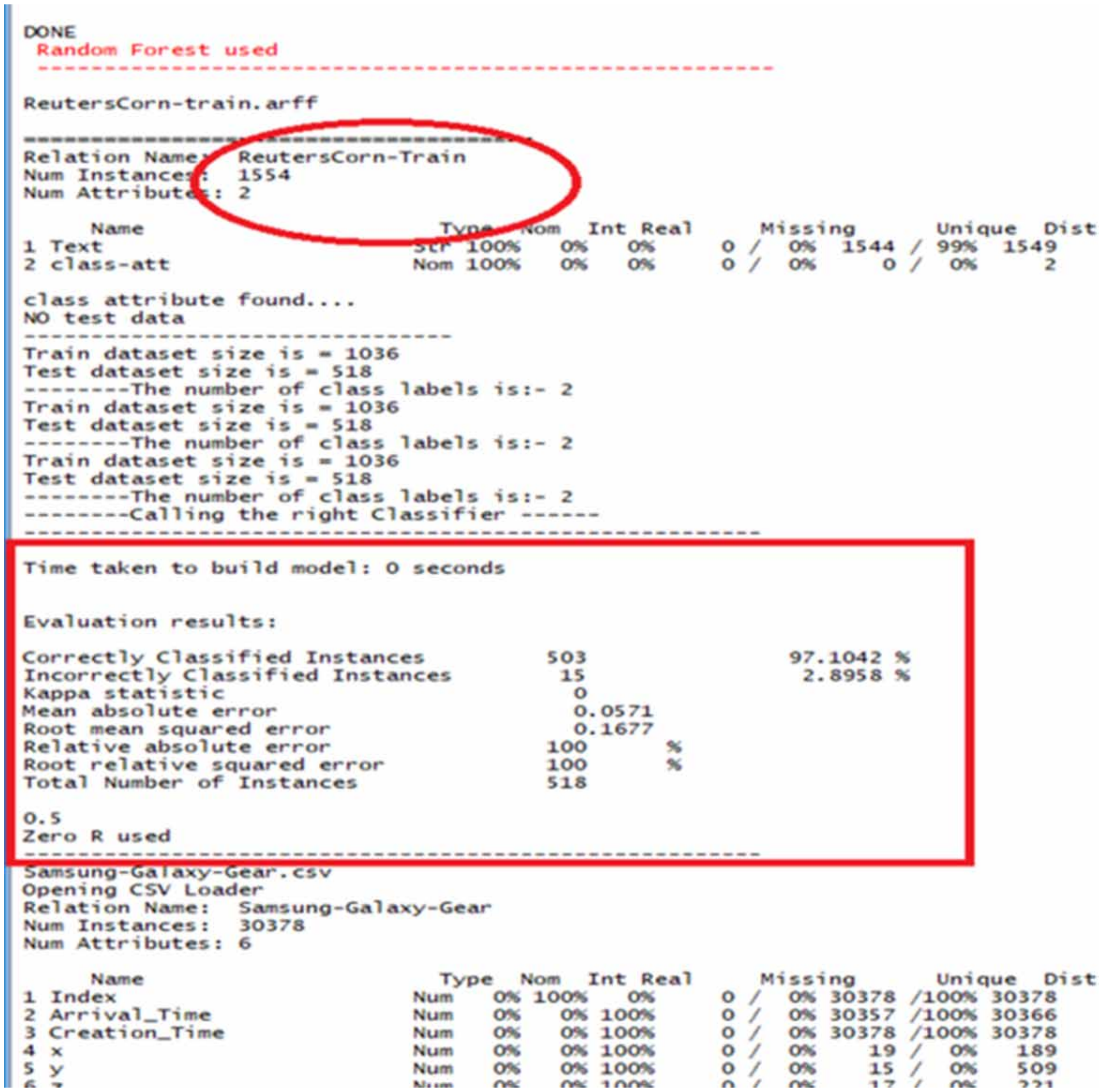
```
0.9946524064171123
 Random Forest used
 ----------------------------------------------------------

labor.arff

==================================================
Relation Name:  labor-neg-data
Num Instances:  57
Num Attributes: 17

     Name                            Type  Nom  Int Real     Missing      Unique  Dist
 1 duration                          Num   0%  98%   0%     1 /  2%     0 /  0%     3
 2 wage-increase-first-year          Num   0%  49%  49%     1 /  2%     7 / 12%    17
 3 wage-increase-second-year         Num   0%  47%  33%    11 / 19%     8 / 14%    15
 4 wage-increase-third-year          Num   0%  14%  12%    42 / 74%     6 / 11%     9
 5 cost-of-living-adjustment         Nom  65%   0%   0%    20 / 35%     0 /  0%     3
 6 working-hours                     Num   0%  89%   0%     6 / 11%     3 /  5%     8
 7 pension                           Nom  47%   0%   0%    30 / 53%     0 /  0%     3
 8 standby-pay                       Num   0%  16%   0%    48 / 84%     6 / 11%     7
 9 shift-differential                Num   0%  54%   0%    26 / 46%     5 /  9%    10
10 education-allowance               Nom  39%   0%   0%    35 / 61%     0 /  0%     2
11 statutory-holidays                Num   0%  93%   0%     4 /  7%     0 /  0%     6
12 vacation                          Nom  89%   0%   0%     6 / 11%     0 /  0%     3
13 longterm-disability-assis         Nom  49%   0%   0%    29 / 51%     0 /  0%     2
14 contribution-to-dental-pl         Nom  65%   0%   0%    20 / 35%     0 /  0%     3
15 bereavement-assistance            Nom  53%   0%   0%    27 / 47%     0 /  0%     2
16 contribution-to-health-pl         Nom  65%   0%   0%    20 / 35%     0 /  0%     3
17 class                             Nom 100%   0%   0%     0 /  0%     0 /  0%     2

class attribute found....
NO test data
--------------------------------
Train dataset size is = 38
Test dataset size is = 19
--------The number of class labels is:- 2
Train dataset size is = 38
Test dataset size is = 19
--------The number of class labels is:- 2
Train dataset size is = 38
Test dataset size is = 19
--------The number of class labels is:- 2
--------Calling the right Classifier ------
-------------------------------------------------------
Time taken to build model: 0 seconds


Evaluation results:

Correctly Classified Instances          19               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0262
Root mean squared error                  0.0585
Relative absolute error                  5.6928 %
Root relative squared error             12.1278 %
Total Number of Instances               19

1.0
 Naive Bayes used
 ----------------------------------------------------------
```

Figure 13. Evaluation result for the 'Results' data set



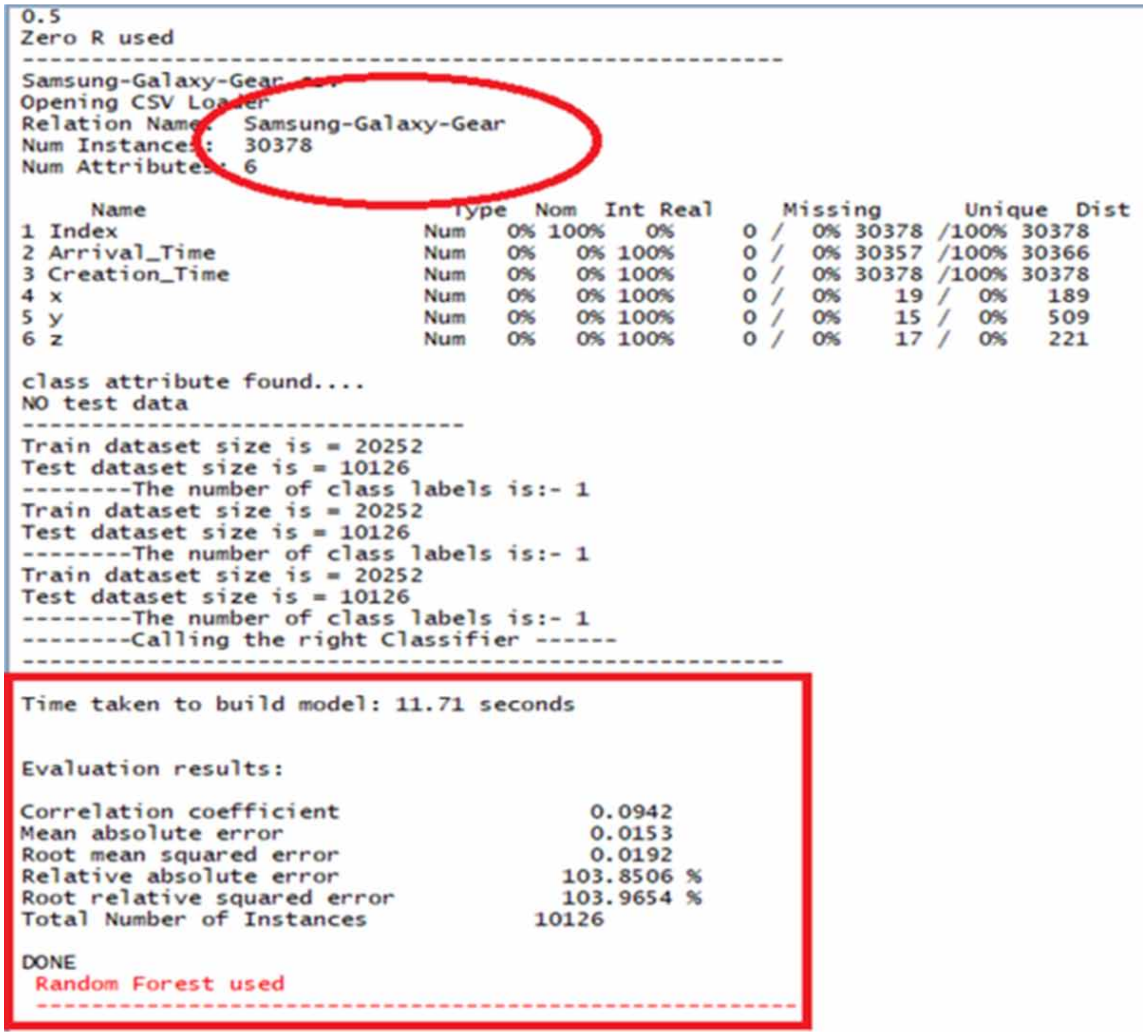
```
1.0
 Naive Bayes used
 -----------------------------------------------------

Results.arff

======================================
Relation Name:  Results
Num Instances:  120
Num Attributes: 13

     Name                          Type  Nom  Int Real     Missing      Unique  Dist
  1 True Positives                 Num   0% 100%   0%     0 /  0%     20 / 17%    21
  2 False Negatives                Num   0% 100%   0%     0 /  0%     20 / 17%    21
  3 False Positives                Num   0% 100%   0%     0 /  0%     99 / 83%   100
  4 True Negatives                 Num   0% 100%   0%     0 /  0%     99 / 83%   100
  5 False Positive Rate            Num   0%  18%  82%     0 /  0%     99 / 83%   100
  6 True Positive Rate             Num   0%  84%  16%     0 /  0%     20 / 17%    21
  7 Precision                      Num   0%  18%  83%     0 /  0%    100 / 83%   101
  8 Recall                         Num   0%  84%  16%     0 /  0%     20 / 17%    21
  9 Fallout                        Num   0%  18%  83%     0 /  0%     99 / 83%   100
 10 FMeasure                       Num   0%   2%  98%     0 /  0%    120 /100%   120
 11 Sample Size                    Num   0%   2%  98%     0 /  0%    120 /100%   120
 12 Lift                           Num   0%  18%  81%     1 /  1%     99 / 83%   100
 13 Threshold                      Num   0%  99%   1%     0 /  0%      1 /  1%     3

class attribute found....
NO test data
--------------------------------
Train dataset size is = 80
Test dataset size is = 40
--------The number of class labels is:- 1
Train dataset size is = 80
Test dataset size is = 40
--------The number of class labels is:- 1
Train dataset size is = 80
Test dataset size is = 40
--------The number of class labels is:- 1
--------Calling the right Classifier ------
-------------------------------------------------------
Time taken to build model: 0.06 seconds


Evaluation results:

Correlation coefficient                  0.9968
Mean absolute error                      0.006
Root mean squared error                  0.03
Relative absolute error                  2.1334 %
Root relative squared error              8.3557 %
Total Number of Instances               40

DONE
 Random Forest used
 -----------------------------------------------------
```

Figure 14. Evaluation results for the ‘ReutersCornTrain’ data set



```
DONE
 Random Forest used
 ------------------------------------------------------------

ReutersCorn-train.arff

=========================================================
Relation Name:  ReutersCorn-Train
Num Instances:  1554
Num Attributes: 2

     Name                          Type  Nom  Int Real     Missing      Unique  Dist
1 Text                             Str 100%   0%   0%     0 /  0%  1544 / 99%  1549
2 class-att                        Nom 100%   0%   0%     0 /  0%     0 /  0%     2

class attribute found....
NO test data
--------------------------------
Train dataset size is = 1036
Test dataset size is = 518
--------The number of class labels is:- 2
Train dataset size is = 1036
Test dataset size is = 518
--------The number of class labels is:- 2
Train dataset size is = 1036
Test dataset size is = 518
--------The number of class labels is:- 2
--------Calling the right Classifier ------
-------------------------------------------------------

Time taken to build model: 0 seconds


Evaluation results:

Correctly Classified Instances         503               97.1042 %
Incorrectly Classified Instances        15                2.8958 %
Kappa statistic                          0
Mean absolute error                      0.0571
Root mean squared error                  0.1677
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              518

0.5
Zero R used
-------------------------------------------------------
Samsung-Galaxy-Gear.csv
Opening CSV Loader
Relation Name:  Samsung-Galaxy-Gear
Num Instances:  30378
Num Attributes: 6

     Name                          Type  Nom  Int Real     Missing      Unique  Dist
1 Index                            Num   0% 100%   0%     0 /  0% 30378 /100% 30378
2 Arrival_Time                     Num   0%   0% 100%     0 /  0% 30357 /100% 30366
3 Creation_Time                    Num   0%   0% 100%     0 /  0% 30378 /100% 30378
4 x                                Num   0%   0% 100%     0 /  0%    19 /  0%   189
5 y                                Num   0%   0% 100%     0 /  0%    15 /  0%   509
6 z                                Num   0%   0% 100%     0 /  0%    17 /  0%   221
```

Figure 15. Evaluation results showing that Random Forest classifier isautomatically used to build the model for the 'Samsung-Galaxy-Gear' data set



```
0.5
Zero R used
--------------------------------------------------------
Samsung-Galaxy-Gear.csv
Opening CSV Loader
Relation Name:  Samsung-Galaxy-Gear
Num Instances:  30378
Num Attributes: 6

     Name                       Type  Nom  Int Real     Missing      Unique  Dist
1 Index                         Num    0% 100%   0%     0 /  0% 30378 /100% 30378
2 Arrival_Time                  Num    0%   0% 100%     0 /  0% 30357 /100% 30366
3 Creation_Time                 Num    0%   0% 100%     0 /  0% 30378 /100% 30378
4 x                             Num    0%   0% 100%     0 /  0%    19 /  0%   189
5 y                             Num    0%   0% 100%     0 /  0%    15 /  0%   509
6 z                             Num    0%   0% 100%     0 /  0%    17 /  0%   221

class attribute found....
NO test data
--------------------------------
Train dataset size is = 20252
Test dataset size is = 10126
--------The number of class labels is:- 1
Train dataset size is = 20252
Test dataset size is = 10126
--------The number of class labels is:- 1
Train dataset size is = 20252
Test dataset size is = 10126
--------The number of class labels is:- 1
--------Calling the right Classifier ------
--------------------------------------------------------

Time taken to build model: 11.71 seconds


Evaluation results:

Correlation coefficient                  0.0942
Mean absolute error                      0.0153
Root mean squared error                  0.0192
Relative absolute error                103.8506 %
Root relative squared error            103.9654 %
Total Number of Instances            10126

DONE
 Random Forest used
 --------------------------------------------------------
```

Table 1. Comparing autoWeka and the Hybrid autoML designed in this research

		ACCURACY (%)		ACCURACY(%)	MAE	MAE	TIME (secs)	TIME (secs)
Dataset	Chosen (by AutoWeka)	AutoWeka	Chosen(by Hybrid)	Hybrid Model	AutoWeka	Hybrid Model	AutoWeka	Hybrid Model
contact-lenses	DecisionTable	70.83	J48*	**87.50**	0.27	**0.10**	762.28	**0.14**
cpu	AdditiveRegression	93.53	RandomForest	**91.80**	31.72	**31.10**	765.88	**0.3**
cpu.with.vendor	MultiLayer Perceptron	99.96	RandomForest	**98.91**	4.99	12.24	769.69	**0.27**
credit-g	RandomForest	70.30	RandomForest*	**73.57**	0.35	0.34	769.37	**0.36**
diabetes	Logistics	75.65	SGD*	**78.13**	0.29	**0.22**	863.28	**0.09**
glass	Lazy.IBK	76.17	RandomForest*	**81.69**	0.11	0.10	769.82	**0.32**
ionosphere	SMO	92.59	RandomForest*	**94.87**	0.12	0.14	758.77	**0.14**
iris.2D	AdaBoostM1	92.67	RandomForest*	**96.00**	0.07	**0.04**	762.11	**0.03**
labor	SMO	85.96	NaiveBayes*	**100.00**	0.18	**0.01**	756.39	**0**
reutersCorn-train	Cannot handle	-	ZeroR*	**97.88**	-	**0.05**	-	**0**
segment-challenge	RandomSubspace	97.27	RandomForest*	**98.00**	0.01	**0.02**	839.05	**0.42**
soybean	LWL	92.53	NaiveBayes*	**93.39**	0.02	**0.01**	1187.38	**0.01**
supermarket	DecisionTable	76.94	SGD	64.53	0.32	0.35	778.03	2.00
unbalanced	SMO	98.60	RandomForest	**98.60**	0.03	**0.03**	781.57	**0.13**
vote	RandomForest	95.63	NaiveBayes	**93.10**	0.08	**0.07**	761.3	**0**
weather.nominal	SMO	64.29	J48	50.00	0.36	0.50	765.9	0.00
weather.numeric	IBk	85.71	NaiveBayes	75.00	0.17	0.38	758.67	0.00
Dorothea	DecisionStump	93.29	RandomForest	**88.81**	0.12	**0.16**	74450.82	**60.93**
Yeast	IBk	59.10	AutoWEKA engaged*	**100.00**	0.10	**0.04**	759.63	762.25
Amazon	NaiveBayes	57.90	RandomForest	20.29	0.02	0.04	1107.2	18.7
Secom	Bagging	93.89	RandomForest*	**95.07**	0.12	**0.11**	770.18	**1.24**
Semeion	Logistics	100.00	RandomForest	**93.55**	0.00	0.09	988.31	**0.84**
Car	AttributeSelected	100.00	NaiveBayes	85.36	0.00	0.12	885.25	0.01
Madelon	lazy.IBk	100.00	RandomForest	61.88	0.01	0.48	770.4	1.63
KR-VS-KP	Tress.LMT	99.91	NaiveBayes	87.53	0.09	0.22	774.35	0.01
Abalone	Logistics	28.90	RandomForest	**23.51**	0.06	**0.06**	837.43	**2.48**
Wine Quality	Ibk	100.00	RandomForest	65.09	0.04	0.09	770.42	1.64
Waveform	SimpleLogistics	87.86	RandomForest	**85.59**	0.13	0.20	865.09	**1.87**
Gisette	RandomForest	99.57	RandomForest	**95.71**	0.03	0.17	1602.26	**11.46**
Convex	RandomForest	55.30	RandomForest*	**73.03**	0.48	**0.39**	822.3	**14.78**
Cifar-10-small	RandomForest	99.19	RandomForest	86.02	0.06	0.16	4376	45.51
Mnist Basic	RandomForest	99.83	RandomForest*	**99.89**	0.02	0.06	1139.34	**15.42**
Shuttle	RandomForest	99.87	AutoWEKA engaged	**99.86**	0.00	**0.00**	844.25	924.66

Chapter 13
Final Remarks and Further Work for the Hybrid-AutoML System

ABSTRACT

This chapter addresses that the various use cases have proved that the aims and contributions of this research to conceptualise, design, and develop a scalable and flexible toolkit for automatic big data ML mode and model selection, on single or multi-varying datasets has been achieved. A major benefit of the hybrid-autoML toolkit is that it reduces the time data scientists and researchers in the field spend, searching through the algorithm selections and hyper parameter space. This advantage was discussed in Section 5.2 where the authors compared the hybrid-autoML tool with autoWeka on about 35 datasets using measures such as accuracy, mean absolute error (MAE), and time.

1. CONCLUSION

In this research, we have presented a toolkit for automatic machine learning (ML) mode and model selection on single or multi-varying datasets.

First, the basic concept of big data ML, ML tools, the algorithm selection problem, the meta-learning (learning-to-learn) paradigm and automated machine learning (autoML) was discussed. We discussed that although some hybrid autoML systems exists, e.g. autoWeka and auto-Sklearn, they do not consider knowledge known about mode selection but focus mainly on the supervised learning space for model selection. Some on one hand do not determine the importance and influence that knowledge of data sets meta features have over the choice of selecting the best ML mode and model automatically. Lastly, none of the known autoML system allows for automatic mode and model selection on multi-varying datasets at the same time. However, the hybrid-autoML system and functions designed in this research eliminates all that by taking them into consideration appropriately.

DOI: 10.4018/978-1-7998-7316-7.ch013

Second, provides more details and discussions from the literatures, that show the link between big data classification or clustering, the Meta learning paradigm, and how generic knowledge obtained about a dataset or about supervised and unsupervised learning, can be used to design a set of functions for automatic ML mode selection and model building on single or multi-varying datasets.

Third, we show and discuss some preliminary experimentations carried out in this research, using Weka (a well known data mining tool in the research community). The purpose of the pre experiments carried out, was to prove, properly identify and define the problems identified from previous discussions of literatures reviewed in chapter 2. The knowledge gained from this pre experiments helped define the rules for the hybrid-autoML system's model and design. The rule based functions modelled, takes into account the execution semantics for automatic ML mode and model selection.

Fouth, reported on the implementation details of the hybrid-autoML, visualisations, simulations and analysis. More specifically, we discussed and showed the design architecture (design consisting of three layers), components, testing strategy and materials of hybrid-autoML, and provided the relevant algorithms.

The toolkit named hybrid-autoML is an open source project that can be retrieved from github and easily used or extended. Hybrid-autoML provides a simple graphical user interface that facilitates automated ML mode and models selection, visualisation or evaluation and prediction capabilities.

Fifth, we addressed the unfolding of hybrid-autoML by evaluating its performance using 5 practical use cases and well known statistical and non-statistical measures. Based on the performance results of the experiments, a variety of observations are made. For example, use case 1 in section 5.1.1 shows an unsupervised mode and a simple and lightweight autoProb clustering function desgined in this research is chosen authomatically, for building a model on a small unlabelled dataset. While use case 2 in section 5.1.2 shows an unsupervised ML mode with a readily available EM clustering algorithm selected automatically for building a model on a larger unlabeled dataset. Use cases 3 and 5 from sections 5.1.3 and 5.1.5, proves that the hybrid-autoML tool knows when to automatically use a supervised ML mode to build an appropriate model on multi-varying datasets in the shortest time possible as compared to conventional autoWeka.

In conclusion, the various use cases have proved that the aims and contributions of this research to conceptualise, design, and develop a scalable and flexible toolkit for automatic big data ML mode and model selection, on single or multi-varying datasets has been achieved. A major benefit of the hybrid-autoML toolkit is that it reduces the time data scientists and researchers in the field spend, searching through the algorithm selections and hyper parameter space. This advantage was discussed in section 5.2 where we compared the hybrid-autoML tool with autoWeka on about 35 datasets using measures such as: accuracy, mean absolute error (MAE) and time.

2. FUTURE WORK

- Expanding the rule based function for model selection to accommodate more practical use case scenarios and algorithms, to further improve the automatic decision learning process.
- Expand the rules to accomodate better automatic data cleansing strategies before the automatic mode and model selection is performed.
- Considering the challenges of big data, incorporate some big data processing methods such as parallel processing to further optimize the process.

- The hybrid-autoML system improvement. This can be achieved by including the hyper parameter space options for some algorithms, then implement this in the system to determine any improvements made.
- Perform new experiments in a less controlled environment by using an observational study methodology to analyse how users interact with the system on different big dataset.
- Improve and commercialise the functionalities and capabilities of the system.

About the Authors

Zhongyu Lu is a professor in the Department of Computer Science and is the research group leader of Information and System Engineering (ISE) in the Centre of High Intelligent Computing (CHIC. Her research interests include information retrieval research, image processing, data technology, mobile computing, and Internet technologies, etc. She published peer reviewed articles and books as author and co-author. She has been the UOH principle investigator for four recent EU interdisciplinary projects, e.g. computer science with psychology, mathematics, Law, Engineering, Physics, etc. Professor Lu has acted as the founder and a program chair for the International XML Technology Workshop for 11 years and serves as Chair of various international conferences. She is the founder and Editor in Chief of International Journal of Information Retrieval Research and serves as a BCS examiner of Database and Advanced Database Management Systems.

Qiang Xu has been Senior Lecturer in Mechanical/Automotive Engineering at the School of Computing and Engineering, since June 2013 to present. He is a specialist in computational creep damage mechanics. Previously, Dr Xu was Senior Lecturer in Mechanical/Manufacturing Engineering in the School of Science and Engineering at Teesside University from 2006 to 2013. In this role, Dr Xu supervised a number of PhD research projects and completed over 15 consultancy and grant applications. In addition he has held academic appointments at the Swansea Institute of Higher Education and as a Research Fellow and Senior Research Fellow at the University of Huddersfield and Research Associate at UMIST, Manchester, UK. Dr Xu's research work mainly on the computational modelling and analysis for engineering and his work has been cited worldwide by researchers in eight nations including China, the USA, Germany, India, Iran, Russia et al. Further information can be found at University website: https://www.hud.ac.uk/ourstaff/profile/index.php?staffid=1212.

Murad Al-Ragab is currently an Assistant Professor in Computer Science and IT at the College of Engineering in Abu University. He has academic teaching experience for several years in the fields of Computer Science and Software Engineering at the Higher Education. He had engaged as a faculty member, and a university IT Director. He holds a PhD in Computer Science from the University of Huddersfield, United Kingdom. His research interests focus on applying machine learning in the study of genetic data for cancer research. Further research interests span in the areas of Mobile Learning & Applications, Smart Cities Applications, and Computer Science Education. He has published several peer-reviewed papers in international journals and conferences.

Lamogha Chiazor is a Research Software Engineer at IBM Research and loves exploring her passion for innovations, improving quality of life, and enthusiasm to make an impact by bringing value to individuals, communities, and the world through automated technology. She has spent the majority of her career in the Software Technology Development industry, gaining experiences in areas such as Data Science and Artificial Intelligence, Graph based Causal Modelling and Analysis, Cognitive Intelligence, Software Testing (Manual and Automated) and Security and Privacy Of Things Computations. She previously held a Postdoctoral role with Newcastle University in collaboration with Clue Computing Limited. You can follow her work here https://www.linkedin.com/in/lamogha/.

Index