HYBRID AUTOMATED MACHINE LEARNING SYSTEM

FOR BIG DATA

By: Lamogha Ighoroje
School of Computing and Engineering
University of Huddersfield

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

September 30th, 2018

Copyright statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the "Copyright") and s/he has given The University of Huddersfield the right to use such copyright for any administrative, promotional, educational and/or teaching purposes.

ii. Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the University Library. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.

iii. The ownership of any patents, designs, trademarks and any and all other intellectual property rights except for the Copyright (the "Intellectual Property Rights") and any reproductions of copyright works, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions

Acknowledgements

I acknowledge God Almighty for making it possible for me to undergo this research.

I will like to take the opportunity to acknowledge and thank my supervisory team, led by Prof. Joan Lu for giving me the opportunity to undergo this PhD research, and develop my research skills and interests in big data machine learning.

Finally, and most importantly I am very grateful to my parents Prof. Mrs Ahbor D.A. Ighoroje and Engr. Mr Richard Ighoroje, who supported my studies financially and morally at every stage. I also appreciate my husband Raymond Chiazor for his love and encouragements and all my family and friends for their immense support throughout my research years at the University.

Abstract

A lot of machine learning (ML) models and algorithms exist and in designing classification systems, it is often a challenge looking for and selecting the best performing ML algorithm(s) to use for a dataset in a short period of time. Often, one must learn thoroughly about the data set structure and content, decide whether to use a supervised, semi-supervised or an unsupervised learning strategy, and then investigate, select or design via trial and error a classification or clustering algorithm that would work most accurately for that specific dataset. This can be quite a time consuming and tedious process. Additionally, a classification algorithm may not perform very well with a dataset as compared to using a clustering algorithm. Meta-learning (learning to learn) and automatic ML (autoML) are data mining-based formalisms for modelling evolving conventional ML functions and toolkit systems. The concept of modelling a decision tree-based combination of both formalisms as a Hybrid-AutoML toolkit extends that of traditional complex autoML systems.

In hybrid-autoML, single or multiple predictive models are built by combining a three-layered decision learning architecture for automatic learning mode and model selection, by engaging formalisms for selecting from a variety of supervised or unsupervised ML algorithms and generic meta information obtained from varying multi-datasets. The work presented in this thesis aims to study, conceptualize, design and develop this hybrid-autoML toolkit. By extending in the simplest form, some existing methodologies for the model training aspect of autoML systems. The theoretical and experimental development focuses on the extension of autoWeka and use of existing meta-learning, algorithm selection and decision tree concepts. It addresses the issue of efficient ML mode (supervised or unsupervised) and model selection for varying multi-datasets, learning methods representations of practical alternative use cases and structuring of layered decision ML unfolding, and algorithms for constructing the unfolding. The implementation aims to develop tools for hybrid-autoML based model visualization or evaluation, use case simulations and analysis on single or multi varying datasets. An open source tool called hybrid-autoML has been developed to support these functionalities. Hybrid-autoML provides a user-friendly graphical interface that facilitates single or multi varying datasets entry, supports automatic learning mode or strategy selection, automatic model selection on single or multi-varying datasets, supports predictive testing, and allows the automatic visualization and use of a set of analytical tools for model evaluation. It is highly extensible and saves a lot of time.

Table of Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

This chapter provides some background information, highlights into the motivations and problems resolved in this thesis and then discusses the aims and contributions of this thesis.

1.1 Background

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).

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 (Charu C. Aggarwal) and a random variable input **X** under consideration, determine correctly which label should be assigned to a new unlabelled instance of **X** (Charu C 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 2014, 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.

1.2 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 section 2.7.3 of this thesis, 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 in this thesis helps 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.

1.3 Aims and Contributions

Our research hypothesis is that the hybrid big data autoML model designed in this thesis, 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 'auto-Prob' 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.

1.4 Outline of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 presents a detailed literature review on big data machine learning, classification and clustering principles and tools (e.g. Weka) in the ML research community and presents detailed discussions on the algorithm selection problem and how meta-learning formalism can be used to help resolve the algorithm selection problem. Finally, it presents discussions on autoML and a comparison of some of the current state of the art autoML related works and tools.

Chapter 3 defines all methods used in this thesis and presents a detailed discussion of all pre-design experimentations including the setup, algorithms considered, problems identified, and knowledge gained during the experiments. The identified problems and knowledge gained in this chapter, served as the basis for the design and modelling in the next chapter.

Chapter 4 describes the Hybrid-AutoML system's design, architecture, components, and characteristics, and presents the theory and algorithms for Hybrid-AutoML based unfolding. It also outlines the design framework and describes additional tools that have been added or used for the models verification, simulation and analysis.

Chapter 5 discusses the results obtained and analyzed from using the Hybrid-AutoML toolkit on five different practical use cases.

Chapter 6 summarizes and concludes the work and proposes directions for further work.

1.5 List of Publications

Portions of the work within this thesis have been documented in the following publications:

Conferences/Workshops

1. Ighoroje, L., Lu, J., & Xu, Q. (2016). Hybrid classification system design using a decision learning approach and three-layered structure - A Meta learning paradigm in Data Mining. In J. Gołuchowski, M. Pańkowska, C. Barry, M. Lang, H. Linger, & C. Schneider (Eds.), Information Systems Development: Complexity in Information Systems Development (ISD2016 Proceedings). Katowice, Poland: University of Economics in Katowice. ISBN: 978-83-7875-307-0.

Summary

We have provided in this section some background introduction into designing big data classification system, shown what the values and research focus in this thesis are, discussed the aims and contributions made in this thesis, and provided an outline for the rest of this thesis. More specifically shown is that, in designing highly efficient and robust big data classification systems, the algorithm selection problem and the time data scientists spend in building ML models can be greatly reduced by engaging the sub fields of autoML and meta-learning. In addressing the limitations of some state of the art autoML systems discussed in the next chapter, this research thesis considers the following contributions:

1. An algorithmic function for automatic learning mode selection.
2. An algorithmic function for automatic clustering model selection, with a new added function into the mix of available Weka clusterers called autoProbClass for class clustering, using euclidean distance estimation.
3. A toolkit that supports authomatic ML model selection on single or varying multi-datasets, depending on the user scenario. Using 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.
4. Saves model build time for multi-datasets ML tasks.
5. Is highly extensible and flexible.

In the next chapter, we provide more detailed discussions on ML and autoML concepts, methods, techniques and tools from state-of-the-art literature reviews.

Chapter 2

2 Literature Review

Introduction

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 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.

2.1 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 opportunities 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 thesis.

2.1.1 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 thesis.

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 thesis incorporates concepts from an unsupervised probabilistic Bayesian classification method called autoClass (Cheeseman, 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 $class^1$ 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 thesis 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 thesis 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.

2.2 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.

2.2.1 Data Classification & 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:

2.2.1.1 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.

2.2.1.2 Testing phase:

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.

2.2.2Data 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 thesis 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.

2.2.3Classification 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.

2.2.3.1 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, & Liu, 2013; Forman, 2003; Haralick et al., 1973; A. K. Jain & Waller, 1978; Kwak & Choi, 2002; T. Li et al., 2004; Huiqing Liu, Li, & Wong, 2002; Huan Liu & Motoda, 1998; Huan Liu & Yu, 2005; Mladeni'c & Grobelnik, 1998; Pal & Foody, 2010; Peng, Long, & Ding, 2005; Punch III et al., 1993; Tang, Alelyani, & Liu, 2014; Zhao & Liu, 2007)

2.2.3.2 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 2.1 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)).

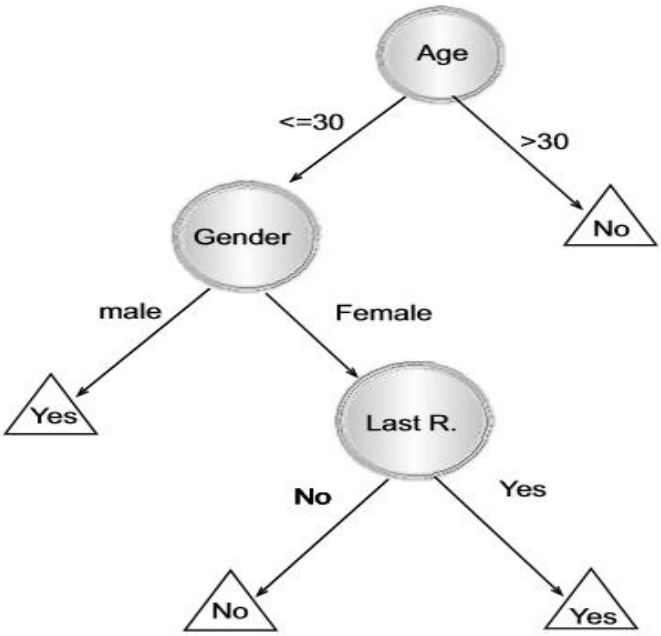


Figure 2.1: A simple decision tree that represents responses to direct mailing (Rokach & Maimon, 2010).

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, Malerba, Semeraro, & Kay, 1997; Lin, Yan, Yan, & Nan, 2008; Murthy, 1998; Nielsen, Rumí, & Salmerón, 2009; J. Ross Quinlan, 1986; Vens, Struyf, Schietgat, Džeroski, & Blockeel, 2008). Two very popular decision tree algorithms are the classification and regression trees (CART) (Breiman, Friedman, Stone, & Olshen, 1984; Loh, 2011) & the C4.5 algorithm (J Ross 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 thesis (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.

2.2.3.3 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; M. Jain et al., 2013; X.-L. Li & Liu; 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.

2.2.3.4 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 $p(\mathrm{C}|x)$ 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 $p(\mathrm{x}|C)$ for each class (C), after which the prior class probability $p(C)$ is then inferred and Bayes theorem used to determine $p(\mathrm{C}|x)$.
- By modelling the joint distribution $p(x, C)$ directly and then normalizing it to obtain the $p(\mathrm{C}|x)$.

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 hypothesis (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 (Lafferty, McCallum, & Pereira, 2001). More on probabilistic methods is discussed in (Bishop, 2006) and (Alsallakh, Hanbury, Hauser, Miksch, & Rauber, 2014; Azar & El-Said, 2013; Bankert, 1994; Iounousse et al., 2015; Lu et al., 2010; Lukasiewicz, 2008; Maravall, De Lope, & Fuentes, 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.

2.2.3.5 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; L. 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, Leisch, & Hornik, 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.

2.3 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:

2.3.1 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).

2.3.2 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, Tang, & Liu, 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.

2.3.3 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).

2.3.4Confusion 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.

2.3.5Discrete 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.

2.3.5.1 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:

$$acc = \frac{tp + tn}{tp + fp + tn + fn} \qquad (2.1)$$

2.3.5.2 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:

$$MAE = \frac{1}{n}\sum|X_o - X_p| \tag{2.2}$$

Where n = errors count, X_o = the observed value and X_p = the predicted value.

2.3.5.3 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:

$$Recall = \frac{tp}{tp + fn} \tag{2.3}$$

2.3.5.4 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?"

$$Precision = \frac{tp}{tp + fp} \tag{2.4}$$

2.3.5.5 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:

$$Specificity = \frac{tn}{fp + tn} \tag{2.5}$$

2.3.5.6 Fall-out

This is also known as the **false positive rate,** and is represented mainly as follows:

$$FallOut = 1 - specificity \tag{2.6}$$

2.3.5.7 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.

2.3.5.8 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.

$A \rightarrow \begin{matrix} t_p = 32 & f_p = 14 \\ f_n = 18 & t_n = 36 \end{matrix}$ with its tpr = 0.32 & fpr = 0.14.

$B \rightarrow \begin{matrix} t_p = 12 & f_p = 44 \\ f_n = 38 & t_n = 6 \end{matrix}$ with tpr = 0.12 & fpr = 0.44

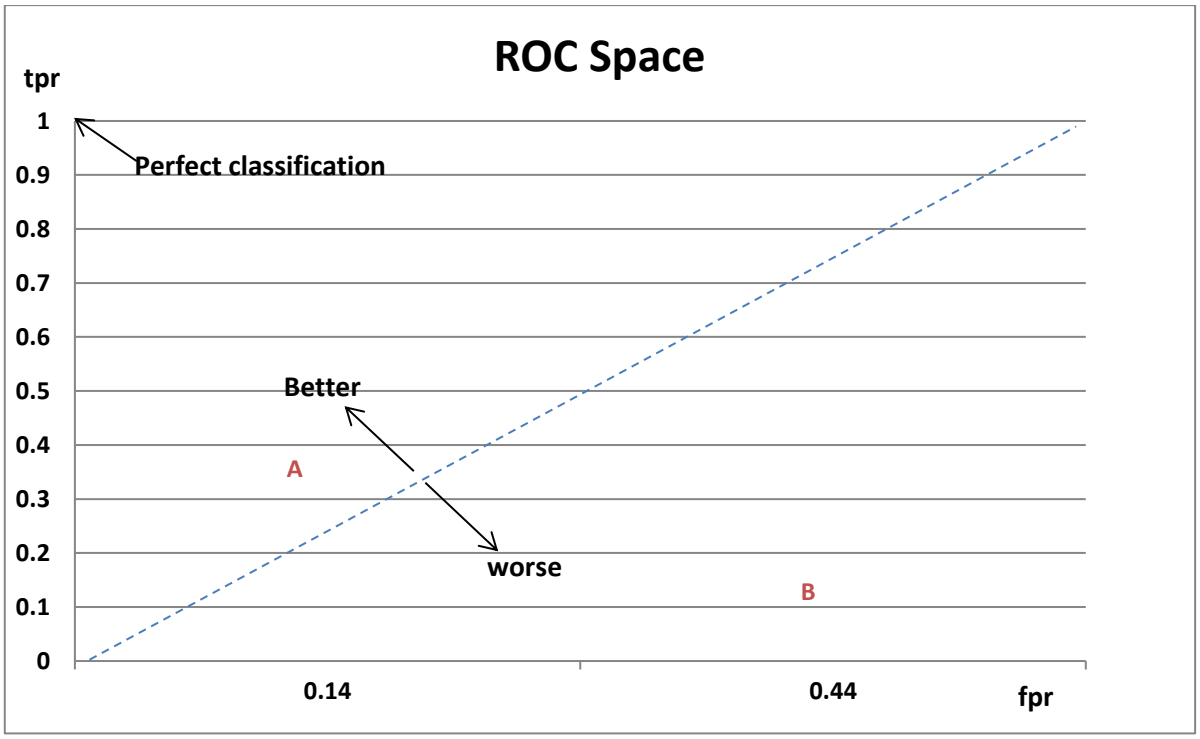


Figure 2.2: The ROC space and plots of the two prediction cases above.

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, Ramamohanarao, Karmakar, Hossain, & Bailey, 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:

2.3.6Integrity 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?"

2.3.7Simplicity

The simplicity of a model, shows "how easy it is to understand the model?" or "how uncomplicated the design of the model is?"

2.3.8Run 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.

2.3.9Reliability

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).

2.3.10 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).

2.4Classification Tools

There are severaldata mining tools that incorporate both data classification and clustering algorithms. However, this thesis 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, Anil, Dunning, & Friedman, 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**) (Francisci 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, Holmes, Kirkby, & Pfahringer, 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 2.1 below. The representation of what each column stands for in the table is shown below the table.

Table 2.1: 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	http://www.cs.waikato.ac.nz/ml/Weka/	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Mahout	http://mahout.apache.org/	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
SAMOA	https://samoa.incubator.apache.org/	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
MOA	http://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. ✓= Yes and ✗ = No.

2.5 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 hypothesis. 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) [10] 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 *selection 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 hypothesis discussed in this research thesis 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.

2.6 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, Sabourin, Cavalcanti, & Ren, 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 (Y. Yan, Zhu, Shyu, & Chen, 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 hypothesis that is then applied in an experiment and based on the performance results of the experiments a set of decision rules are drawn.

2.7 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, 2015) as:

Formal Definition: *For* $i = 1, \ldots, n + m$, *let* $x_i \in \mathbb{R}^d$ *signify a feature vector and* $y_i \in Y$ *the corresponding target value. Given a training dataset* $\mathcal{D}_{train} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ *and the feature vectors* $x_{n+1}, \ldots, x_{n+m}$ *of a test dataset* $\mathcal{D}_{test} = \{(x_{n+1}, y_{n+1}), \ldots, (x_{n+m}, y_{n+m})\}$ *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:*

$$\frac{1}{m} \sum_{j=1}^{m} \mathcal{L}\left(\dot{y}_{n+j}, \quad y_{n+j}\right) \tag{2.8}$$

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.

2.7.1 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.

2.7.2 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.

2.7.3 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, Thornton, Hoos, Hutter, & Leyton-Brown, 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.

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 thesis.

2.7.3.1 Summary

Table 2.2 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 2.2,

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.

Table 2.2: 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

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.

Chapter 3

3 Methodology

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 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.

3.1 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.

3.1.1 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 thesis.

3.1.2 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.

3.1.3 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 $\mathcal{D}$, 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 $\mathcal{D}$, without pre-labelled training data instances and the knowledge that unsupervised learners are best used when no pre-labelled training dataset exists.

3.2 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.

3.2.1 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 7 shows a representation of the GUI and other related tabs of the GUI for Weka.

3.2.2 Big Data

A variety of datasets collected from the UCI machine learning (Dua & Karra Taniskidou, 2017) & KDnuggets data repository, as well as from the Weka tool data and auto-Weka (Lars, Chris, Frank, Holger, & Kevin, 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 3. 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 3.1: 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

3.2.3 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:

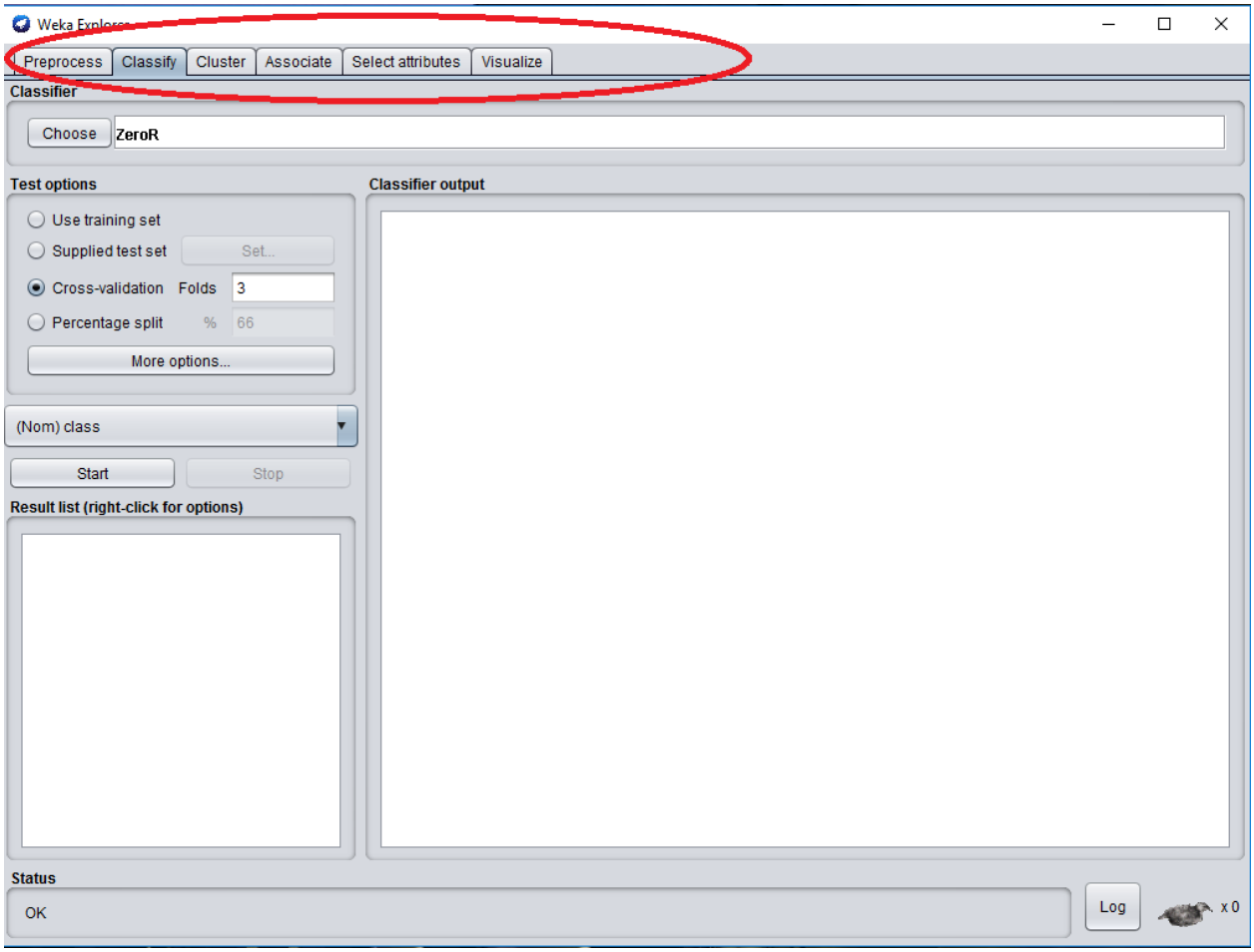


Figure 3.1: Weka explorer 'Classifier' tab.

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:

```
=== 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 3.2: Output Result window display for a ‘classifier’ in Weka.

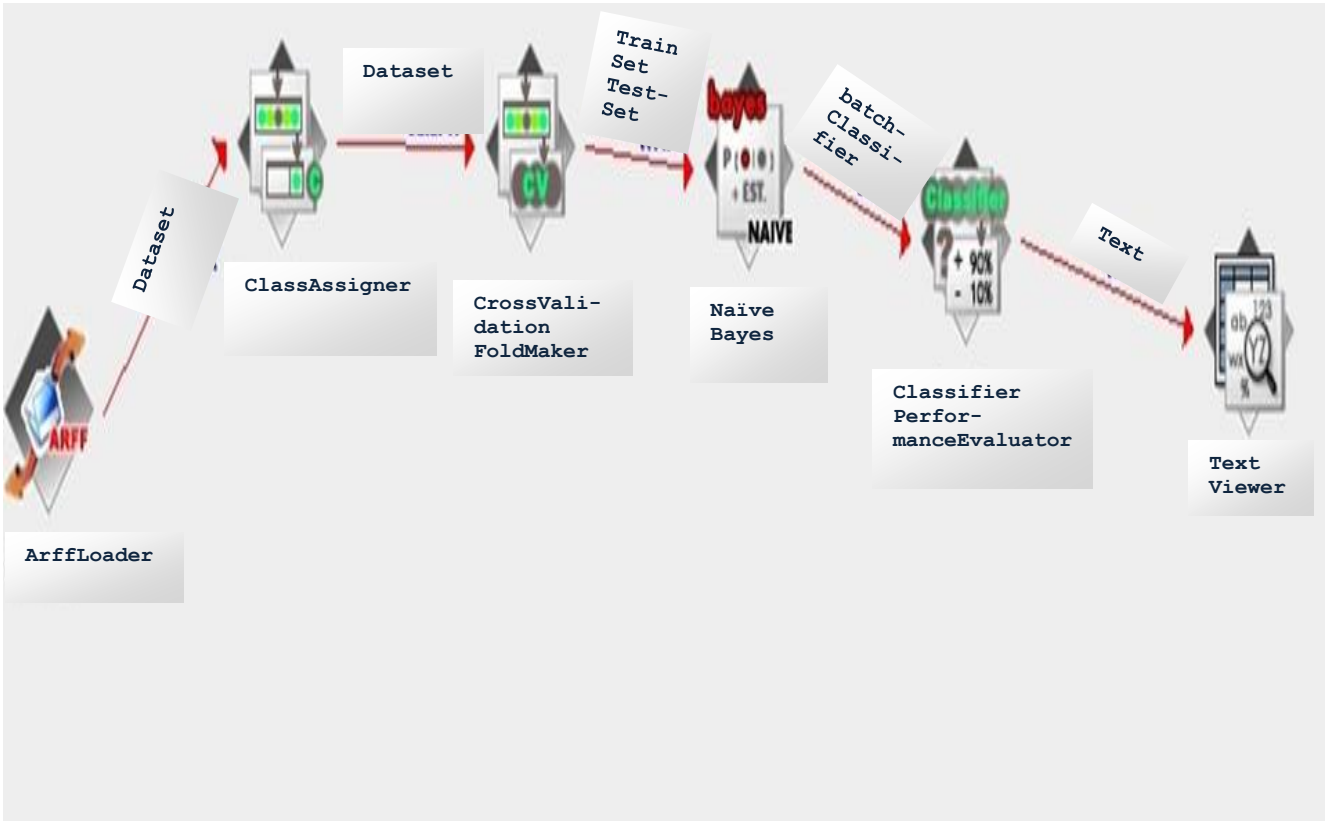


Figure 3.3: Example of a single supervised Learning knowledge flow setup in Weka.

In the experiment seen in Figure 3.3 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.

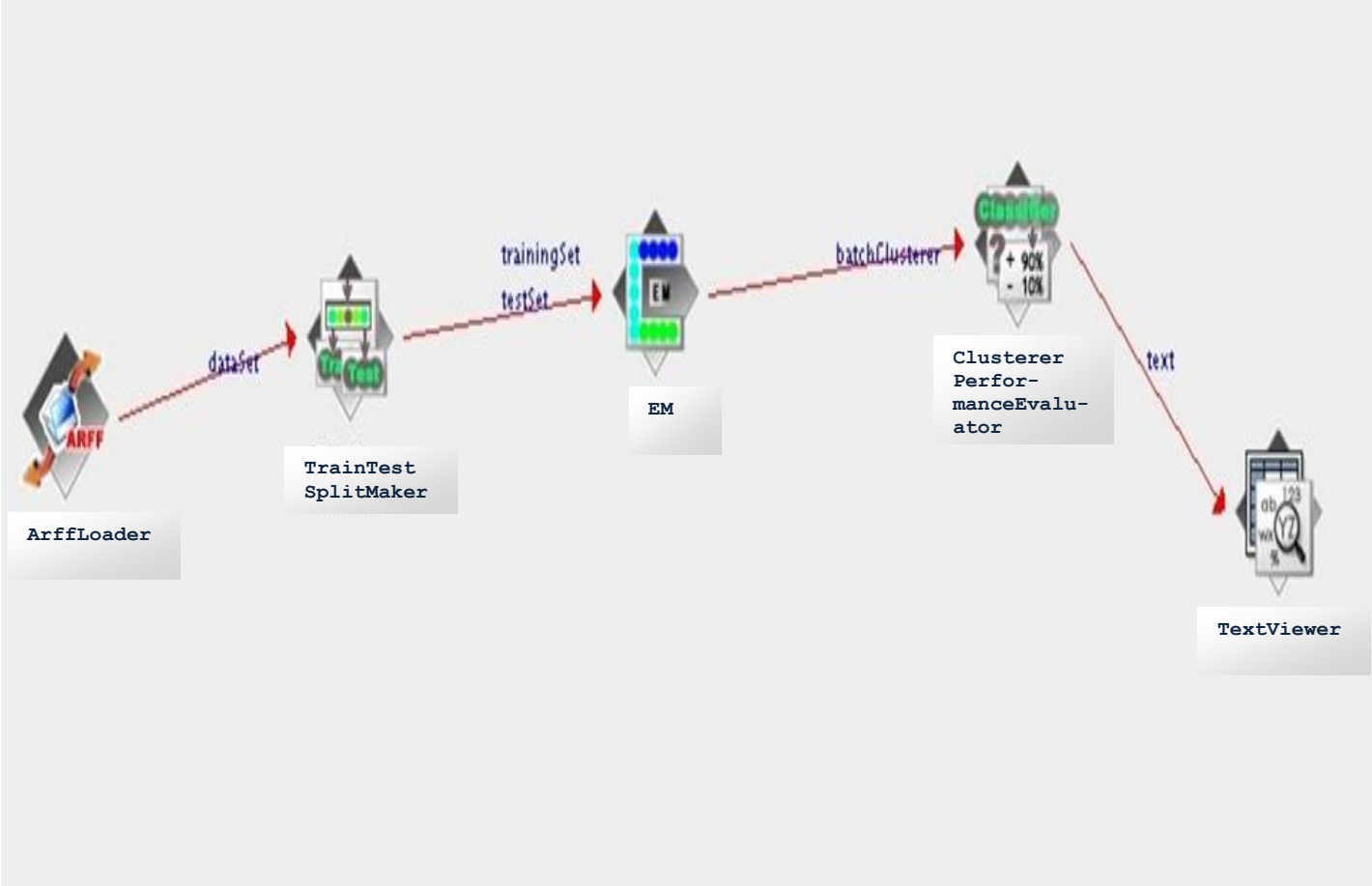


Figure 3.4: Unsupervised Learning knowledge flow setup in Weka.

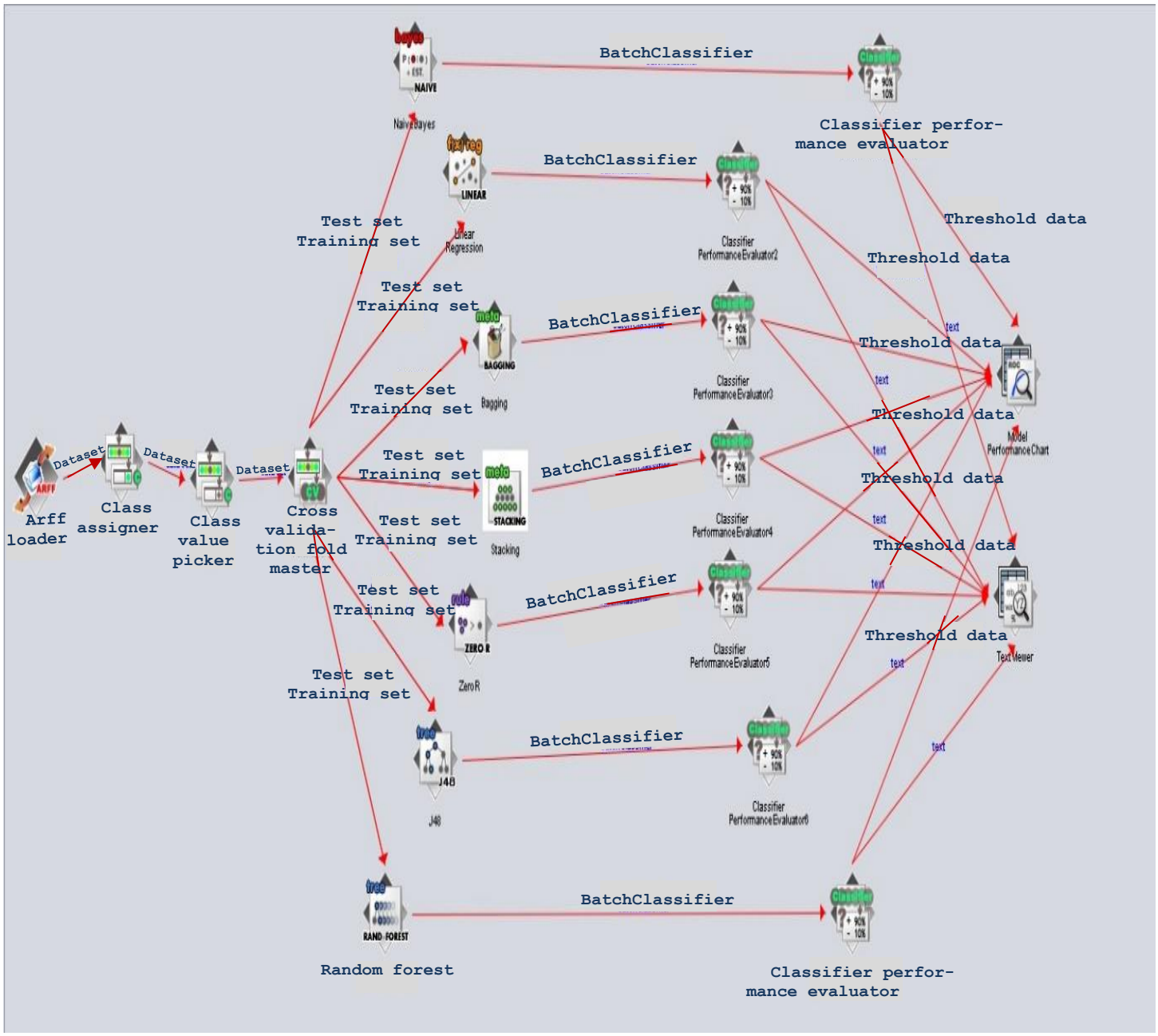


Figure 3.5: Knowledge flow setup for testing several classification algorithms on a given dataset in parallel.

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 3.5 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:

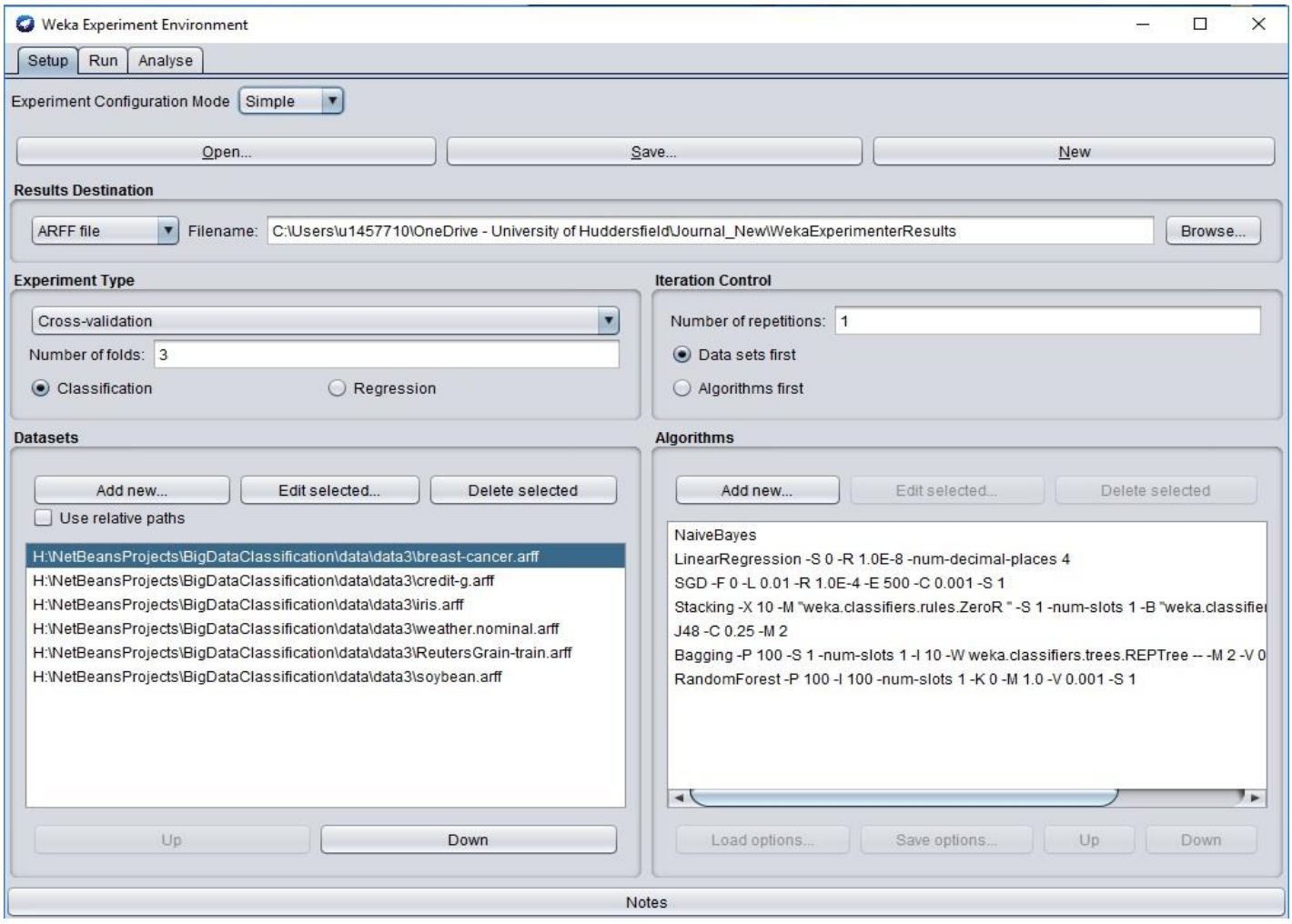
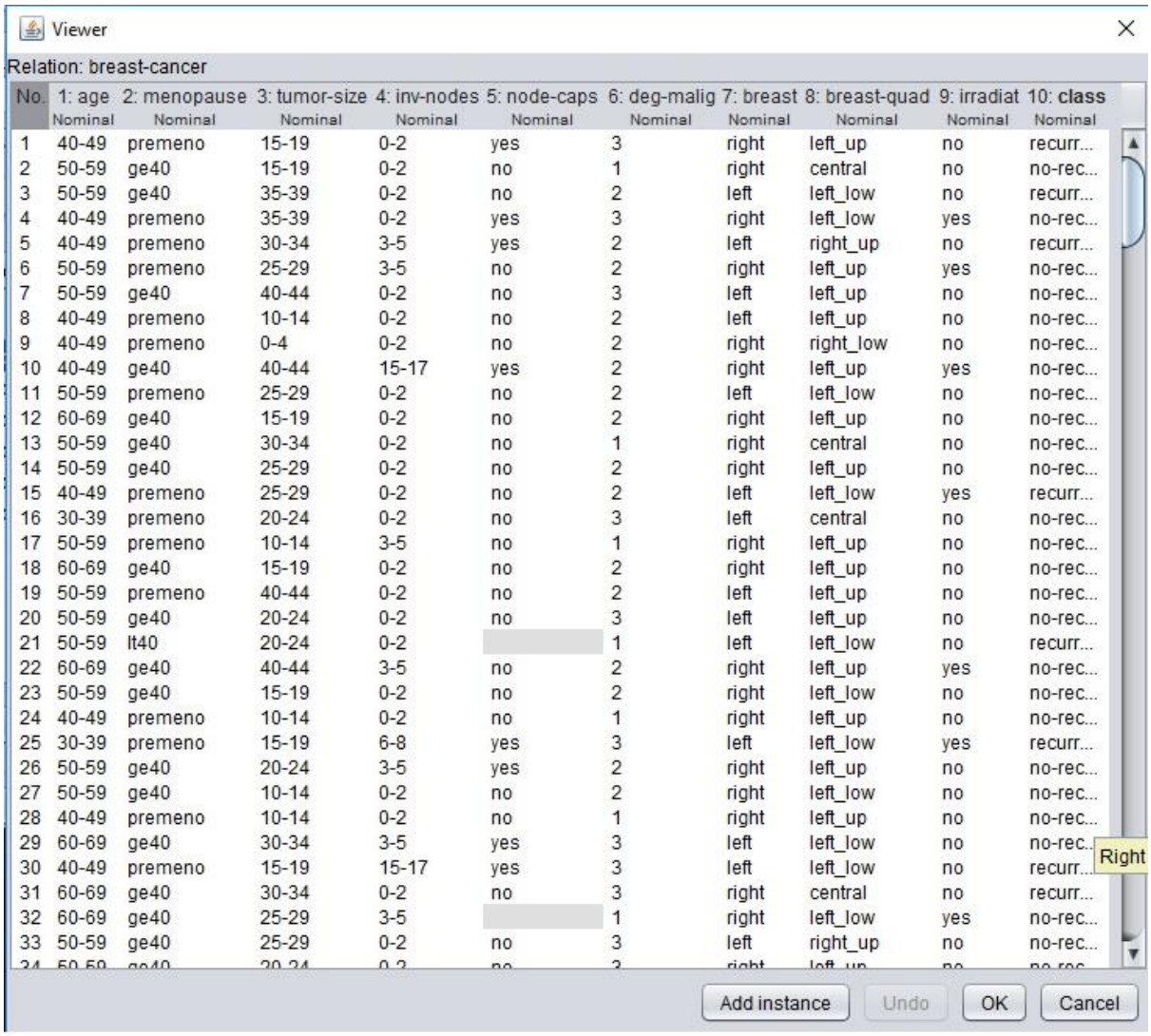


Figure 3.6: Experimenter setup for testing several classification algorithms on various datasets.

In Figure 3.6 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.



No.	1: age	2: menopause	3: tumor-size	4: inv-nodes	5: node-caps	6: deg-malig	7: breast	8: breast-quad	9: irradiat	10: class
	Nominal	Nominal	Nominal	Nominal	Nominal	Nominal	Nominal	Nominal	Nominal	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...

Figure 3.7: Dataset view in tabular format from the Experimenter

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 3.7, 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 3.8 below.

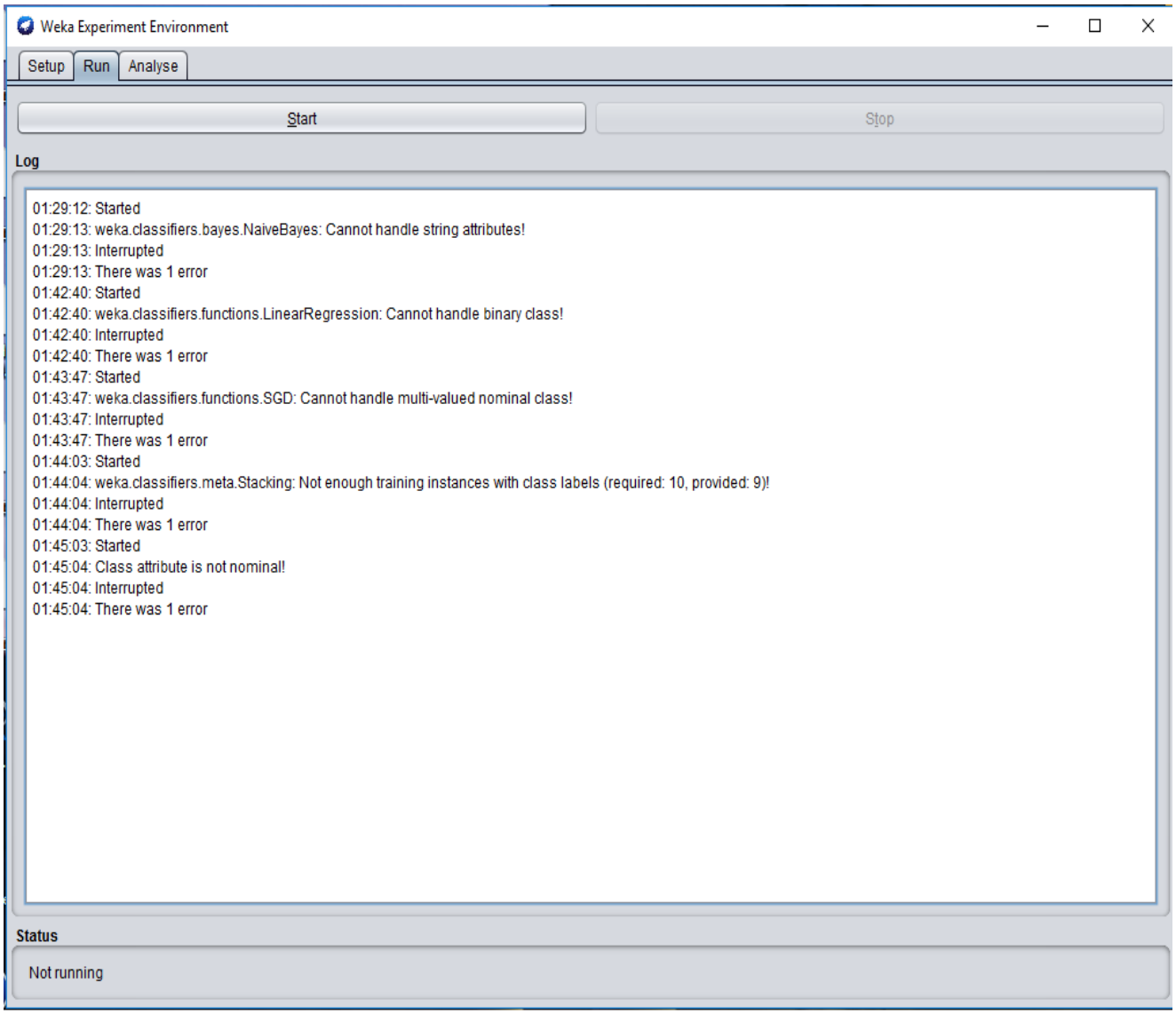


Figure 3.8: Possible errors faced when running the experimenter on several datasets and algorithms

From Figure 3.8 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.

3.2.4Preliminary 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 4*.

Table 3.2: Area Under the Curve (AUC)

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
contact-lenses	0.93	0.50	-	0.85	-	0.88	0.50	0.50	0.82	0.84
cpu	-	-	-	-	-	-	-	-	-	-
cpu.with.vendor	-	-	-	-	-	-	-	-	-	-
credit-g	0.78	0.49	0.66	0.67	-	0.75	0.50	0.50	0.66	0.76
diabetes	0.82	0.50	0.72	0.74	-	0.83	0.50	0.50	0.74	0.82
glass	0.70	0.74	-	0.81	-	0.85	0.49	0.49	0.75	0.92
ionosphere	0.93	0.83	0.82	0.88	-	0.94	0.50	0.50	0.88	0.97
iris.2D	1.00	1.00	-	1.00	-	1.00	0.49	0.49	0.98	1.00
labor	0.98	0.87	0.86	0.96	-	0.88	0.47	0.47	0.73	0.94
reutersCorn-test	-	-	-	-	-	-	0.50	0.50	-	-
reutersCorn-train	-	-	-	-	-	-	0.50	0.50	-	-
segment-challenge	1.00	0.72	-	1.00	-	1.00	0.50	0.50	1.00	1.00
soybean	1.00	1.00	-	1.00	-	0.98	0.50	0.50	0.97	1.00
supermarket	0.50	0.50	0.50	0.50	-	0.50	0.50	0.50	0.50	0.50
unbalanced	0.59	0.50	0.50	0.65	-	0.65	0.50	0.50	0.50	0.70
vote	0.97	0.95	0.95	0.98	-	0.99	0.50	0.50	0.98	0.99
weather.nominal	0.82	0.50	0.63	0.78	-	0.76	-	0.43	0.67	0.84
weather.numeric	0.80	0.50	0.53	0.69	-	0.70	-	0.43	0.63	0.64

Table 3.2 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.

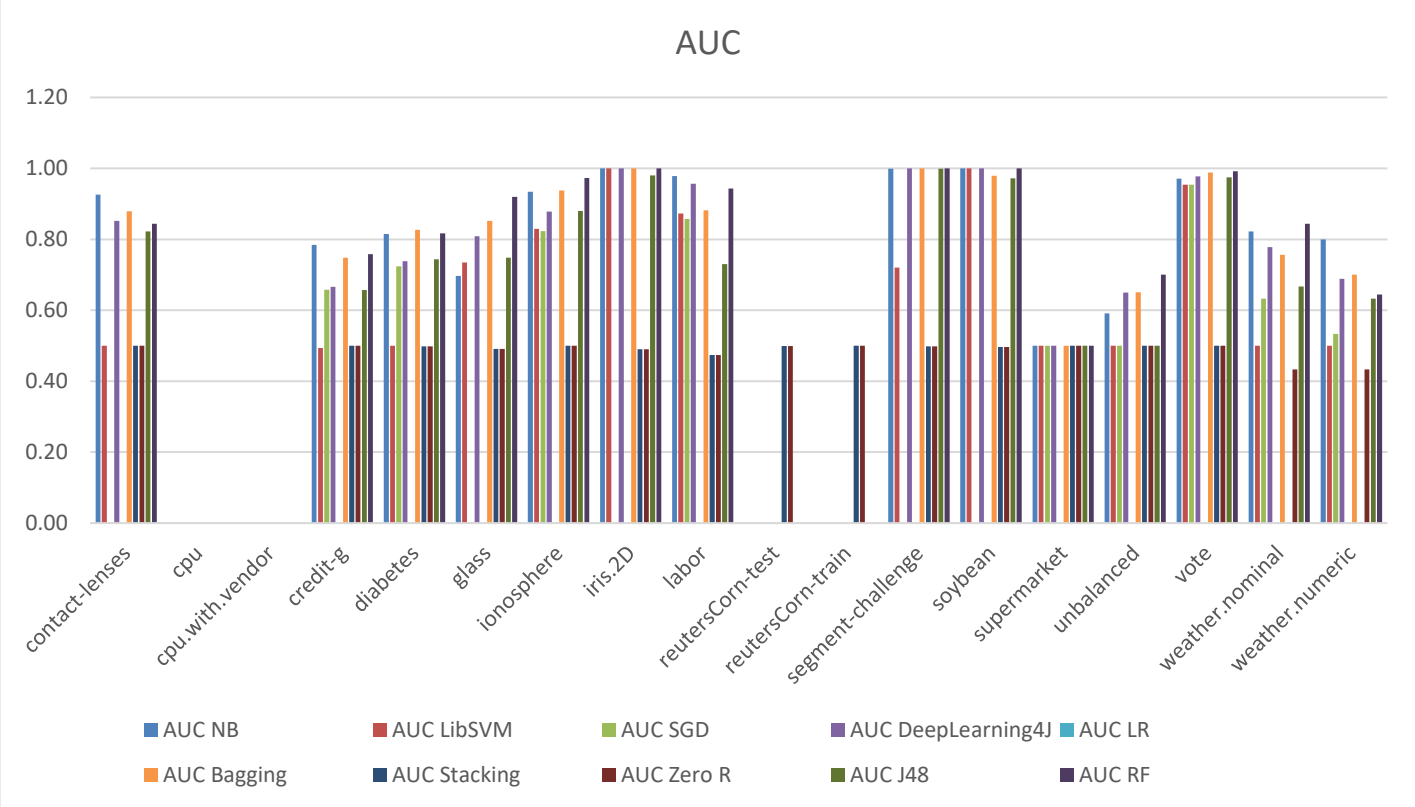


Figure 3.9: Area Under ROC (AUC)

From Figure 3.9 above, it is expected that the AUC for choosing the best performing algorithm given a dataset, should be the figure closest to 1.

Table 3.3: F-Measure for datasets per algorithm.

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
contact-lenses	0.83	0.77	-	0.83	-	0.50	0.77	1.00	0.89	0.83
cpu	-	-	-	-	-	-	-	-	-	-
cpu.with.vendor	-	-	-	-	-	-	-	-	-	-
credit-g	0.83	0.82	0.84	0.77	-	0.81	0.82	0.82	0.80	0.84
diabetes	0.82	0.79	0.83	0.76	-	0.82	0.79	0.79	0.79	0.82
glass	0.23	0.64	-	0.64	-	0.70	0.52	0.52	0.67	0.78
ionosphere	0.86	0.88	0.90	0.81	-	0.93	0.78	0.78	0.93	0.95
iris.2D	1.00	1.00	-	1.00	-	1.00	0.32	0.32	0.98	1.00
labor	0.96	0.84	0.89	0.84	-	0.86	0.79	0.79	0.84	0.90
reutersCorn-test	-	-	-	-	-	-	0.98	0.98	-	-
reutersCorn-train	-	-	-	-	-	-	0.99	0.99	-	-
segment-challenge	0.96	0.61	-	0.98	-	0.99	0.27	0.27	0.99	1.00
soybean	1.00	1.00	-	1.00	-	0.81	0.24	0.24	0.90	1.00
supermarket	0.78	0.53	0.78	0.35	-	0.78	0.78	0.78	0.78	0.78
unbalanced	0.96	0.99	0.99	0.99	-	0.99	0.99	0.99	0.99	0.99
vote	0.92	0.96	0.96	0.96	-	0.96	0.76	0.76	0.97	0.98
weather.nominal	0.84	0.78	0.71	0.75	-	0.82	-	0.78	0.67	0.78
weather.numeric	0.82	0.78	0.67	0.78	-	0.78	-	0.78	0.76	0.78

Table 3.3 above, gives us the F-measure estimated for each dataset per algorithm.

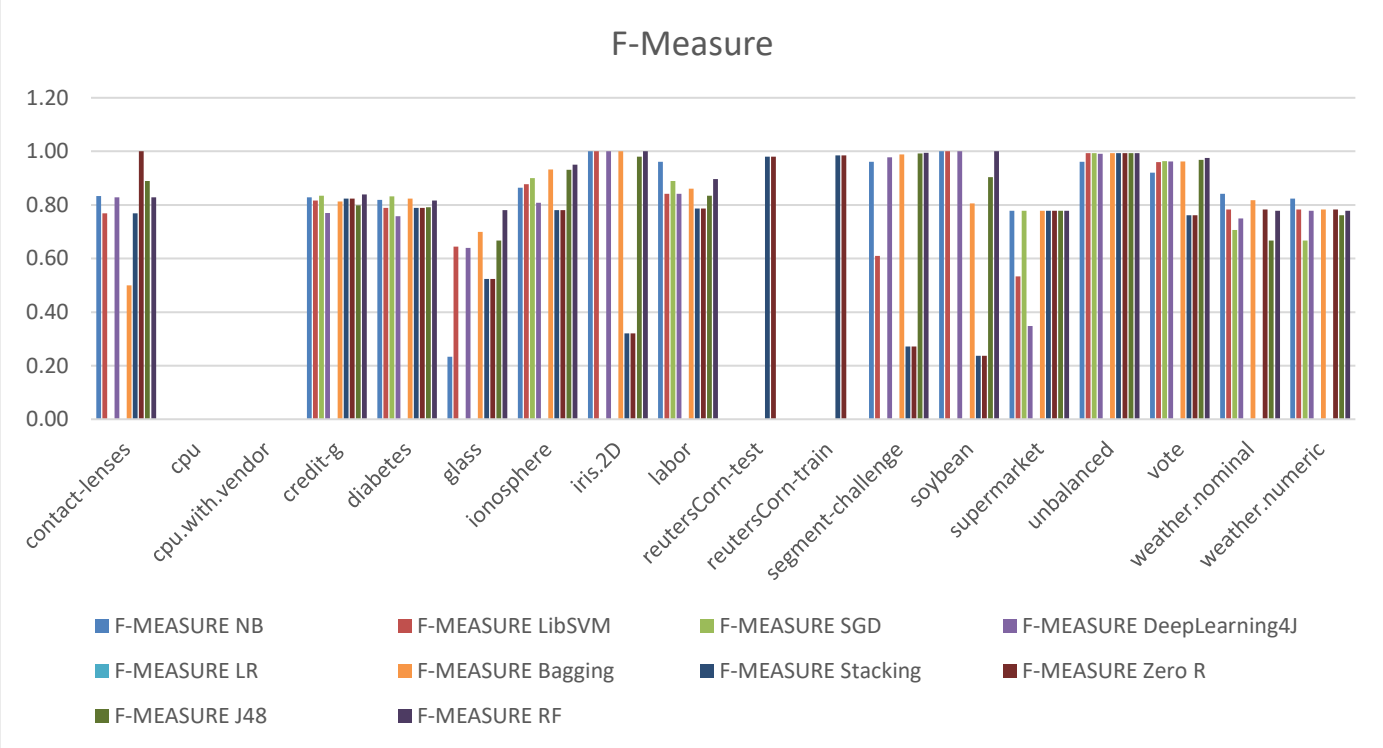


Figure 3.10: F-Measure for each dataset against several classification algorithms

From Figure 3.10 above, it is expected that an F-Measure score closer to 1 is more desirable for any given dataset.

Table 3.4: Table of the Mean Absolute Error (MAE) for the various datasets.

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
contact-lenses	0.24	0.25	-	0.18	-	0.31	0.37	0.37	0.15	0.22
cpu	-	-	-	206.27	43.79	35.73	96.35	96.36	-	26.97
cpu.with.vendor	-	-	-	196.48	35.40	29.67	87.64	87.64	-	14.74
credit-g	0.30	0.31	0.25	0.33	-	0.34	0.42	0.42	0.34	0.35
diabetes	0.28	0.35	0.23	0.33	-	0.31	0.45	0.45	0.31	0.31
glass	0.16	0.10	-	0.13	-	0.12	0.21	0.21	0.10	0.10
ionosphere	0.17	0.08	0.14	0.13	-	0.14	0.46	0.46	0.10	0.14
iris.2D	0.03	0.03	-	0.05	-	0.05	0.45	0.49	0.06	0.03
labor	0.08	0.11	0.14	0.12	-	0.28	0.46	0.46	0.31	0.23
reutersCorn-test	-	-	-	-	-	-	0.08	0.08	-	-
reutersCorn-train	-	-	-	-	-	-	0.06	0.06	-	-
segment-challenge	0.06	0.15	-	0.03	-	0.03	0.24	0.24	0.02	0.02
soybean	0.01	0.01	-	0.01	-	0.03	0.10	0.10	0.02	0.03
supermarket	0.46	0.64	0.36	0.48	-	0.46	0.46	0.46	0.46	0.46
unbalanced	0.09	0.01	0.01	0.02	-	0.03	0.03	0.03	0.03	0.03
vote	0.10	0.05	0.04	0.05	-	0.07	0.47	0.47	0.06	0.07
weather.nominal	0.37	0.36	0.36	0.32	-	0.40	-	0.47	0.37	0.33
weather.numeric	0.38	0.36	0.43	0.32	-	0.42	-	0.47	0.40	0.40

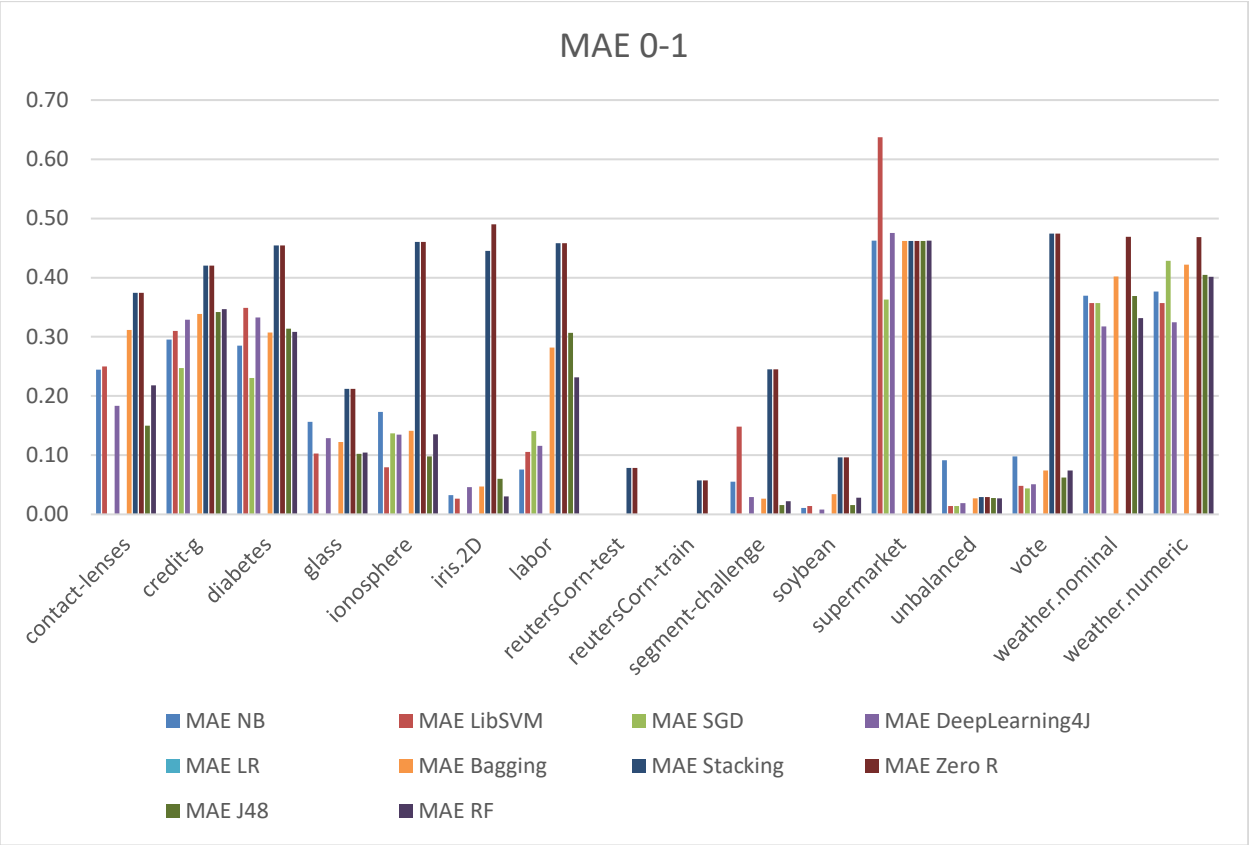


Figure 3.11: Mean Absolute Error (MAE) 0-1

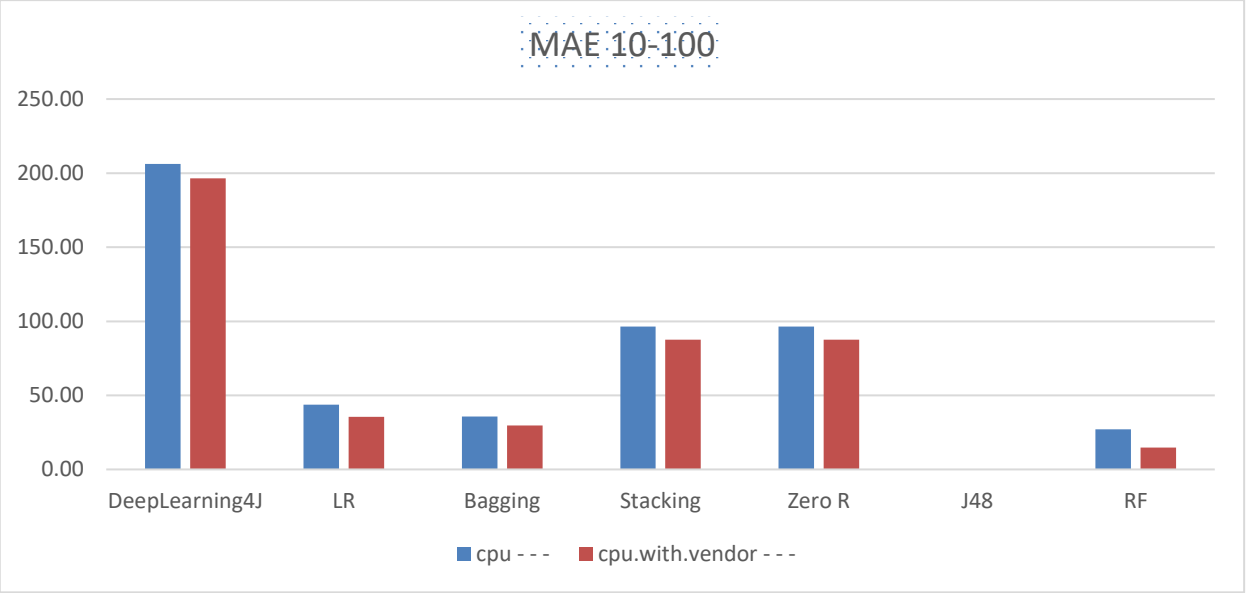


Figure 3.12: Mean Absolute Error (MAE) for Cpu and Cpu.with.vendor Datasets

Figure 3.11 and Figure 3.12 are graphs derived from using the figures in the above Table 3.4. Figure 3.12 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.

Table 3.5: Accuracy in % and Correlation Coefficients for Cpu and CPu.With.Vendor datasets

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
contact-lenses	83.33	62.50	-	75.00	-	70.83	62.50	62.50	87.50	79.17
cpu	-	-	-	-0.11	0.89	0.88	-0.11	-0.11	-	0.95
cpu.with.vendor	-	-	-	-0.11	0.93	0.90	-0.11	-0.11	-	0.97
credit-g	75.00	69.00	75.30	67.40	-	72.10	70.00	70.00	70.50	75.50
diabetes	75.91	65.10	76.95	68.36	-	76.17	65.10	65.10	73.31	75.39
glass	44.39	64.02	-	57.01	-	71.50	35.51	35.51	67.76	78.04
ionosphere	83.48	92.02	86.33	86.61	-	91.17	64.10	64.10	90.88	93.45
iris.2D	96.00	96.00	-	94.67	-	96.00	32.00	32.00	93.33	97.33
labor	94.74	89.47	85.96	89.47	-	80.70	64.91	64.91	77.19	85.96
reutersCorn-test	-	-	-	-	-	-	96.03	96.03	-	-
reutersCorn-train	-	-	-	-	-	-	97.10	97.10	-	-
segment-challenge	81.00	48.13	-	90.20	-	95.87	15.73	15.73	95.13	97.33
soybean	92.39	86.82	-	92.39	-	81.70	13.47	13.47	89.75	91.95
supermarket	63.71	36.29	63.71	54.59	-	63.71	63.71	63.71	63.71	63.71
unbalanced	92.52	98.60	98.60	98.13	-	98.60	98.60	98.60	98.60	98.60
vote	90.57	95.17	95.63	95.40	-	95.40	61.38	61.38	96.09	97.01
weather.nominal	78.57	64.29	64.29	71.43	-	71.43	-	64.29	64.29	71.43
weather.numeric	78.57	64.29	57.14	71.43	-	64.29	-	64.29	64.29	71.43

In Table 3.5 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 3.5 displays both. There was however a need to plot this separately on two different plots because of the scale differences.

The graphs plotted in Figure 3. and Figure 3.13 below is used to represent the data from Table 3.5.

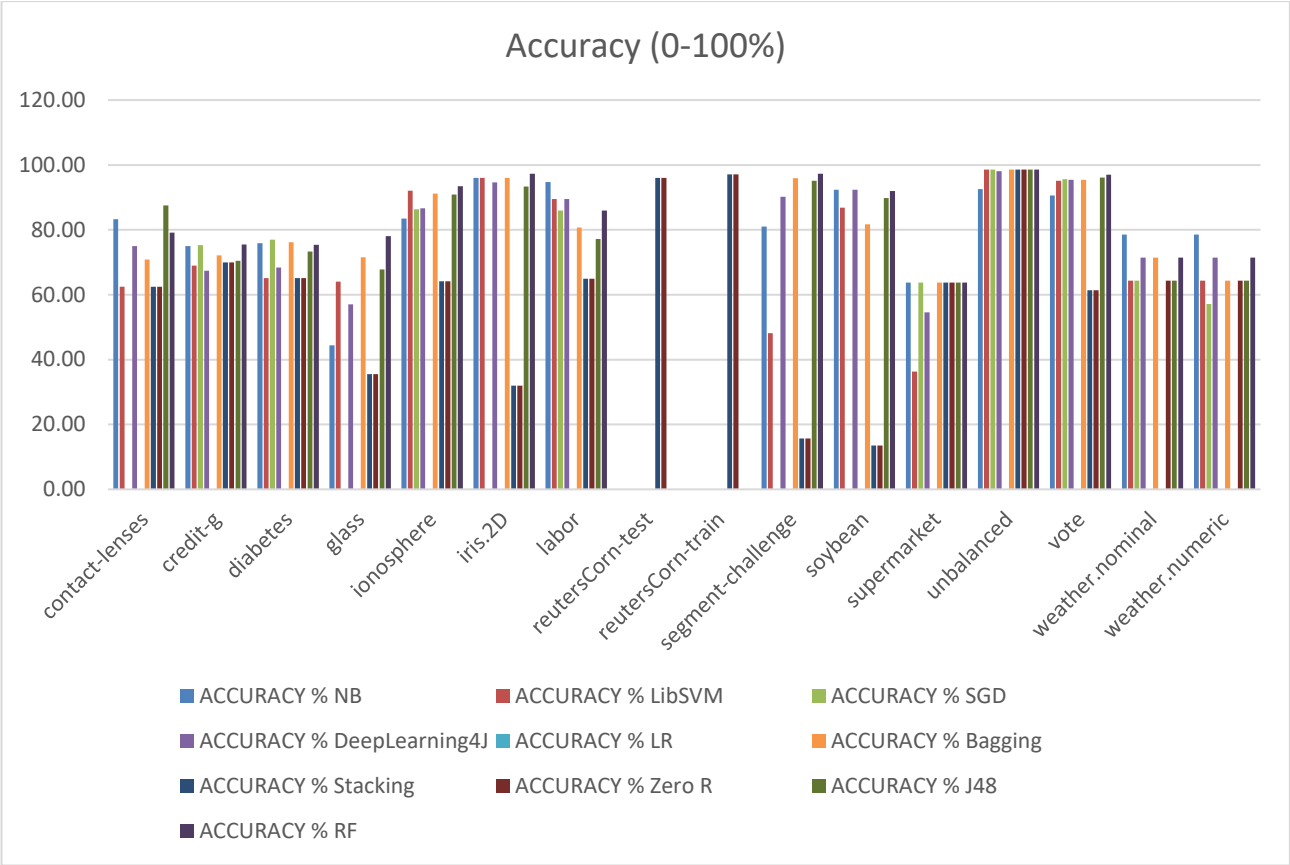


Figure 3.13: Shows the accuracy in percentage (0-100%) of various classification models on a variety of datasets.

From Figure 3.13 above, it is expected that the Accuracy closest to 100% is more desirable for any given dataset.

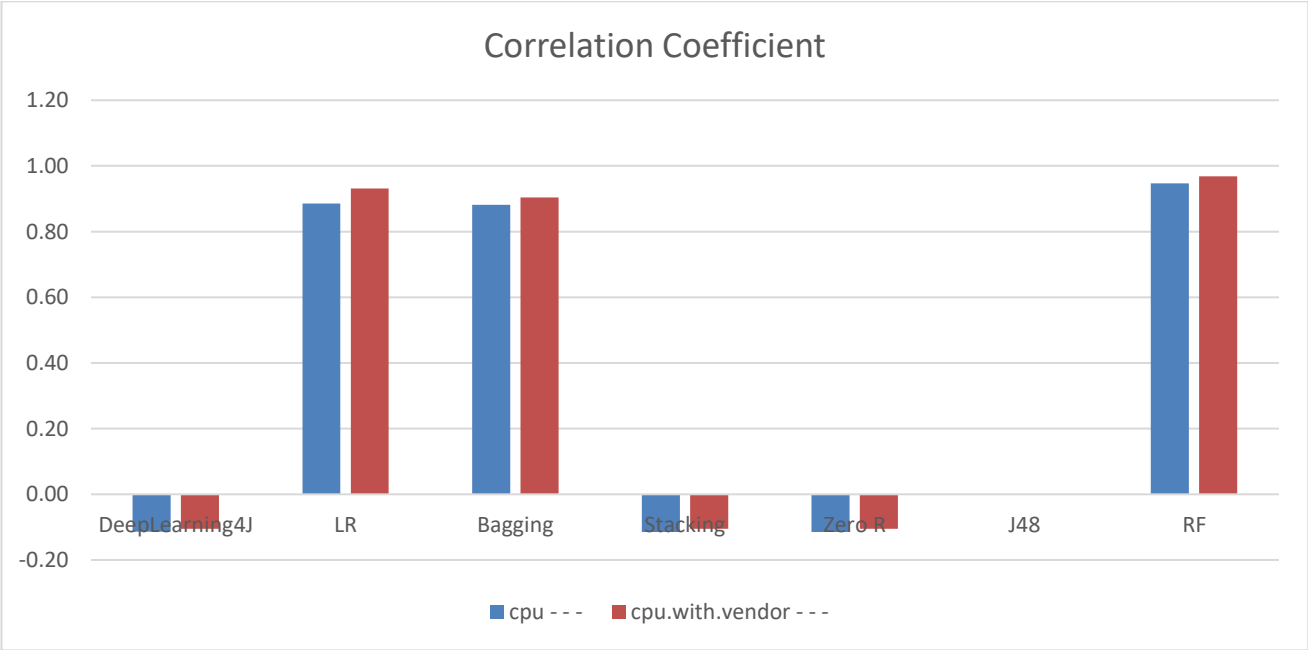


Figure 3.13: Correlation Coefficient 0-1 (Cpu and Cpu.With.Vendor Datasets)

From Figure 3.13 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 3.6: 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 3.6, 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.

3.2.5Size 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 3.7: 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

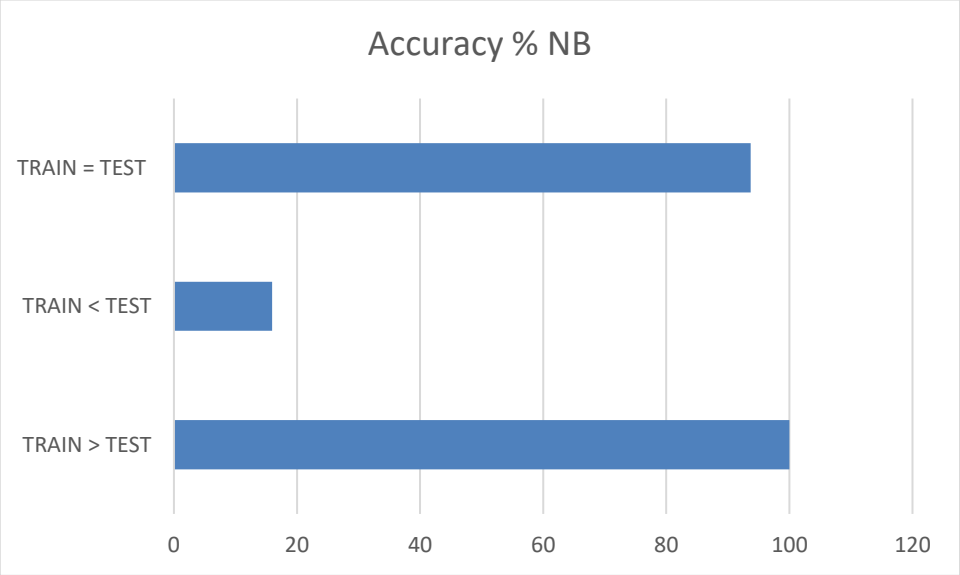


Figure 3.14: Size Effect on Accuracy (%)

From Figure 3.14 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.

3.3 Machine Learning Algorithms Considered

Table 3.8: 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 GreedyStep-Wise & 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

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 thesis (Section 2.2.3) is taken into consideration.

3.3.1Feature 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).

3.3.2Supervised 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, Hassan, Arshaq, Khan, & Shamail, 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 (J. Ross Quinlan, 1986; J Ross 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 (X. 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 (Frank, 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 $p(C|x)$ for each of the possible classes a test instance may belong to (Charu 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 $p(x|C)$ 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 $p(C|x)$.
- 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.

3.3.3Unsupervised 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, Bajpai, & Litoriya, 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, Nigam, & Ungar, 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, Langley, & Fisher, 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).

3.3.4Evaluation 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 (Charu C. 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), r**ecall** from equation (2.3), p**recision** from equation (2.4), s**pecificity** from equation (2.5), f**all-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.

3.4 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 thesis 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.

3.5 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.

Summary

This chapter describes and discusses a combination of research methodologies e.g. experimental, theoretical and systems design used in this thesis. 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.

Chapter 4

4 Hybrid-AutoML System

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 thesis 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.

4.1 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.

4.2 The Model Design

4.2.1 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.

4.2.2 Model Architecture

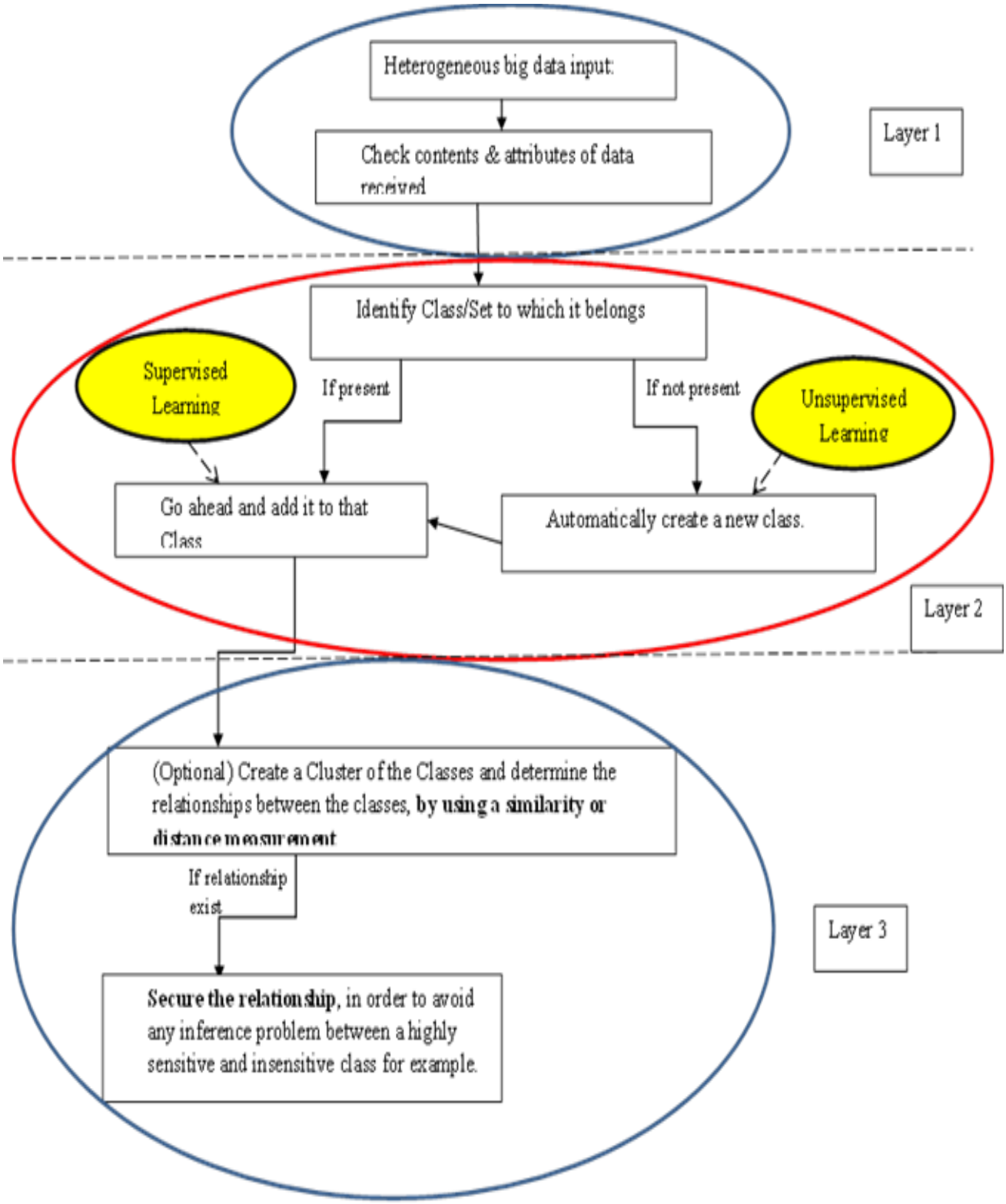


Figure 4.1: Three Layered Decision architecture for the hybrid auto machine learning system proposed after experiments.

4.2.3 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).

4.2.4 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 4.1 below).

Table 4.1: 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

From Table 4.1, 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.

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 → 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.

a. IF training labeled set exists then check the size of the labeled set.
b. IF size of the training set > than the test set, THEN invoke a supervised learning method.
c. IF no training set exists, THEN use an unsupervised algorithm.
d. IF the size of the training set < or = test set, use an unsupervised algorithm.
e. IF no labeled instances exist, use an unsupervised algorithm.
f. 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 measurement. 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.

$$\tilde{I} += V_i \text{ where } i < n_a \quad (4.1)$$

- 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 Ĩ 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:

$$D = 100(F/n)\% \qquad (4.2)$$

    - 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.

4.4.3Program

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 thesis?

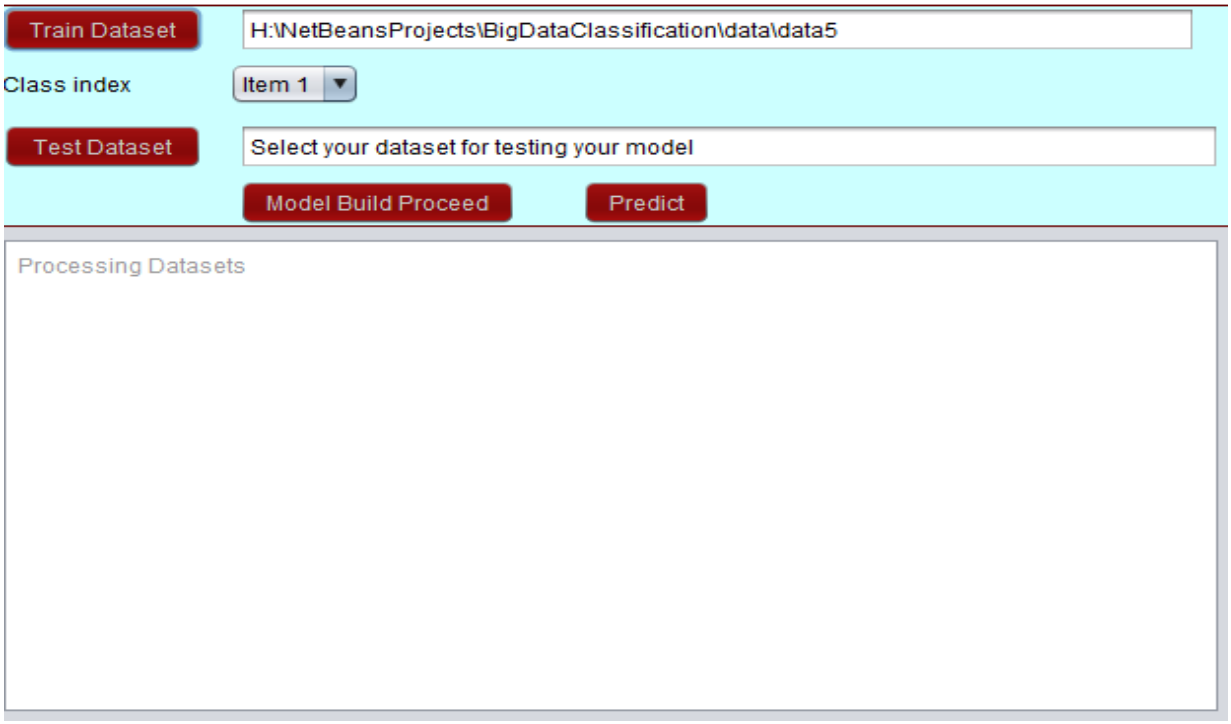


Figure 4.2: Simple GUI interface for the Implementation of the Hybrid Auto Classification System.

Figure 4.2 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. The results from this stage is discussed in chapter 5. This case study is to show 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.

4.5.2Case Study 2

This is the stage of implementing an alternative 'autoProb' function designed in this thesis. 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 5*).

```
% 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
```

Figure 4.3: Details of the contact-lenses-test dataset used.

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 thesis 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.

Chapter 5

5 Results and Discussion

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 thesis. We map each use case to our aims and contributions as outlined in section 1.3 of this research thesis. 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 thesis has been met fully. Also, the performance comparison shows that the hybrid autoML system performs 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.

5.1 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.

5.1.1 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 thesis to aid the automatic selection of an unsupervised ML algorithm e.g. the ‘AutoProbClass’ when supplied with an unlabelled dataset has been achieved. Figure 5.1 shows the data summary for the ‘soy-test’ dataset used in this first user scenario.

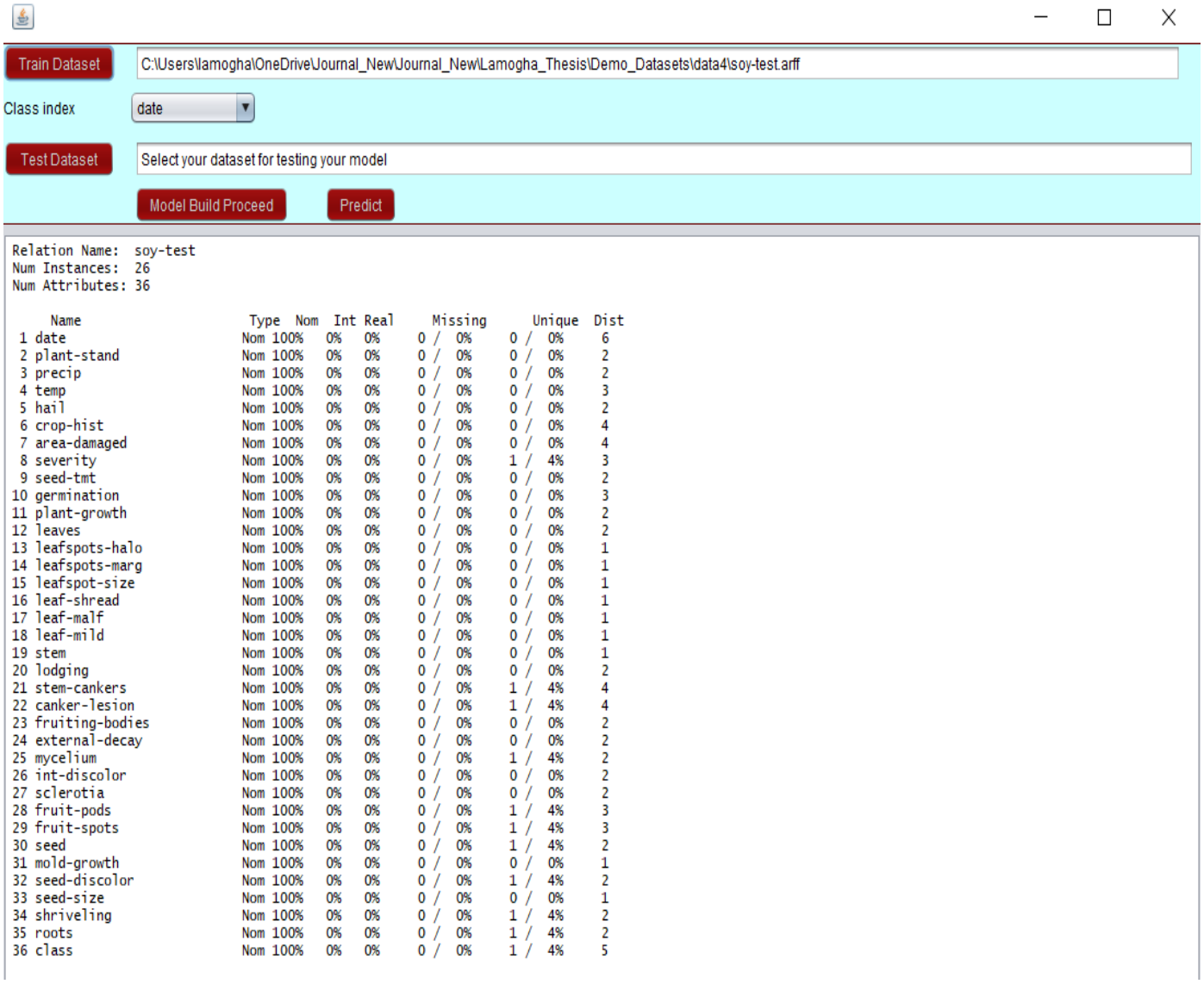


Figure 5.1: Shows a data summary on upload of the small unlabelled dataset (soy-test).

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.

```
Instance identifier for the new instance is: 60021021001102200011030002104000001
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: 40010221011102200010030002104000001
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: 30020221021102200010030002104000001
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: 50021221021102200010030002104000001
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: 11200212021002200010110110034000002
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]
The closest is 0.0% at index point 0
==========

Instance identifier for the new instance is: 11200112011002200010110100034000002
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, 91.666664]
The closest is 91.666664% at index point 22
==========

Instance identifier for the new instance is: 40210000110002200010030000000010100010
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: 51210331100002200010320000012000000011
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 5.2: 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.

From Figure 5.2 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 thesis 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.

5.1.2 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 thesis 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.

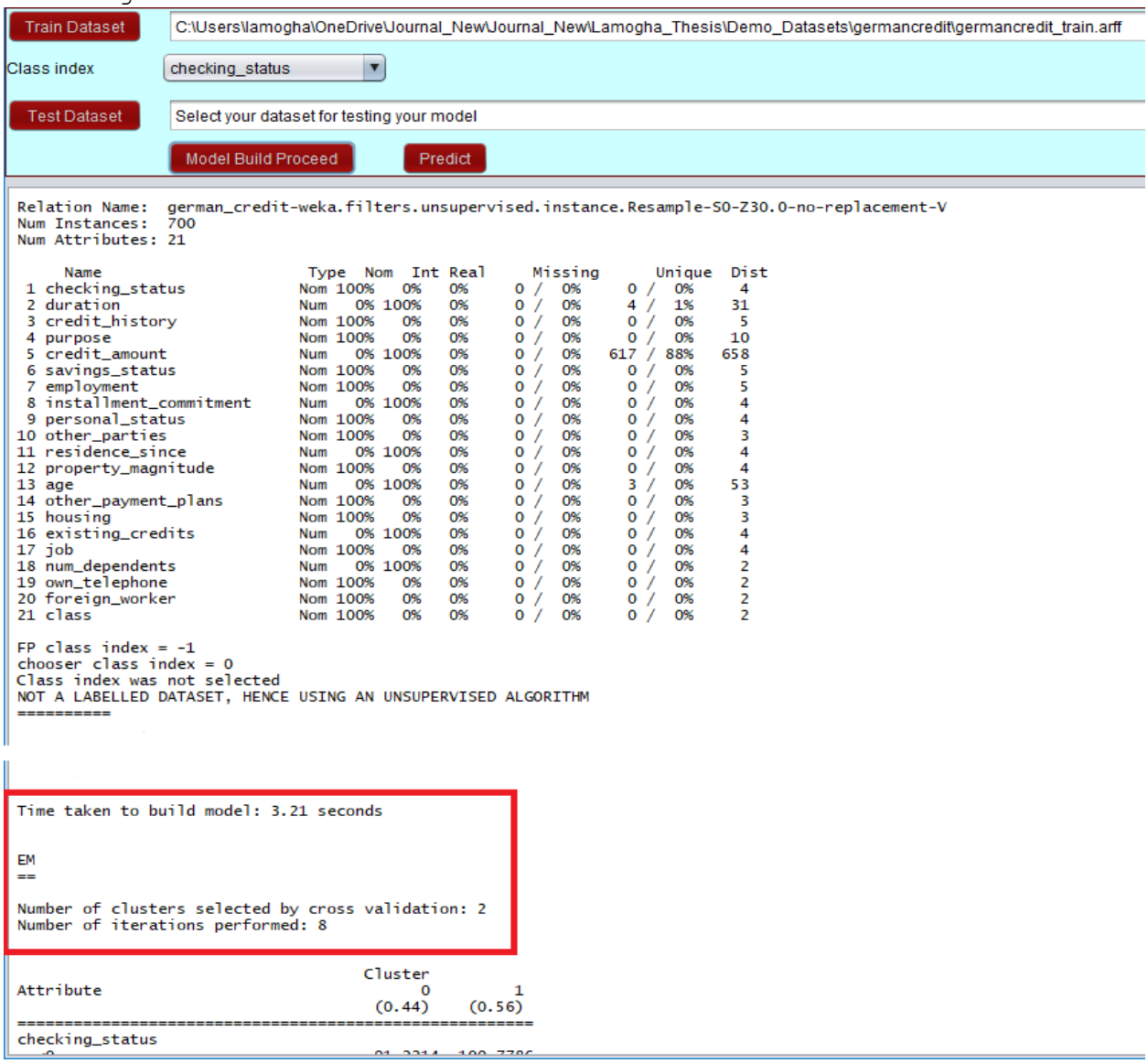


Figure 5.3: shows an unsupervised ML mode using the EM clustering algorithm was automatically chosen as the best to use for this given task. Two clusters where derived and the EM model built in 3.21 seconds.

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.

5.1.3 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.

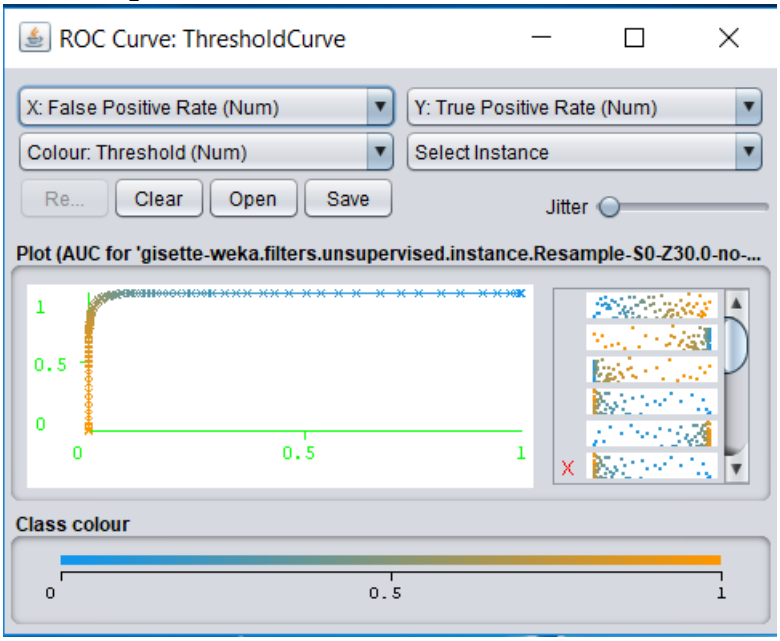


Figure 5.4: The ROC curve obtained after a model was built and tested using the gisette data set.

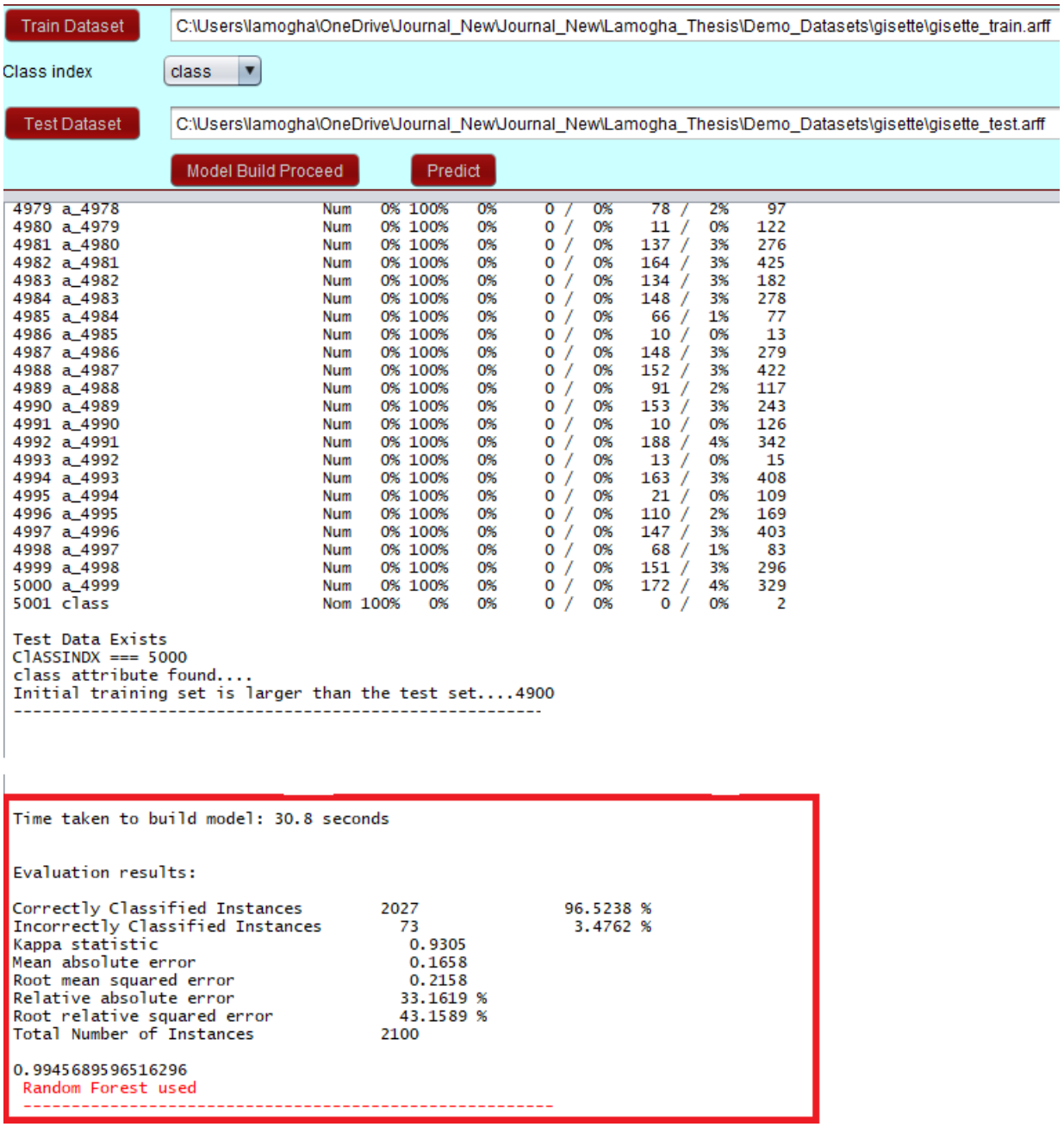


Figure 5.5: Shows the evaluation result obtained from using the hybrid autoML system on the 'gisette' data set.

From Figure 5.5 above, we can see that the hybrid system by following the rule base algorithm designed in this research thesis, automatically uses the random forest to build a model for our given dataset. It is also observed from Figure 5.5 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 5.4). 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 5.5 above, we

can see that the classifier automatically used by our hybrid system performed highly in this use case.

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.

5.1.4 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 thesis 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.

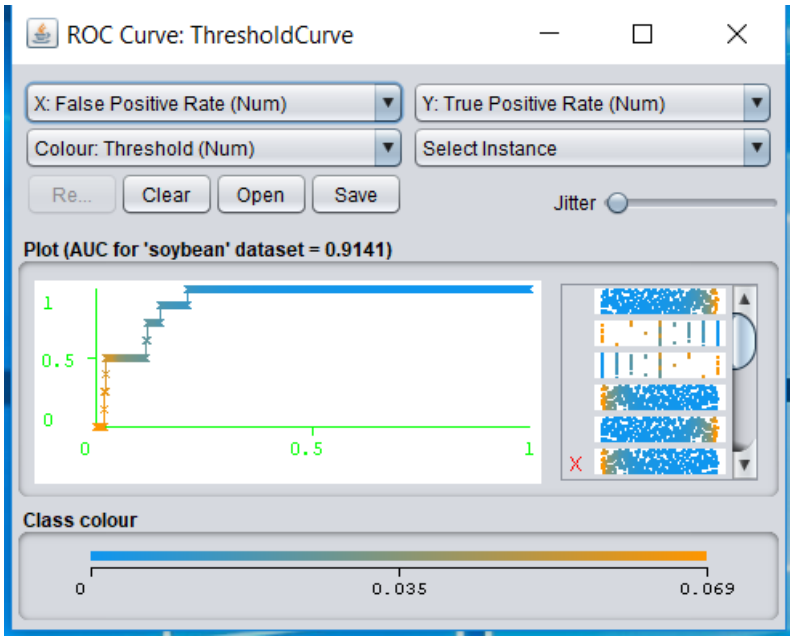


Figure 5.6: ROC curve obtained from training the model on the given train data set.

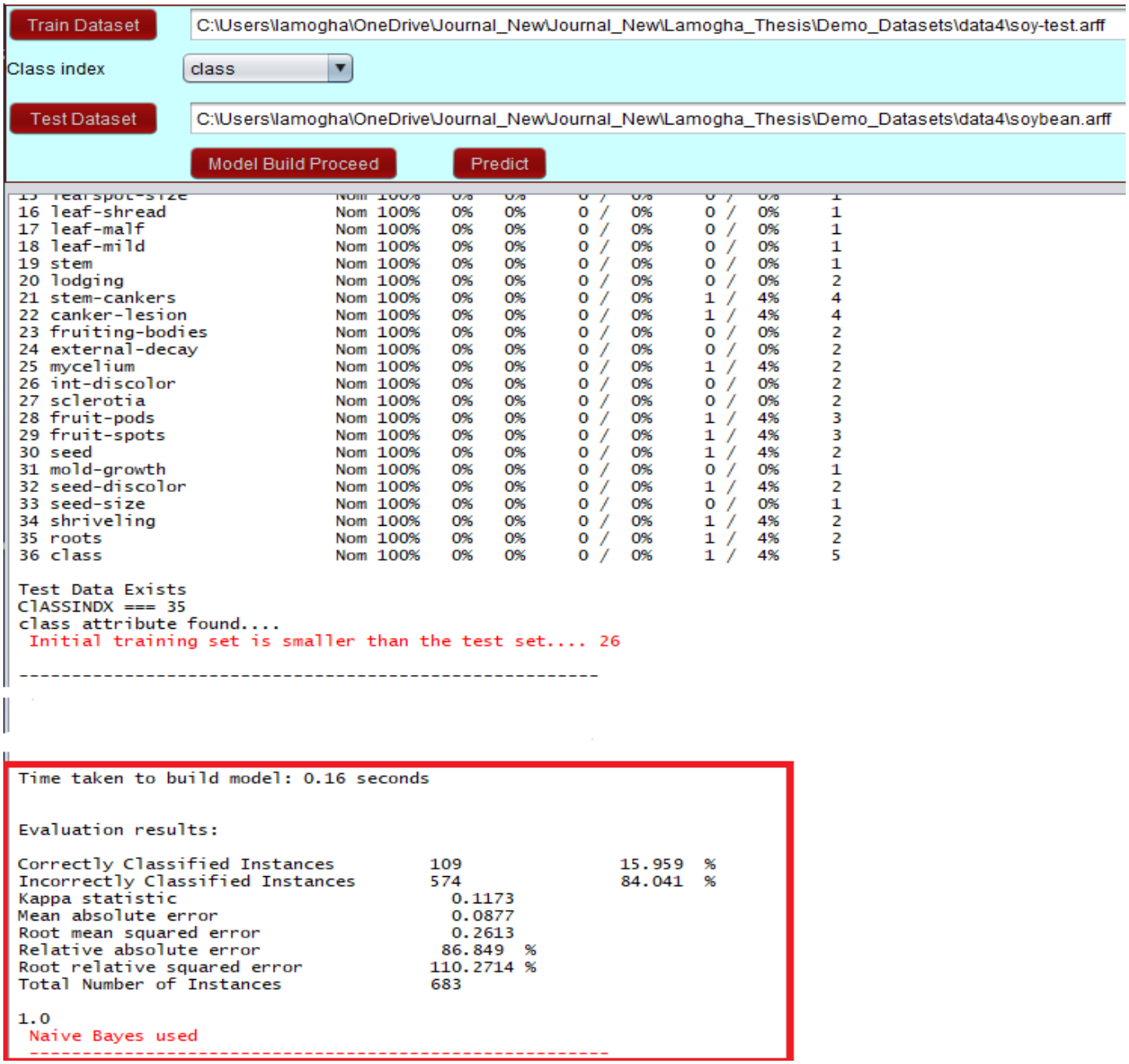


Figure 5.7: Evaluation metrics obtained from using a small trained data set and large test set in useCase4.

From the following ROC curve above in Figure 5.6, 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 5.7 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.

5.1.5 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 thesis 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.

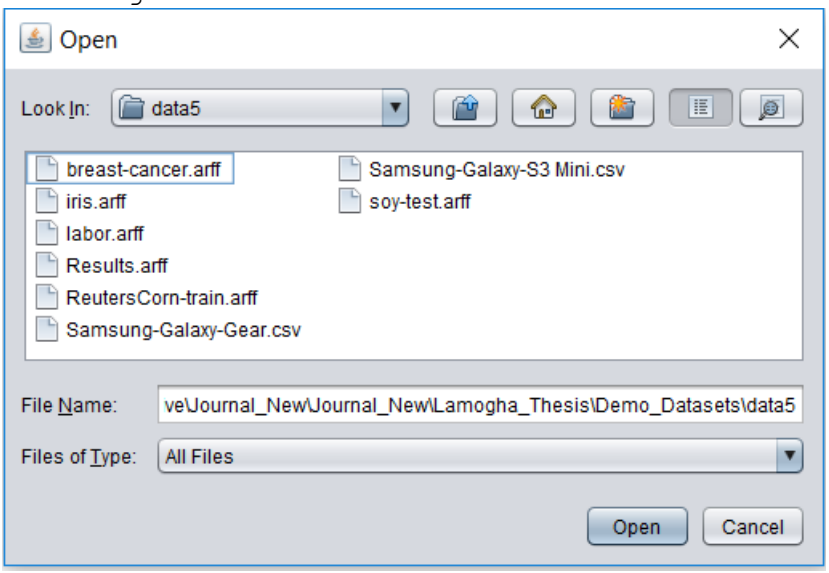


Figure 5.8: A file directory supplied as the location containing the varying data sets to be supplied in one run. It shows a total of 8 datasets that we use to test this user scenario.

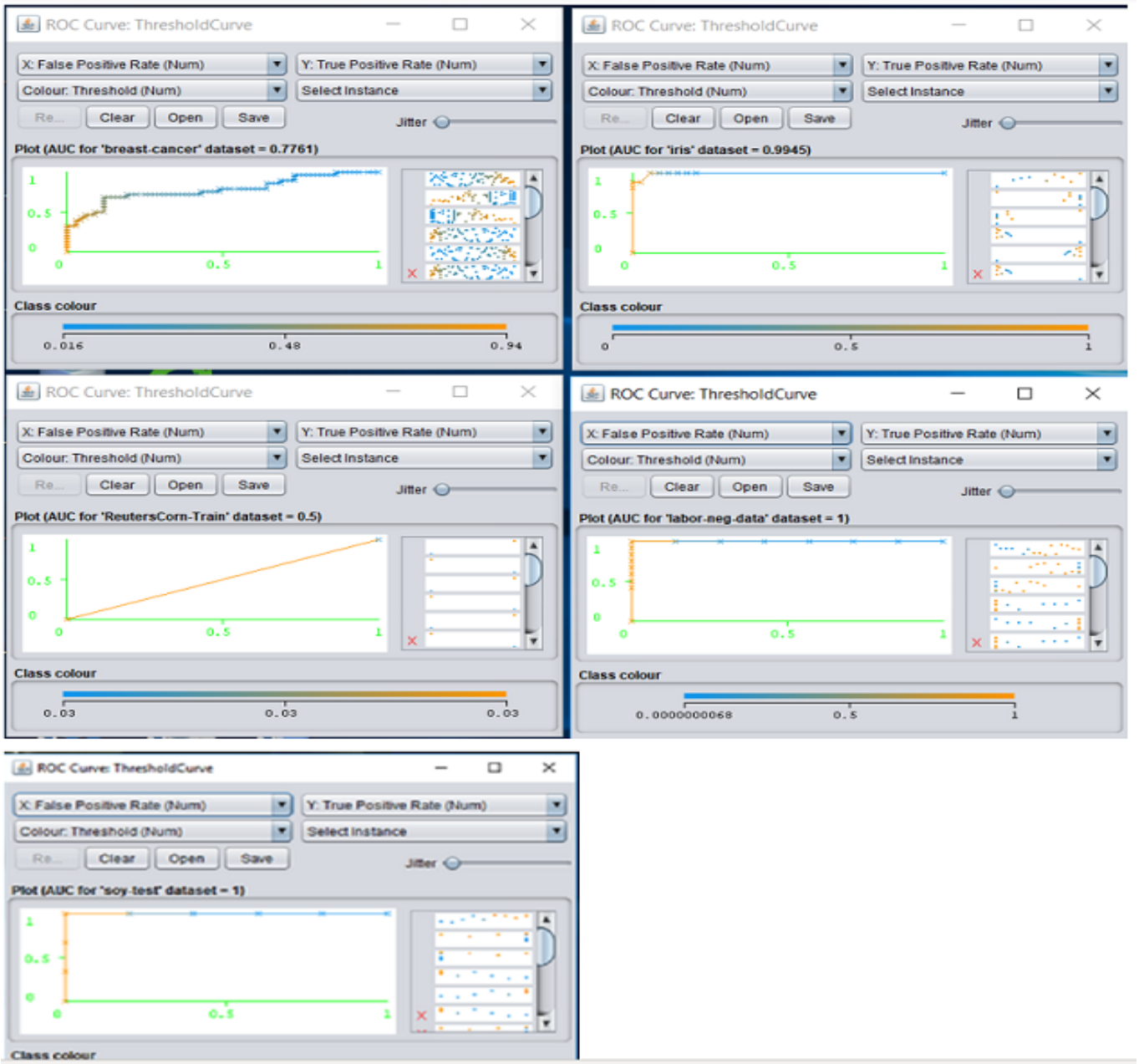


Figure 5.9: ROC curved obtained for five out of the 8 multi-varying data sets in our data location.

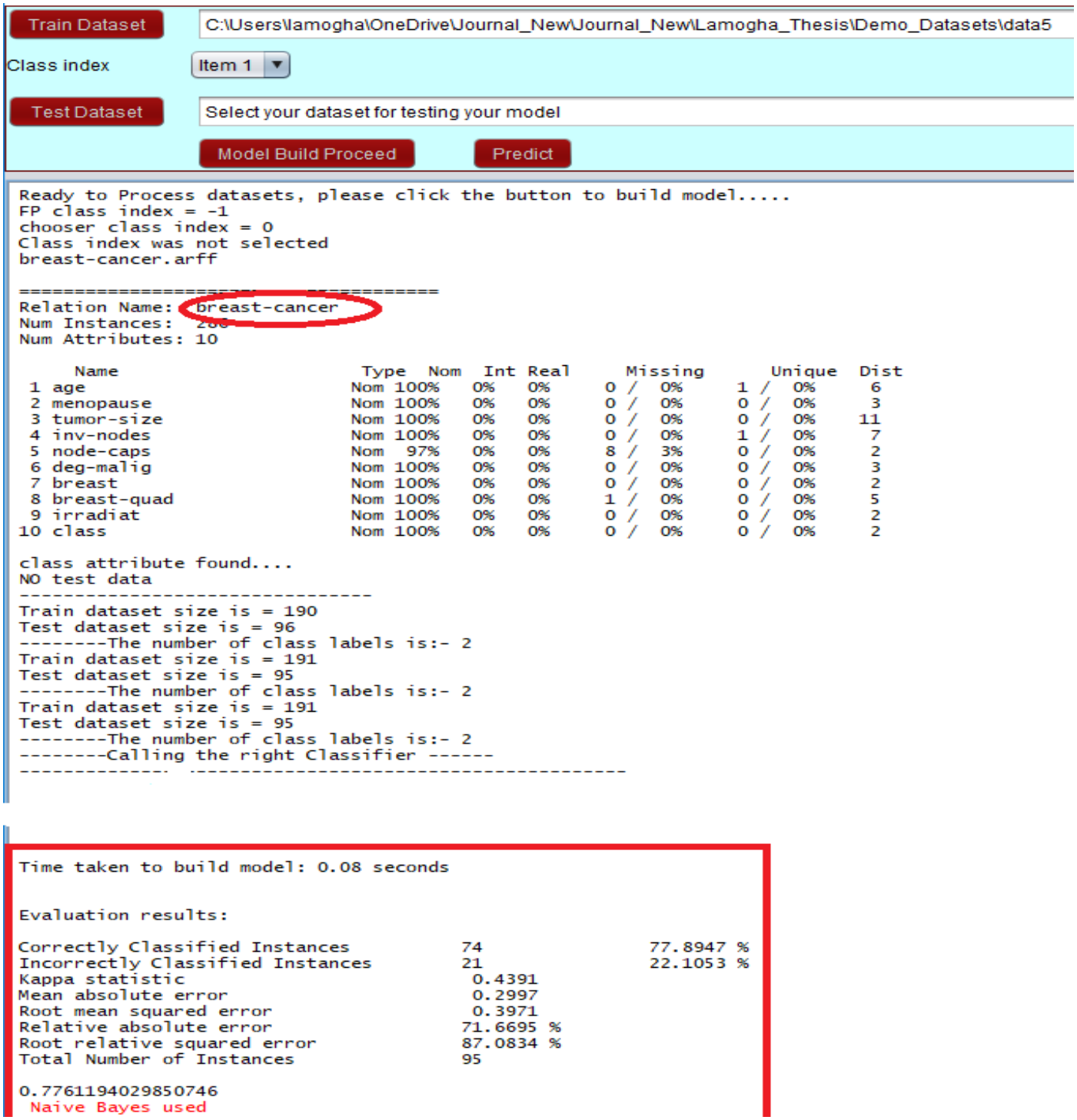


Figure 5.10: Shows the evaluation for the 'breast cancer' data set and Naive Bayes automatically chosen for it as the classifier.

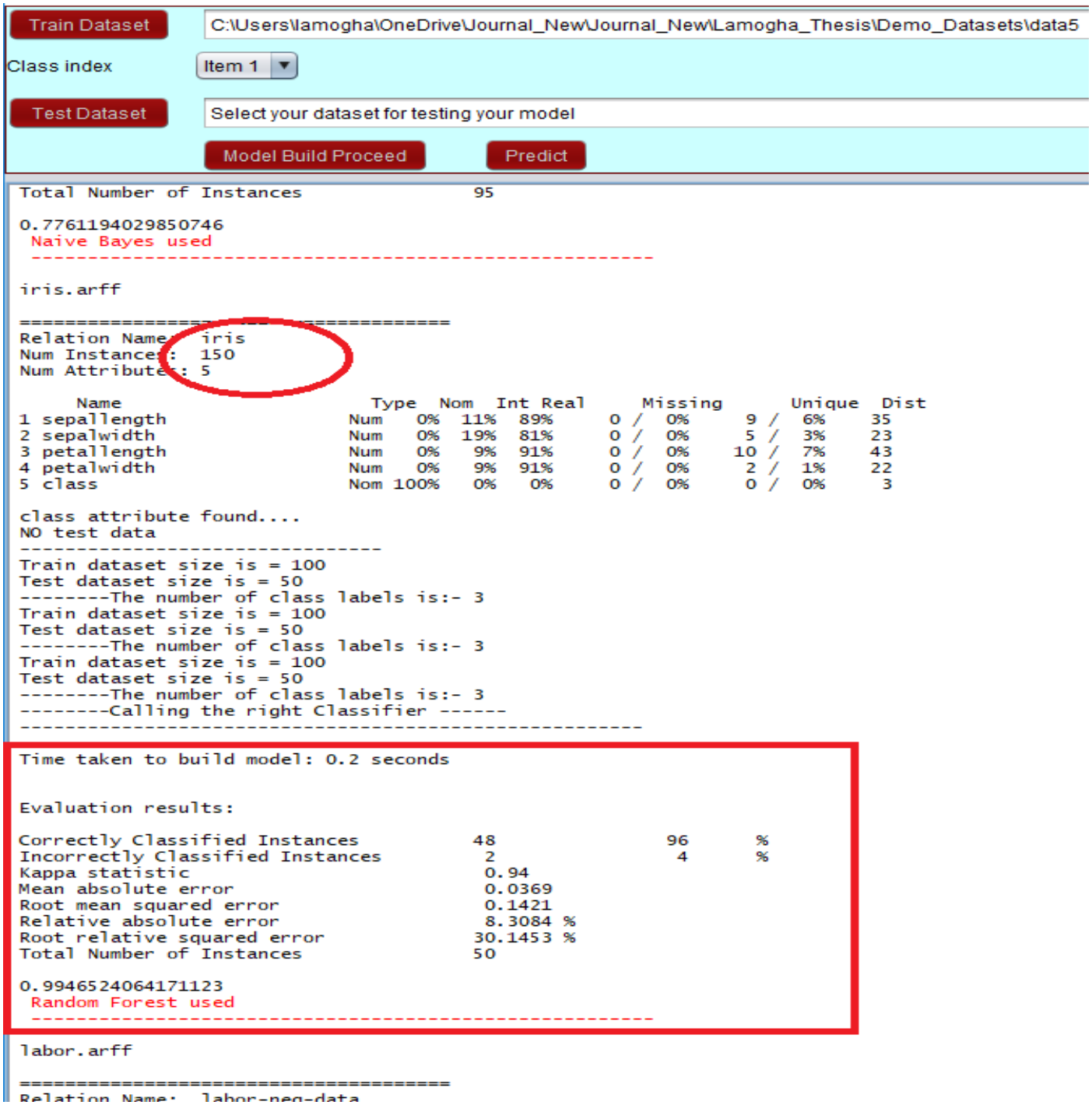


Figure 5.11: Shows that Random forest was chosen for the 'iris' dataset.

Figure 5.11 above, also shows the evaluation results for the chosen random forest model on this data set.

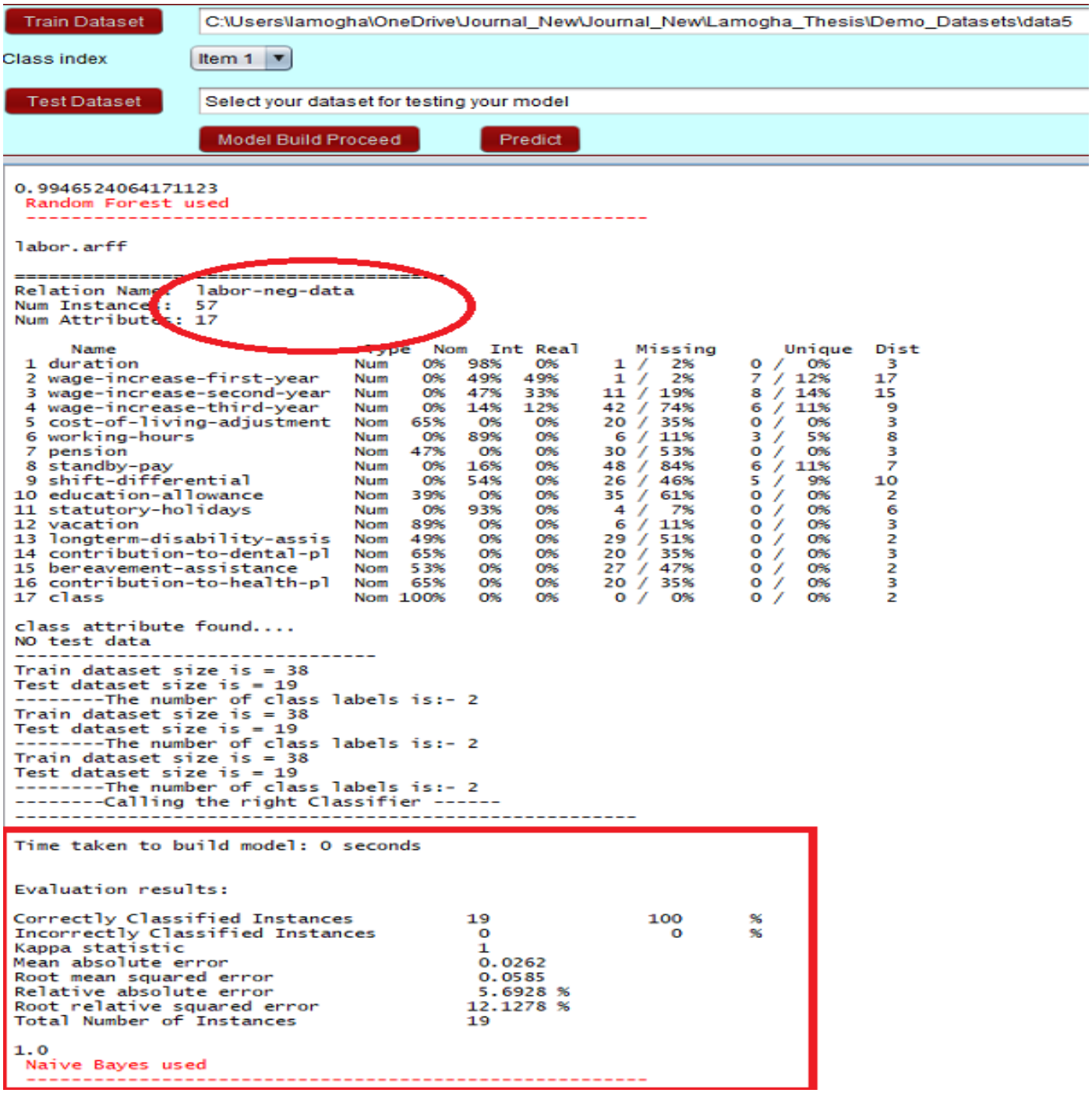


Figure 5.12: Evaluation results shown for the 'labour' data set.

Figure 5.12 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%.

```
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 5.13: Evaluation result for the 'Results' data set.

From figure 5.13 above, it is shown that Random Forest was the classifier of choice used based on the rule-based algorithm designed in this thesis.

```
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 5.14: Evaluation results for the 'ReutersCornTrain' data set.

Figure 5.14 above, shows that the zeroR classifier was automatically chosen on the 'ReutersCorn training' 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
 ------------------------------------------------------------
```

Figure 5.15: Evaluation results showing that Random Forest classifier isautomatically used to build the model for the 'Samsung-Galaxy-Gear' data set.

Figure 5.10 to Figure 5.15 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 thesis. 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.

5.2 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.

Table 5.1: Comparing autoWeka and the Hybrid autoML designed in this thesis.

		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

In Table 5.1 above, we can see the performance the hybrid autoML system designed in this thesis 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 thesis 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 thesis. 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.

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 thesis. 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.

Chapter 6

6 Conclusion and Further Work

6.1 Conclusion

In this thesis, we have presented a toolkit for automatic machine learning (ML) mode and model selection on single or multi-varying datasets.

In Chapter 1, 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 thesis eliminates all that by taking them into consideration appropriately.

Chapter 2 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.

In Chapter 3, we show and discuss some preliminary experimentations carried out in this thesis, 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.

Chapter 4 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.

Then in Chapter 5, 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 thesis 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 thesis 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.

6.2 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.

Appendices

Appendix 1

Classification of big data has several advantages and benefits, all of which includes:

1. It allows management of big data in a way that reflects organizational values.
2. It allows big data integrity management.
3. For big data management optimization.
4. Helps in determining easily what data should be distributed, how it should be distributed and to whom it should be shared with?
5. Which data needs to be kept where and who should have access to the data.
6. Better performance optimization.
7. Ease of use of information.
8. Better use of resources which may improve the revenue generation of the infrastructure.
9. Provide improved security measures and policies.
10. Can be applied in different domains and scenarios e.g. in the **health sector** (Austin, Tu, Ho, Levy, & Lee, 2013; Azar & El-Said, 2013; Hu, Palreddy, & Tompkins, 1997; Strauss, Bartko, & Carpenter, 1973; Tortajada, Robles, & García-Gómez, 2015), **geoscience** (Angus Webb et al., 2007; Baum, Tovinkere, Titlow, & Welch, 1997; Iounousse, Er-Raki, El Motassadeq, & Chehouani, 2015; Leiva-Murillo, Gomez-Chova, & Camps-Valls, 2013), **social network analysis**, **Document and text classification & filtering** (Mladeni'c & Grobelnik, 1998; Zhu, Ghahramani, & Lafferty, 2003), **multimedia data analysis** (Bankert, 1994; Haralick, Shanmugam, & Dinstein, 1973), **biological data analysis** (Achcar, Camadro, & Mestivier, 2009; T. Li, Zhang, & Ogihara, 2004), **language processing (Bird, Klein, & Loper, 2009)**, face recognition systems (Pavani et al., 2012), etc.

Appendix 2

To achieve the aims set out above, the following objectives where achieved at different stages of this thesis:

1. Studied, designed, conceptualised and developed high performing ML models on the fly in the best time possible and given limited resources (e.g. time, CPU power, etc.).
2. Used more general knowledge about ML methods (e.g. supervised, unsupervised & semi-supervised learning), as well as general knowledge about input datasets (e.g. size, class type, presence of labelled training data, absence of labelled training data, etc.) to automatically help in the decision making process.
3. Experimented extensively with Weka to determine the general knowledge and ideas that can be used.
4. Designed a three layered decision tree-based hybrid autoML model.
5. Designed and implemented a prototype of a Meta learning (learning to learn) algorithm for automatically deciding whether to invoke a supervised learning or an unsupervised learning algorithm.
6. Studied, designed, conceptualised and developed a robust self-evolving unsupervised function that allows for the derivation of clusters from scratch without having to train the model using labelled train dataset. Since the number of cluster labels is not restricted, the algorithm allows for automatic re-grouping of the clusters based on similarity and distance measurements between the clusters.

Appendix 3

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

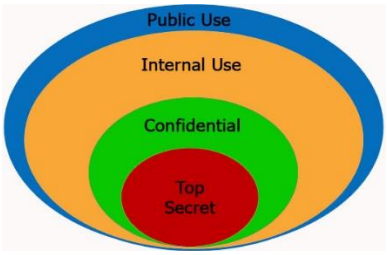
However, the actual survey itself has since been closed and results analysed.

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 4

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.

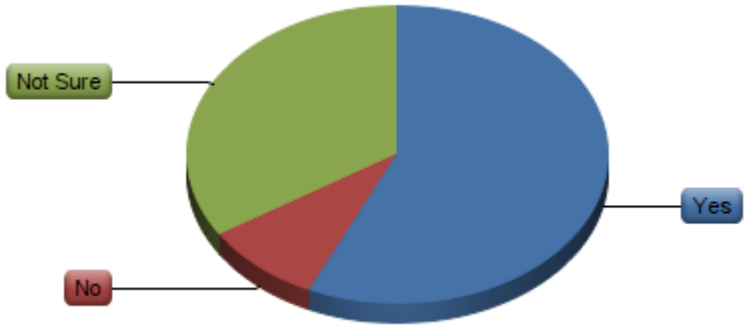


Figure 6.1: shows data from the survey carried out, that data science professionals are well aware of data classification as a good management technique.

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.

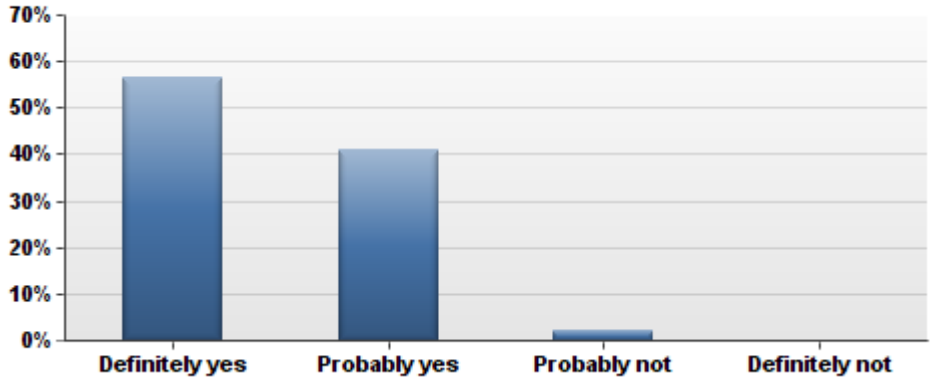


Figure 6.2: Survey results, showing data science professionals thoughts on whether big data classification measures in place, effectively improves security.

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.

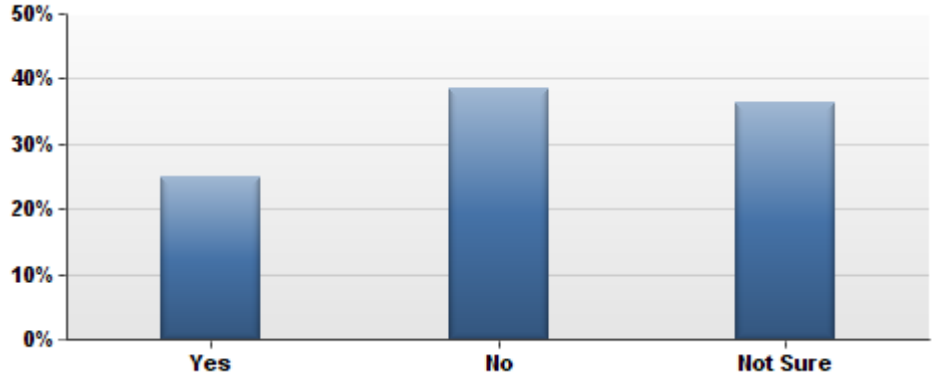


Figure 6.3: Survey results on the use of big data classification tools by several data science professionals.

Appendix 5

Lists of most datasets used throughout this project. Subsets of this list, are referred to at different points within the main content area.

Table 6.1: 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 6

Table 6.2: Area Under Curve using 10-folds cross validation

AUC										
Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
contact-lenses	0.95	0.50	-	0.82	-	0.91	0.22	0.22	0.95	0.86
cpu	-	-	-	-	-	-	-	-	-	-
cpu.with.vendor	-	-	-	-	-	-	-	-	-	-
credit-g	0.79	0.49	0.69	0.69	-	0.76	0.50	0.50	0.64	0.79
diabetes	0.82	0.50	0.73	0.77	-	0.81	0.50	0.50	0.75	0.89
glass	0.72	0.80	-	0.80	-	0.85	0.47	0.47	0.77	0.93
ionosphere	0.94	0.91	0.84	0.87	-	0.95	0.49	0.49	0.89	0.98
iris.2D	1.00	1.00	-	1.00	-	1.00	0.50	0.50	0.99	1.00
labor	0.97	0.91	0.90	0.97	-	0.86	0.47	0.47	0.70	0.94
reutersCorn-test	-	-	-	-	-	-	0.45	0.45	-	-
reutersCorn-train	-	-	-	-	-	-	0.47	0.47	-	-
segment-challenge	1.00	0.76	-	1.00	-	1.00	0.49	0.49	1.00	1.00
soybean	0.99	1.00	-	1.00	-	0.99	0.49	0.49	1.00	1.00
supermarket	0.50	0.50	0.50	0.50	-	0.50	0.50	0.50	0.50	0.50
unbalanced	0.56	0.50	0.50	0.70	-	0.52	0.43	0.43	0.43	0.72
vote	0.97	0.96	0.97	0.98	-	0.98	0.49	0.49	0.97	0.99
weather.nominal	0.58	0.50	0.69	0.69	-	0.31	0.18	0.18	0.63	0.53
weather.numeric	0.44	0.50	0.49	0.53	-	0.19	0.48	0.18	0.79	0.44

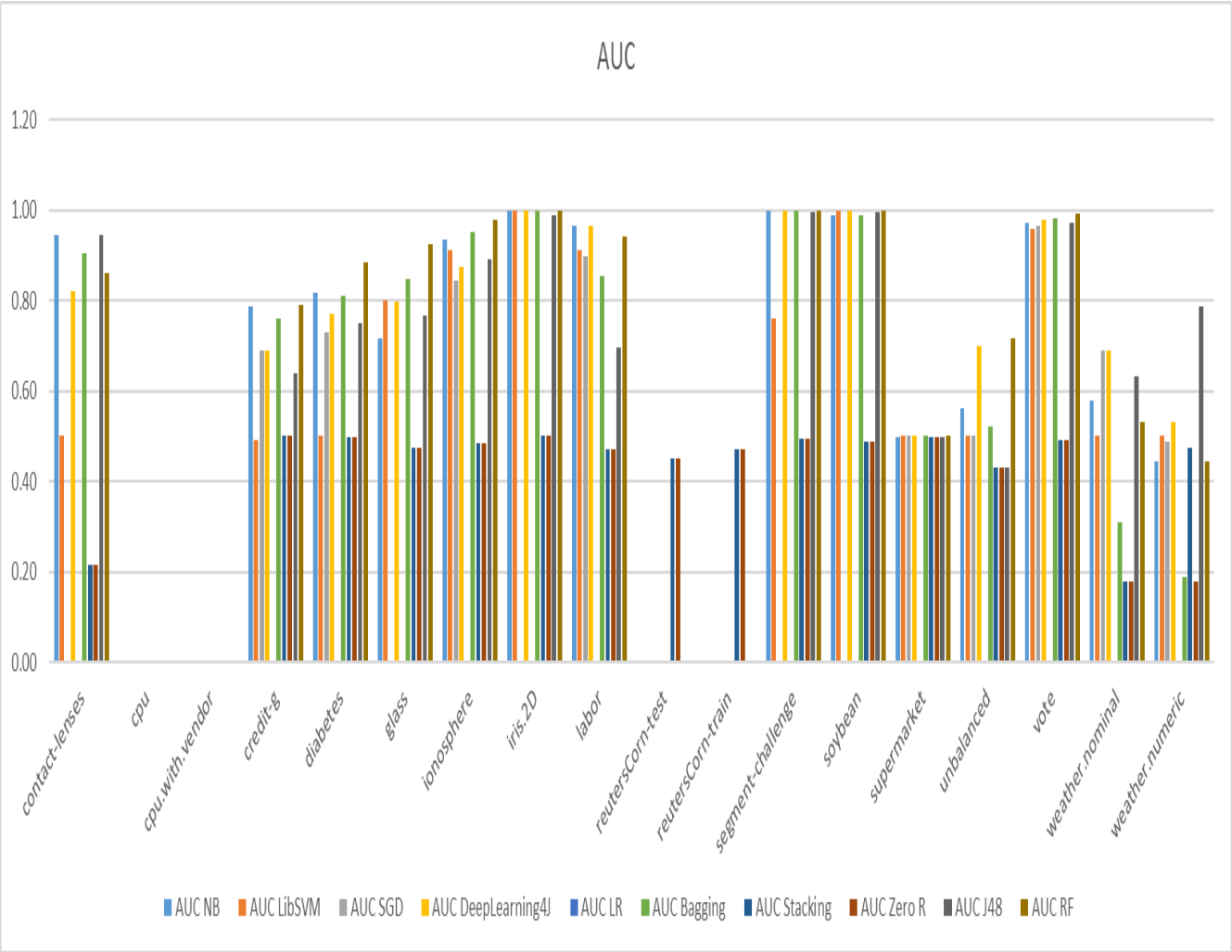


Figure 6.4: 10-folds Analyses of the Area Under the Curve performance measure.

Table 6.3: 10-folds F-Measure evaluation on the datasets.

F-MEASURE										
Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
contact-lenses	0.77	0.77	-	0.73	-	0.71	0.77	0.77	0.86	0.80
cpu	-	-	-	-	-	-	-	-	-	-
cpu.with.vendor	-	-	-	-	-	-	-	-	-	-
credit-g	0.83	0.81	0.83	0.77	-	0.83	0.82	0.82	0.80	0.85
diabetes	0.82	0.79	0.84	0.79	-	0.82	0.79	0.79	0.80	0.82
glass	0.25	0.72	-	0.56	-	0.70	0.52	0.52	0.64	0.81
ionosphere	0.78	0.90	0.81	0.79	-	0.93	0.78	0.78	0.94	0.95
iris.2D	1.00	1.00	-	1.00	-	1.00	0.50	0.50	0.99	1.00
labor	0.92	0.90	0.93	0.91	-	0.88	0.79	0.79	0.79	0.92
reutersCorn-test	-	-	-	-	-	-	0.98	0.98	-	-
reutersCorn-train	-	-	-	-	-	-	0.99	0.99	-	-
segment-challenge	0.96	0.68	-	0.98	-	0.99	0.27	0.27	0.99	1.00
soybean	0.89	1.00	-	1.00	-	0.85	0.24	0.24	0.92	0.92
supermarket	0.78	0.30	0.78	0.78	-	0.78	0.78	0.78	0.78	0.78
unbalanced	0.95	0.99	0.99	0.99	-	0.99	0.99	0.99	0.99	0.99
vote	0.92	0.96	0.97	0.96	-	0.96	0.76	0.76	0.97	0.97
weather.nominal	0.70	0.78	0.78	0.78	-	0.60	0.78	0.78	0.59	0.80
weather.numeric	0.76	0.78	0.70	0.74	-	0.67	0.78	0.78	0.74	0.74

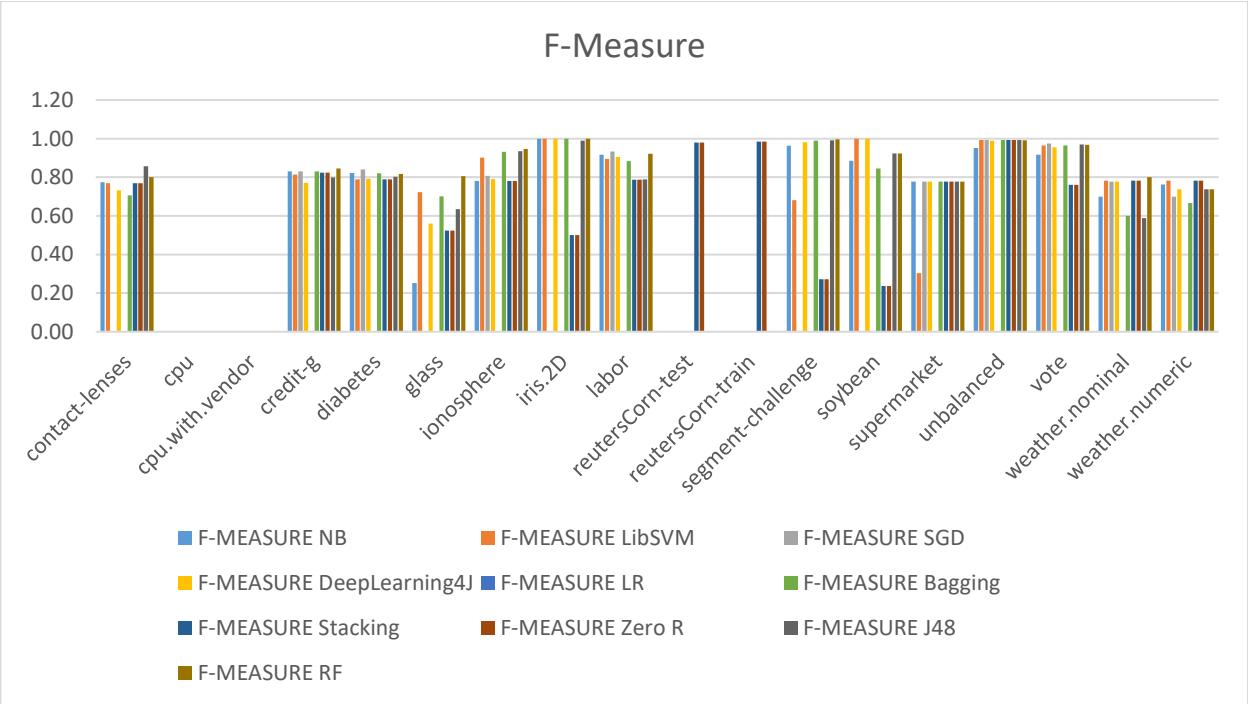


Figure 6.5: 10 Folds F-Measure Evaluation

Table 6.4: 10 Folds Mean Absolute Error Measures

MAE										
Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	Zero R	J48	RF
contact-lenses	0.25	0.25	-	0.20	-	0.31	0.38	0.38	0.15	0.23
cpu	-	-	-	205.56	41.09	34.04	96.24	96.24	-	25.61
cpu.with.vendor	-	-	-	196.13	36.97	25.28	87.66	87.66	-	13.69
credit-g	0.29	0.31	0.25	0.32	-	0.33	0.42	0.42	0.35	0.34
diabetes	0.28	0.35	0.22	0.30	-	0.32	0.45	0.45	0.32	0.31
glass	0.15	0.09	-	0.14	-	0.12	0.21	0.21	0.10	0.10
ionosphere	0.17	0.07	0.13	0.15	-	0.14	0.46	0.46	0.09	0.13
iris.2D	0.03	0.02	-	0.04	-	0.05	0.44	0.44	0.04	0.04
labor	0.10	0.07	0.09	0.07	-	0.30	0.46	0.46	0.32	0.32
reutersCorn-test	-	-	-	-	-	-	0.08	0.08	-	-
reutersCorn-train	-	-	-	-	-	-	0.06	0.06	-	-
segment-challenge	0.06	0.13	-	0.02	-	0.02	0.24	0.24	0.01	0.02
soybean	0.01	0.01	-	0.01	-	0.03	0.10	0.10	0.01	0.02
supermarket	0.46	0.58	0.36	0.46	-	0.46	0.46	0.46	0.46	0.46
unbalanced	0.11	0.01	0.01	0.02	-	0.03	0.03	0.03	0.03	0.03
vote	0.10	0.04	0.03	0.06	-	0.07	0.47	0.47	0.06	0.07
weather.nominal	0.44	0.36	0.29	0.31	-	0.52	0.48	0.48	0.42	0.44
weather.numeric	0.46	0.36	0.43	0.39	-	0.53	0.48	0.48	0.29	0.47

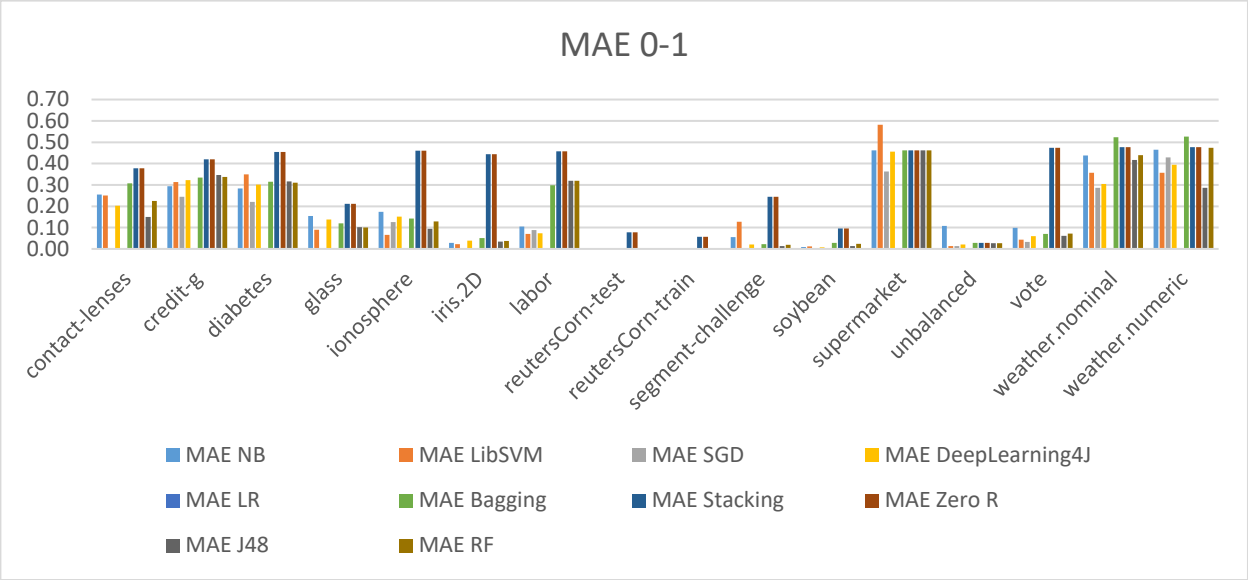


Figure 6.6: 10 folds MAE evaluation

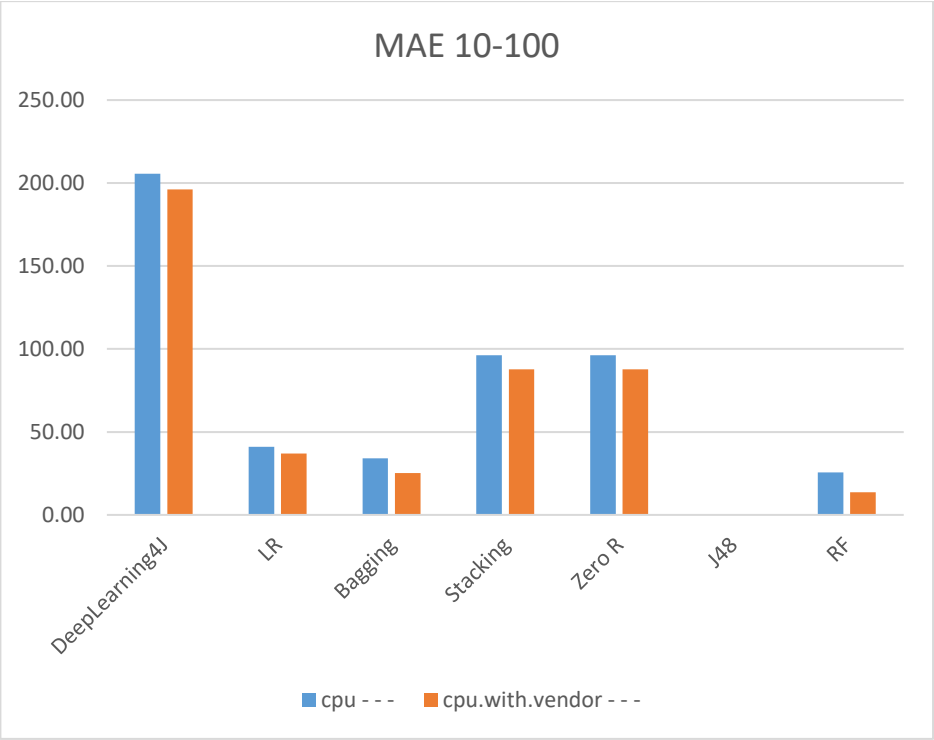


Figure 6.7: 10 folds MAE evaluated measures for cpu and cpu.with.vendor datasets.

Table 6.5: 10 folds Accuracy measures. In terms of the number of correctly classified instances.

Dataset	NB	LibSVM	SGD	DL4J	LR	Bagging	Stacking	ZeroR	J48	RF	
contact-lenses	70.83	62.50	-	66.67	-	58.33	62.50	62.50	83.33	70.83	
cpu	-	-	-	-0.12	0.90	0.90	-0.15	-0.15	-	0.95	Correlation coef
cpu.with.vendor	-	-	-	-0.12	0.93	0.92	-0.14	-0.14	-	0.97	Correlation coef
credit-g	75.40	68.70	75.50	67.90	-	74.70	70.00	70.00	70.50	76.40	
diabetes	76.30	65.10	77.99	72.92	-	75.78	65.10	65.10	73.83	75.78	
glass	0.15	68.69	-	56.07	-	72.43	35.51	35.51	66.82	79.91	
ionosphere	82.62	93.45	87.46	85.47	-	91.17	64.10	64.10	91.45	92.88	
iris.2D	96.00	96.67	-	95.33	-	95.33	33.33	33.33	96.00	94.00	
labor	89.47	92.98	91.23	92.98	-	84.21	64.91	64.91	73.68	89.47	
reutersCorn-test	-	-	-	-	-	-	96.03	96.03	-	-	
reutersCorn-train	-	-	-	-	-	-	97.10	97.10	-	-	
segment-challenge	81.07	55.40	-	93.40	-	95.87	15.73	15.73	95.73	97.87	
soybean	92.97	88.73	-	92.83	-	85.65	13.47	13.47	91.51	92.97	
supermarket	63.71	41.78	63.71	63.71	-	63.71	63.71	63.71	63.71	63.71	
unbalanced	90.77	98.60	98.60	97.90	-	98.60	98.60	98.60	98.60	98.48	
vote	90.11	95.63	96.78	94.48	-	95.63	61.38	61.38	96.32	96.09	

weather.nominal	57.14	64.29	71.43	71.43	-	42.86	64.29	64.29	50.00	71.43
weather.numeric	64.29	64.29	57.14	64.29	-	50.00	64.29	64.29	64.29	64.29

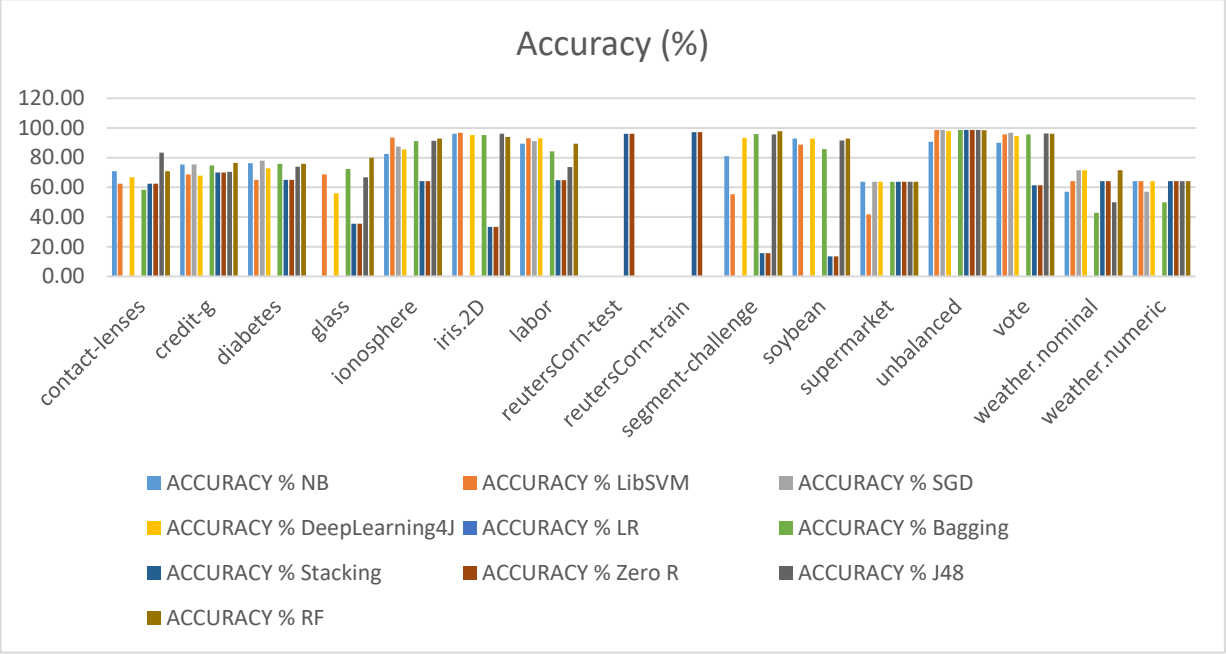


Figure 6.8: 10 folds % Accuracy

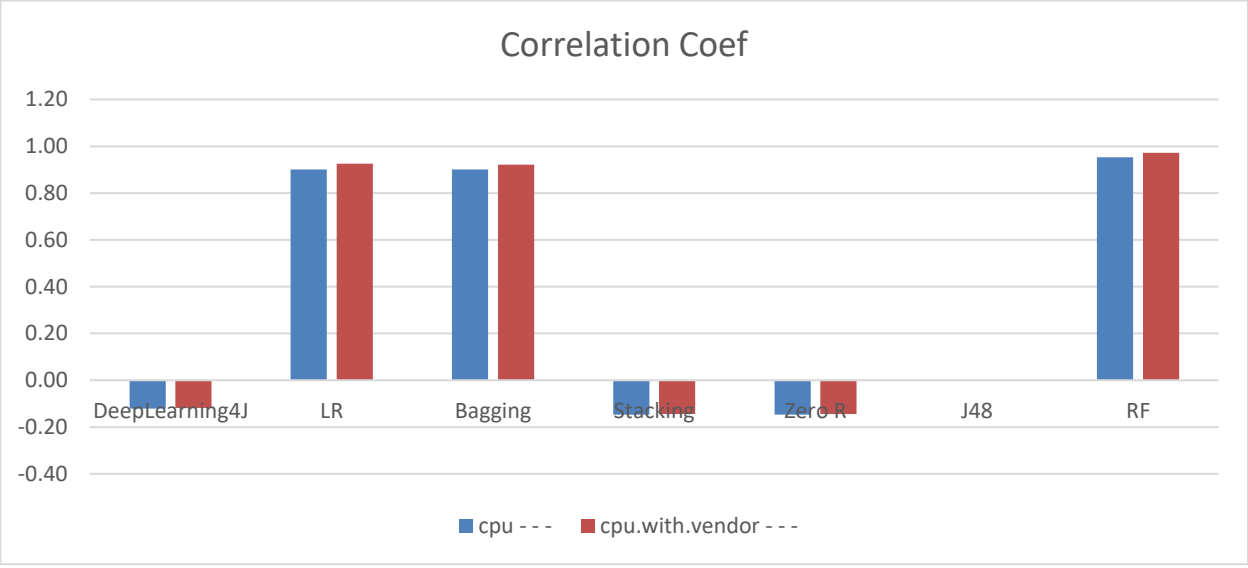


Figure 6.9: 10 folds Correlation Coefficient of cpu and cpu.with.vendor

Appendix 7

Weka GUI

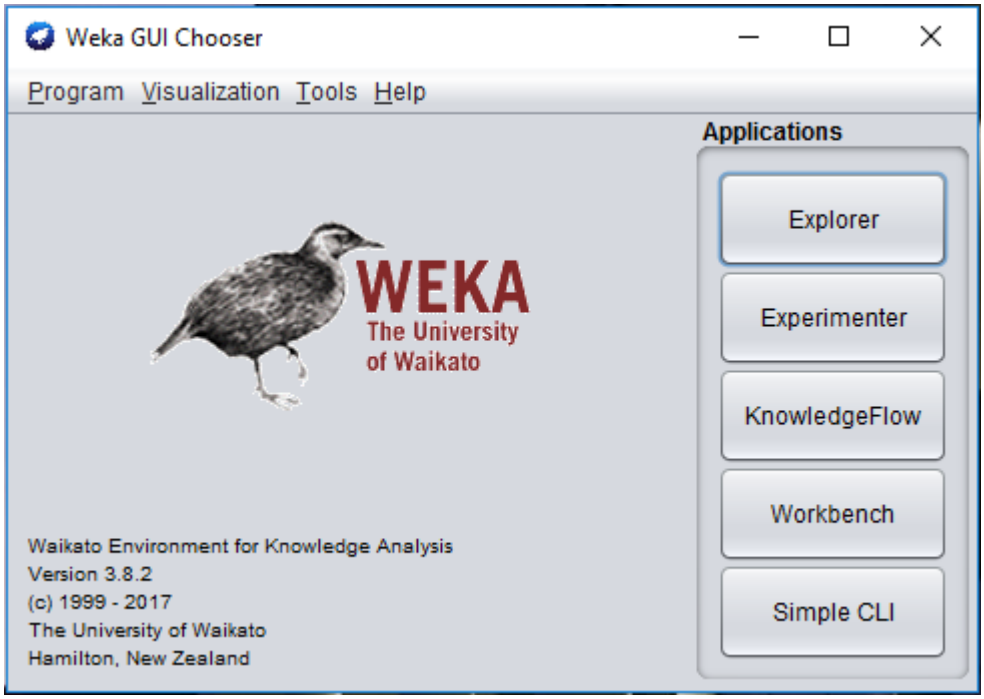


Figure 6.10: Weka GUI when initially launched.

The Explorer

Clicking on the 'Explorer' tab after launching the Weka GUI, launches the Weka explorer.

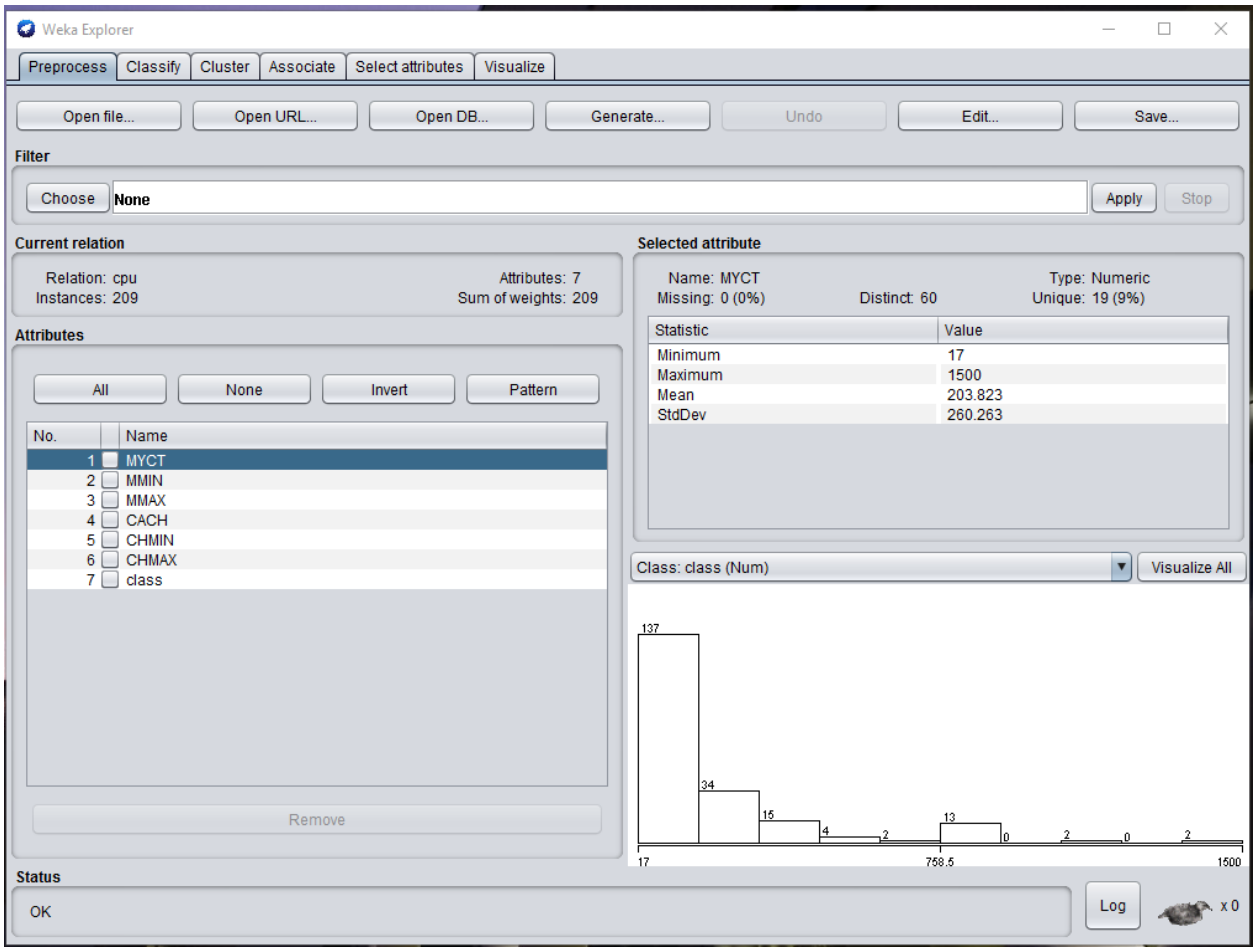


Figure 6.11: The Weka Explorer GUI

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.

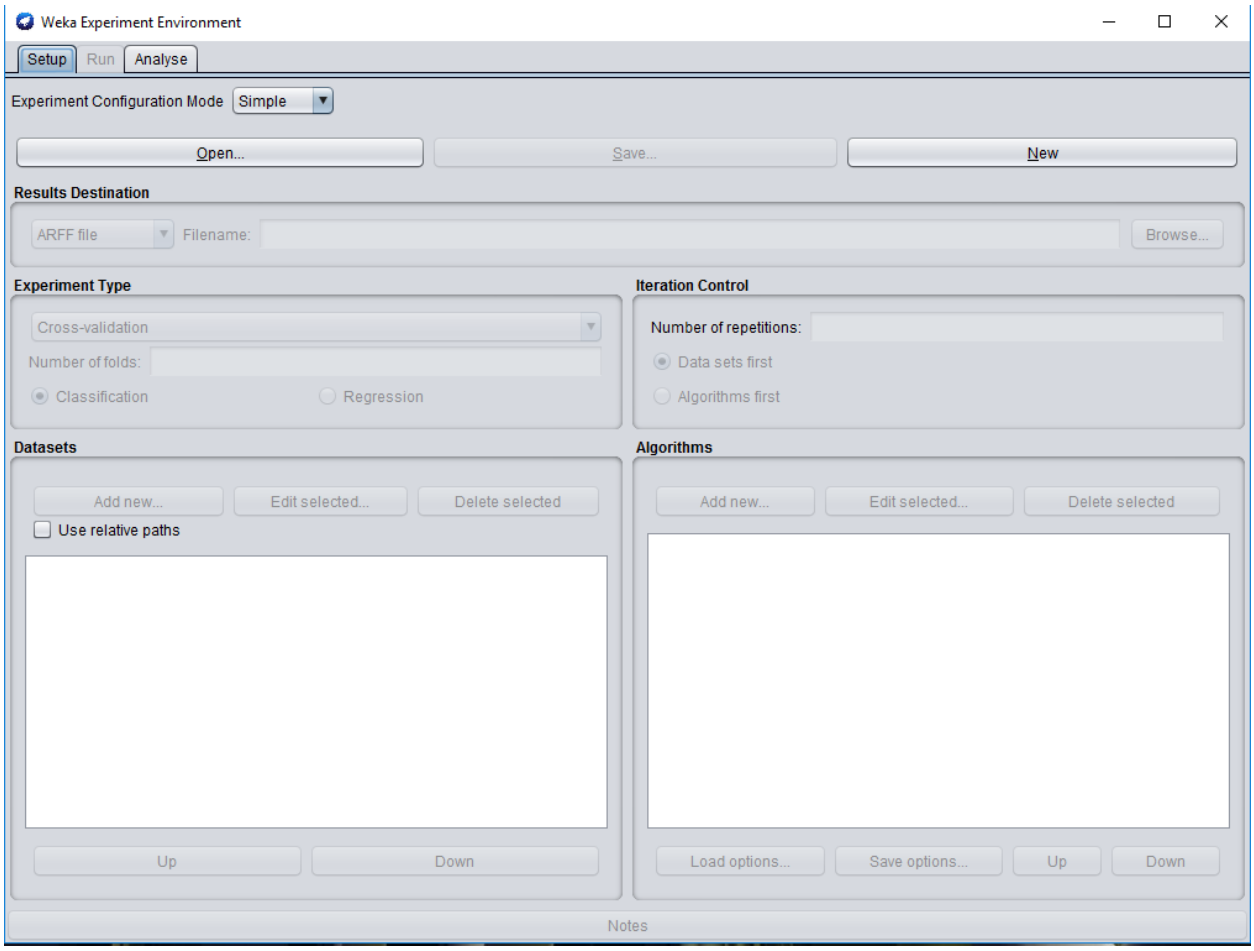


Figure 6.12: The Weka Experimenter GUI

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.

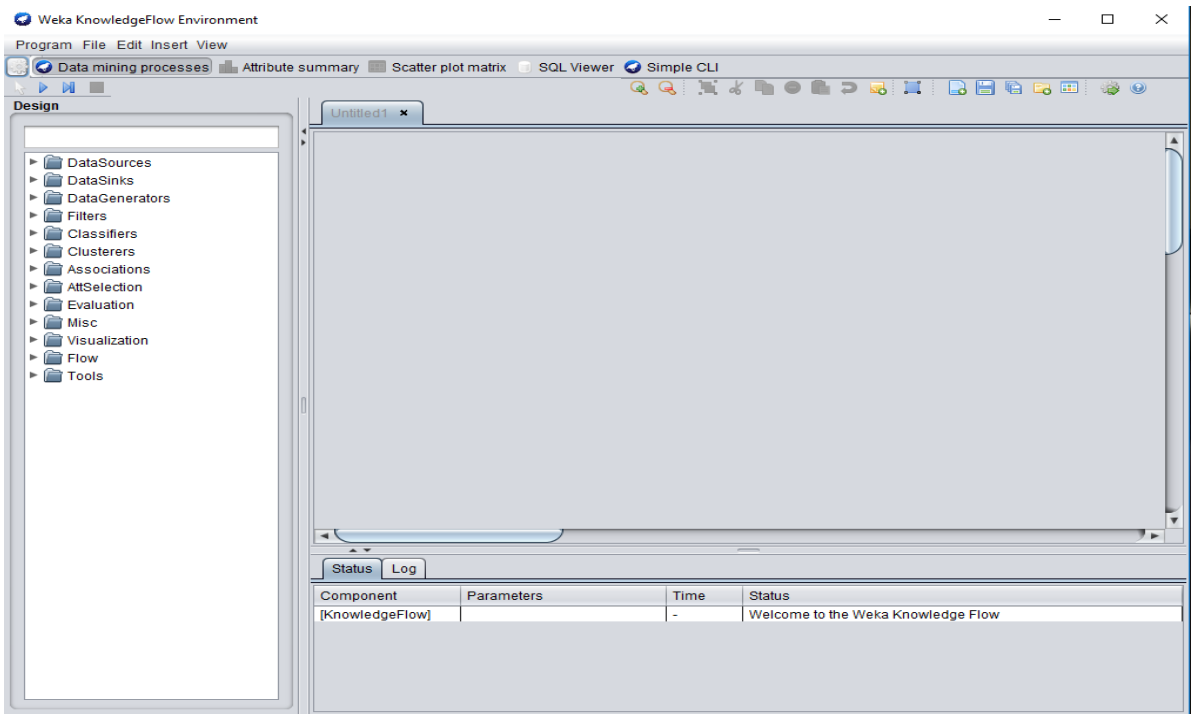


Figure 6.13: The Weka Knowledge Flow GUI

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.

Bibliography

Achcar, F., Camadro, J.-M., & Mestivier, D. (2009). AutoClass@ IJM: a powerful tool for Bayesian classification of heterogeneous data in biology. Nucleic acids research, gkp430.

Aggarwal, C. C. (2014). Data Classification : Algorithms and Applications. Hoboken: Chapman and Hall/CRC.

Aggarwal, C. C. (2014a). Instance-Based Learning: A Survey. Data Classification: Algorithms and Applications, 157.

Aggarwal, C. C. (2014b). An Introduction to Data Classification. Data Classification: Algorithms and Applications, 1.

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). Hoboken: Chapman and Hall/CRC.

Aha, D. W. (1992). Generalizing from case studies: A case study. Paper presented at the Proc. of the 9th International Conference on Machine Learning.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. Machine learning, 6(1), 37-66.

Aher, S. B., & Lobo, L. (2012). Comparative study of classification algorithms. International Journal of Information Technology, 5(2), 239-243.

Akerkar, R. (2013). Big Data Computing. Hoboken: CRC Press.

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

Angelov, P., & Yager, R. (2012). A new type of simplified fuzzy rule-based system. International Journal of General Systems, 41(2), 163-185.

Angus Webb, J., Bond, N. R., Wealands, S. R., Mac Nally, R., Quinn, G. P., Vesk, P. A., & Grace, M. R. (2007). Bayesian clustering with AutoClass explicitly recognises uncertainties in landscape classification. Ecography, 30(4), 526-536.

Austin, P. C., Tu, J. V., Ho, J. E., Levy, D., & Lee, D. S. (2013). Using methods from the data-mining and machine-learning literature for disease classification and prediction: a case study examining classification of heart failure subtypes. Journal of clinical epidemiology, 66(4), 398-407.

Azar, A. T., & El-Said, S. A. (2013). Probabilistic neural network for breast cancer classification. Neural Computing and Applications, 23(6), 1737-1751. doi:10.1007/s00521-012-1134-8

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.

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.

Barandiaran, I. (1998). The random subspace method for constructing decision forests. IEEE transactions on pattern analysis and machine intelligence, 20(8).

Baum, B. A., Tovinkere, V., Titlow, J., & Welch, R. M. (1997). Automated cloud classification of global AVHRR data using a fuzzy logic approach. Journal of Applied Meteorology, 36(11), 1519-1540.

Ben-David, A. (2008). Comparison of classification accuracy using Cohen's Weighted Kappa. Expert Systems with Applications, 34(2), 825-832.

Bertini, J. R., & Zhao, L. (2013, 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.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive Online Analysis. J. Mach. Learn. Res., 11, 1601-1604.

Bird, S., Klein, E., & Loper, E. (2009). Categorizing and Tagging words & Learning to Classify Text Natural language processing with Python: " O'Reilly Media, Inc.".

Bishop, C. M. (2006). Pattern recognition and machine learning: springer.

Blunsom, P. (2004). Hidden markov models. Lecture notes, 15, 18-19.

Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). Classification and regression trees: CRC press.

Bremner, D., Demaine, E., Erickson, J., Iacono, J., Langerman, S., Morin, P., & Toussaint, G. (2005). Output-sensitive algorithms for computing nearest-neighbour decision boundaries. Discrete & Computational Geometry, 33(4), 593-604.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 27.

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.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. Machine learning, 3(4), 261-283.

Cleary, J. G., & Trigg, L. E. (1995). K*: An instance-based learner using an entropic distance measure Machine Learning Proceedings 1995 (pp. 108-114): Elsevier.

Cohen, W. W. (1995). Fast effective rule induction. Paper presented at the Proceedings of the twelfth international conference on machine learning.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

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.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. Information Theory, IEEE Transactions on, 13(1), 21-27.

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.

Datarobot, & Triffacta. The 8 Core Activities For Automated Data Preparation & Machine Learning: Trifacta & Datarobot. Retrieved from https://www.datarobot.com/resource/trifacta-8-core-activities-automated-data-preparation-machine-learning/.

Doug, L. (2001). Data Management: Controlling Data Volume, Velocity, and Variety: Application Delivery Strategies", META Group (currently with Gartner).

Dua, D., & Karra Taniskidou, E. (2017). UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets.html

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.

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.

Fan, W., & Bifet, A. (2013). Mining big data: current status, and forecast to the future. ACM SIGKDD Explorations Newsletter, 14(2), 1-5.

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.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. Machine learning, 2(2), 139-172.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. The 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.

Frank, E., Hall, M., & Pfahringer, B. (2002). Locally weighted naive bayes. Paper presented at the 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.

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.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter, 11(1), 10-18.

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 Advances in Knowledge Discovery and Data Mining (pp. 107-120): Springer.

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.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79(8), 2554-2558.

Hu, Y. H., Palreddy, S., & Tompkins, W. J. (1997). A patient-adaptable ECG beat classifier using a mixture of experts approach. Biomedical Engineering, IEEE Transactions on, 44(9), 891-900.

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.

Jacques, J., & Preda, C. (2014). Functional data clustering: a survey. Advances in Data Analysis and Classification, 8(3), 231-255. doi:10.1007/s11634-013-0158-y

Jain, A., Murty, M., & Flynn, P. (1999). Data clustering: a review. ACM Computing Surveys (CSUR), 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.

Jain, M., Dua, P., & Lukiw, W. (2013). Data adaptive rule-based classification system for Alzheimer classification. J Comput Sci Syst Biol, 6, 291-297.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. Paper presented at the 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. The 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.

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/

Law, Y.-N., & Zaniolo, C. (2005). An adaptive nearest neighbor classification algorithm for data streams Knowledge Discovery in Databases: PKDD 2005 (pp. 108-120): Springer.

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 (pp. 17-52): 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, 20(15), 2429-2437.

Li, X.-L., & Liu, B. Rule-based Classification.

Lin, Z., Yan, C., Yan, L., & Nan, L. (2008, 19-21 Dec. 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.

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.

Liu, W., Liu, H., Tao, D., Wang, Y., & Lu, K. (2015). Manifold regularized kernel logistic regression for web image annotation. Neurocomputing.

Loh, W. Y. (2011). Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1), 14-23.

Lu, N., Mabu, S., Mabu, S., Li, W., Hirasawa, K., & Hirasawa, K. (2010, 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. Paper presented at the Proceedings of the fourth international conference on knowledge discovery and data mining.

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).

Majnik, M., & Bosnic, Z. (2013). ROC analysis of classifiers in machine learning: A survey. Intelligent Data Analysis, 17(3), 531-558.

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

McCallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. Paper presented at the Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. Neurocomputing, 55(1), 169-186.

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.

Moon, T. K. (1996). The expectation-maximization algorithm. IEEE Signal processing magazine, 13(6), 47-60.

Mossman, D. (1999). Three-way rocs. Medical Decision Making, 19(1), 78-89.

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.

Ng, K., & Lippmann, R. P. (1991). A comparative study of the practical characteristics of neural network and conventional pattern classifiers. Paper presented at the Advances in Neural Information processing systems.

Nielsen, J. D., Rumí, R., & Salmerón, A. (2009). Supervised classification using probabilistic decision graphs. Computational Statistics and 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

Owen, S., Anil, R., Dunning, T., & Friedman, E. (2011). Mahout in action: Manning Shelter Island.

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.

Pavani, S.-K., Sukno, F. M., Delgado-Gomez, D., Butakoff, C., Planes, X., & Frangi, A. F. (2012). An Experimental Evaluation of Three Classifiers for Use in Self-Updating Face Recognition Systems. Information Forensics and Security, IEEE Transactions on, 7(3), 932-943.

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 III, W. F., 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.

Quinlan, J. R. (2014). C4. 5: programs for machine learning: Elsevier.

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 (pp. 532-538): Springer.

Rice, J. R. (1975). The algorithm selection problem.

Richard, M. D., & Lippmann, R. P. (1991). Neural network classifiers estimate Bayesian a posteriori probabilities. Neural computation, 3(4), 461-483.

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.

Sharma, N., Bajpai, A., & Litoriya, M. R. (2012). Comparison the various clustering algorithms of Weka tools. facilities, 4(7).

Sheikholesalmi, F., Mardani, M., & Giannakis, G. B. (2014, 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.

Small, M. (2013). Securing Big Data. ITNOW, 55(3), 10-11.

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys (CSUR), 41(1), 6.

Sokal, R. R. (1974). Classification: purposes, principles, progress, prospects. Science, 185(4157), 1115-1123.

Sparks, E. R., Talwalkar, A., Haas, D., Franklin, M. J., Jordan, M. I., & Kraska, T. (2015). Automating model search for large scale machine learning. Paper presented at the Proceedings of the Sixth ACM Symposium on Cloud Computing.

Strauss, J. S., Bartko, J. J., & Carpenter, W. T. (1973). The use of clustering techniques for the classification of psychiatric patients. The British Journal of Psychiatry, 122(570), 531-540.

Suthaharan, S. (2014). Big data classification: problems and challenges in network intrusion prediction with machine learning. ACM SIGMETRICS Performance Evaluation Review, 41(4), 70-73. doi:10.1145/2627534.2627557

Tang, J., Alelyani, S., & Liu, H. (2014). Feature selection for classification: A review. Data Classification: Algorithms and Applications. Editor: Charu Aggarwal, CRC Press In Chapman & Hall/CRC Data Mining and Knowledge Discovery Series.

Tankard, C. (2012). Big data security. Network security, 2012(7), 5-8.

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.

Tortajada, S., Robles, M., & García-Gómez, J. M. (2015). Incremental Logistic Regression for Customizing Automatic Diagnostic Models Data Mining in Clinical Medicine (pp. 57-78): Springer.

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.

Tung, A. K. (2009). Rule-based Classification Encyclopedia of Database Systems (pp. 2459-2462): 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.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. Artificial Intelligence Review, 18(2), 77-95.

Wang, X., & Pardalos, P. M. (2015). A Survey of Support Vector Machines with Uncertainties. Annals of Data Science, 1(3-4), 293-309.

Weisberg, S. (2005). Applied linear regression (Vol. 528): John Wiley & Sons.

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.

Yan, Y., Zhu, Q., Shyu, M.-L., & Chen, S.-C. (2016). A Classifier Ensemble Framework for Multimedia Big Data Classification.

Yunck, T. P. (1976). A technique to identify nearest neighbors. Systems, Man and Cybernetics, IEEE Transactions on(10), 678-683.

Zhang, C., & Zhang, S. (2002). Association rule mining: models and algorithms: Springer-Verlag.

Zhao, Z., & Liu, H. (2007). Semi-supervised Feature Selection via Spectral Analysis. Paper presented at the SDM.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. Paper presented at the ICML.