UNIVERSITY OF Hull

AUTOENCODER FOR CLINICAL DATA ANALYSIS AND CLASSIFICATION: DATA IMPUTATION, DIMENSIONAL REDUCTION, AND PATTERN RECOGNITION

Being a Thesis submitted for the Degree of

Doctor of Philosophy

in the University of Hull

Department of Computer Science

By

Mohammad Al Khaldy BSc, MSc.

July 2017

ABSTRACT

Over the last decade, research has focused on machine learning and data mining to develop frameworks that can improve data analysis and output performance; to build accurate decision support systems that benefit from real-life datasets. This leads to the field of clinical data analysis, which has attracted a significant amount of interest in the computing, information systems, and medical fields. To create and develop models by machine learning algorithms, there is a need for a particular type of data for the existing algorithms to build an efficient model. Clinical datasets pose several issues that can affect the classification of the dataset: missing values, high dimensionality, and class imbalance. In order to build a framework for mining the data, it is necessary first to preprocess data, by eliminating patients' records that have too many missing values, imputing missing values, addressing high dimensionality, and classifying the data for decision support.

This thesis investigates a real clinical dataset to solve their challenges. Autoencoder is employed as a tool that can compress data mining methodology, by extracting features and classifying data in one model. The first step in data mining methodology is to impute missing values, so several imputation methods are analysed and employed. Then high dimensionality is demonstrated and used to discard irrelevant and redundant features, in order to improve prediction accuracy and reduce computational complexity. Class imbalance is manipulated to investigate the effect on feature selection algorithms and classification algorithms.

The first stage of analysis is to investigate the role of the missing values. Results found that techniques based on class separation will outperform other techniques in predictive ability. The next stage is to investigate the high dimensionality and a class imbalance. However it was found a small set of features that can improve the classification performance, the balancing class does not affect the performance as much as imbalance class.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude to my teachers, colleagues, and friends. Without them, the results presented in this thesis could not have been accomplished.

First and foremost I would like to express my deepest gratitude to those who supported me during the course of the PhD. I am especially thankful to Dr Chandra Kambhampati for his tireless support as my supervisor, his guidance and constructive criticism during the course of this research. Chandra has supported me academically through the rough road to finishing this thesis and without his persistent help this thesis would not have been possible.

I am also very grateful to Professor Ping Jiang who passed away during my study, I would like to acknowledge his support, enthusiastic encouragement and insightful advice, which helped me greatly.

I would like to thank Professor Yiannis Papadopoulos and Dr Yongqiang Cheng, my panel members, for their constructive criticisms and insightful advice, which contributed to the improvement of the overall project.

I would also like to thank all the staff members in the Department of Computer Science.

I would like to acknowledge and extend my heartfelt gratitude to my fellow colleagues, who made the PhD experience more bearable through their support and friendship in good and bad times. I say thank you- Dr Sohag Kabir, Dr Lisa Moore, Dr Youcef Gheraiba, Dr Lamis Al-Qoran, Tareq Aljaber, John Stamford, Robert Munnoch, Brian Peach, Seyed Sadegh Zadeh, Dongfei Xue, Sahar Archi, Syed Kazmi, Steven Balding, Lucie Leveque, Walid Tuari and Luis Torrao.

Finally, I would like to give my deepest thanks to my dear father Ali, and my beloved mother Tamam. This thesis is dedicated to my lovely wife Garam and my sons (Almothana, Taym, and Abdulrahman), for all their love and support, I would like to thank them for their patience and support throughout this journey, both emotionally and mentally. Also I would like to give my deepest thanks to my brothers, my sisters, and other relatives for their continued support and encouragement throughout my study and for teaching me many useful lessons for life.

DECLARATION

Parts of the work reported in this thesis were published in the following research papers:

1- Al khaldy, M., and Kambhampati, C. 2016. Performance Analysis of Various Missing Value Imputation Methods on Heart Failure Dataset. *SAI Intelligent Systems Conference*. London, UK: Springer.

2- Al Khaldy, M., Kambhampati, C., Cleland, J., Clark, A. 2017. *Selecting Relevant Features in a Heart Failure Dataset to Improve Outcome Prediction.*

*Submitted to International Journal of Automation and Computing.

3- Al Khaldy, M. and Kambhampati, C. 2017. *Resampling Imbalanced Class and the Effectiveness of Feature Selection Methods for Heart Failure Dataset*.

*Accepted by: The International Arab Conference on Information Technology (ACIT'2017), Tunisia, 2017.

LIST OF ABBREVIATIONS

ANN: Autoencoder Neural Network.

BP: Backpropagation.

CDSS: Clinical Decision Support System.

Cfs: Correlation-based Feature Selection.

CMCI: Concept Most Common Imputation.

CPOE: Computer-based physician order entry.

CRISP-DM: CRoss Industry Standard Process for Data Mining.

DML: Deep Machine Learning

DSS: Decision Support Systems.

EuroRec: European Institute for Electronic Health Records.

EMI: Expectation Maximization.

EMR: Electronic Medical Records.

FN: false negative.

FP: false positive.

ICA: Independent Component Analysis.

KDD: Knowledge Discovery in Database.

KNNI: K Nearest Neighbour Imputation.

KPCA: Kernel Principal Component Analysis.

LDA: Linear Discriminant Analysis.

LR: Likelihood ratios.

MAR: Missing at Random

MCAR: Missing Completely at Random.

MCI: Most Common Imputation.

MI: Multiple Imputation.

ML: Machine Learning.

MNAR: Missing Not at Random.

MSE: Mean Squared Error.

NPV: Negative Predictive Value.

PCA: Principal Component Analysis.

PPV: Positive Predictive Value.

PRAISE1: Prospective Randomized Amlodipine Survival Evaluation.

RBM: Restricted Boltzmann Machine.

REPTree: Reduced Error Pruning Tree.

RF: Random Forest.

ROC: Receiver Operating Characteristic.

SBS: sequential backward selection.

SFS: sequential forward selection.

SHFM: Seattle Heart failure model.

SVD: Single Value Decomposition.

SVM: Support Vector Machine.

TN: true negative.

TP: true positive.

NOTATIONS

$x_{i,j}$	Each data object, each data element.
μ_{ij}	Mean of each variable.
σ_{ij}	Standard deviation of each variable.
σ_{ij}^2	Variance of each variable.
n	Number of dataset attributes.
N	Number of samples.
$P(.)$	Probability.
F	Features (input features)
f_i	Feature i; $i = 1,2, \dots n$
Y	Output space/response variable/target
S	Subsets
S_i	Subset i
g	Maps from F to Y
T	Training
P	Probability
r	Vector
v	Values
C	Class label
$\overline{r_{zi}}$	Average of correlation (variable to concept)
$\overline{r_{ii}}$	Average of correlation (variable to variable
χ^2	Chi-squared
O	Observed instances
E	Expected instance
$R(i)$	Pearson's correlation
H	Nearest Hit
M	Nearest Miss
R	Random sample
W	Weight of attribute
$H(T)$	Entropy of training set
$IG(F)$	Information gain for feature F
$d(x, y)$	Distance between (x, y)
w	Weight.
b	Bias

TABLE OF CONTENTS

ABSTRACT i
ACKNOWLEDMENT ii
DECLARATION iii
LIST OF ABBREVIATIONS iv
NOTATIONS vi
LIST OF FIGURES x
LIST OF TABLES xii

Contents

Chapter 1. Introduction 1
1.1 Background 1
1.2 Motivation and Scope 3
1.2.1 Research dataset 5
1.2.2 Clinical data challenges 6
1.3 Research Questions 8
1.4 Aim and Objectives 9
1.5 Thesis Structure 10
Chapter 2. Literature Review on Clinical Data Mining 12
2.1 Introduction 12
2.2 Knowledge Discovery and Data Mining 13
2.2.1 Data mining methodologies. 15
2.2.2 Medical data mining 18
2.2.3 Data mining techniques 22
2.3 Missing Values and Imputation Techniques 23
2.3.1 Single imputation: 24
2.3.2 Multiple imputations MI: 26
2.3.3 Machine learning (ML) 27
2.4 High Dimensionality 27
2.4.1 Feature selection 28
2.4.2 Feature extraction 30
2.5 Pattern Recognition 31
2.5.1 Patterns Measures 32
2.6 Classification 33
2.6.1 Classification methods 35
2.7 Machine Learning Process 42
2.7.1 Machine learning models 43

2.7.2 Training methods 44
2.8 Neural Networks 46
2.8.1 Autoencoder Neural Network 48
2.8.2 Feedforward backpropagation learning process 49
2.8.3 Conjugate Gradient Backpropagation 50
2.8.4 Stacked autoencoder 50
2.8.5 Restricted Boltzmann Machine (RBM) 51
2.8.6 Stacked RBM 52
2.9 Summary 52
Chapter 3. Effect of Imputation on Classification 54
3.1 Introduction 54
3.2 Missing values imputation 55
3.2.1 Most Common Imputation (MCI) 56
3.2.2 Concept Most Common Imputation (CMCI) 57
3.2.3 Expectation maximisation imputation (EM) 57
3.2.4 K-Nearest Neighbour Imputation (KNNI) 58
3.2.5 K-mean imputation 59
3.2.6 Support Vector Machine (SVM) imputation 60
3.3 Experiments and Results 60
3.3.1 Chronic Heart Failure Dataset 60
3.3.2 Evaluation of Classification Performance 61
3.3.3 Analysis of Results 64
3.3.4 Numeric complexity imputation algorithms 65
3.4 Conclusion 69
Chapter 4. Dimensionality Reduction - feature selection 70
4.1 Introduction 70
4.2 Feature Selection Problem 71
4.3 Materials and Methods 73
4.3.1 Data pre-processing 73
4.3.2 Feature Selection Methods 73
4.3.3 Numeric complexity of feature selection algorithms 81
4.4 Analysis Results 82
4.4.1 The common features selected 89
4.5 Conclusion 91
Chapter 5. Effect of Class Imbalance on Feature Selection and Classification 93
5.1 Introduction 93
5.2 Class Imbalance 93

5.3 Evaluation Measures 96
5.4 Class Balancing Techniques 97
5.5 Analysis Results 98
5.5.1 Feature selection with balanced data 102
5.6 Training Datasets on Balance Data 103
5.7 Conclusion 108
Chapter 6. Autoencoder Framework for Dimensionality Reduction and Classification .. 110
6.1 Introduction 110
6.2 Artificial Neural Network (ANN) 111
6.2.1 Activation Functions 112
6.2.2 The Perceptron Function Space 112
6.3 Multilayer Neural Network (MLP) 113
6.3.1 Feed-forward architecture 113
6.3.2 Learning process 114
6.4 Deep Neural Networks 115
6.4.1 Autoencoder Architecture 116
6.4.2 Autoencoder Parameters 119
6.4.3 Autoencoder Algorithm 122
6.5 Analysis Results 124
6.5.1 One Hidden Layer 124
6.5.2 Two Hidden Layers 127
6.5.3 Three Hidden Layers 130
6.5.4 Autoencoder with class balance 131
6.6 Conclusion 133
Chapter 7. Conclusion and Future Research 135
7.1 Introduction 135
7.2 Contributions of the research 136
7.3 Scope for future research 139
BIBLIOGRAPHY 141
Appendix I: Variables of the Hull LifeLab Dataset. 165
Appendix II: Statistics on the Research dataset for heart failure from HYMS. 167
Appendix III: Performance Results of Applying Several Hidden Layers for Autoencoder..... 169

LIST OF FIGURES

Figure 2-1 KDD, key steps in an iterative and interactive process 14
Figure 2-2 CRISP-DM Methodology 16
Figure 2-3 SEMMA Methodology (Original from SAS Institute) 18
Figure 2-4 Types of Data Mining Techniques 22
Figure 2-5 Example to Building a Decision Tree 36
Figure 2-6 Example of Random Forest 37
Figure 2-7 KNN algorithm example 39
Figure 2-8 SVM algorithm example 41
Figure 2-9 Single neuron architecture 46
Figure 2-10 Model of a Single Neuron, see (Haykin [2009]) 47
Figure 2-11 The Architecture of Autoencoder with One Hidden Layer 48
Figure 2-12 Deep Learning, One hidden layer 50
Figure 2-13 Deep Learning, Three hidden layer 51
Figure 3-1 Process of Imputation of Missing Values in a Dataset 56
Figure 3-2 Support Vector Machine. 60
Figure 3-3 ROC curve example. 63
Figure 3-4 The line Charts that Shows the Outcomes of Imputation Methods Using RF method for Classification, (a) Accuracy, (b) Sensitivity, and (c) Specificity 67
Figure 3-5 The Complexity Time of Different Classification Algorithms Used to Classify the Dataset Imputed By Different Imputation Methods 68
Figure 4-1 Pearson's Correlation between Features with High Positive Correlation in the Thesis Dataset. 77
Figure 4-2 Relief algorithm 77
Figure 4-3 The Wrapper Method for Feature Selection 79
Figure 4-4 Hill-Climbing Search Algorithm. 79
Figure 4-5 Algorithm Iterative Dimensionality Reduction for SVM 80
Figure 4-6 Feature selection process (Dash & Liu, 1997) 81
Figure 4-7 The Accuracy Obtained by Ranking Feature Selection Methods for Different Numbers of Features Used 84
Figure 4-8. Performance measures using three different classification methods; (a) Specificity (b) Sensitivity 88
Figure 4-9 Box plot whiskers for different feature ranking methods in feature selection 89
Figure 5-1 Oversampling increases the minority class by copying instances. Under-sampling removes instances from the majority class 96
Figure 5-2 The trade-off between Sensitivity and Specificity 97
Figure 6-1 Neuron architecture 111
Figure 6-2 Activation function types 113
Figure 6-3 Feedforward neural network 114
Figure 6-4 Autoencoder Architecture 117
Figure 6-5 Pseudo Code to Programming Autoencoder Model 123
Figure 6-6 ROC curve for Autoencoder with 30 neurons in One Hidden Layer 127
Figure 6-7 ROC curve for the two Hidden Layers , (a) Hidden Layers [30,40] ,(b) Hidden Layers [35,30] 130

LIST OF TABLES

Table 1-1: The Hull-Life-Lab Dataset Distribution of the Classes 8
Table 2-2 Classification methods pros and cons 41
Table 3-1: Confusion Matrix 62
Table 3-2: Performance Measures for Several Imputation Methods Implemented on Heart Failure Dataset, Using Different Classification Methods 66
Table 4-1: The truth table of $\boldsymbol{Y = f1 \oplus f2}$ where $\boldsymbol{f4 = (f2)}$ and $\boldsymbol{f5 = (f3)}$ 72
Table 4-2: Categorization of Feature Selection Methods 73
Table 4-3: The number of features that gain the best performance in the feature selection methods 82
Table 4-4: Classification Performance for Selected Number of Features from Different Feature Selection Methods using RF Algorithm 86
Table 4-5: The Order List of the First 22 Features from All Features Selection Methods 87
Table 4-6: The Most Common Features and the Number of Methods 90
Table 4-7: Performance result of different classification methods for 14 features with common variables that appear in four methods and more 91
Table 4-8: Performance result of different classification methods for 25 features with common variables that appear in three methods and more 91
Table 5-1: Target Classes Distribution on Hull-LifeLab 94
Table 5-2: Accuracy, Specificity, Sensitivity, and PPV Results in Implementing Random Forest Classification for Several Feature Selection Methods on Different Numbers of Subsets compared with class imbalance, and balanced classes using resampling and spread subsampling for the subsets from chapter 3 100
Table 5-3: Accuracy, SPEC, SEN, and PPV Results in Implementing J48 Learning Algorithm for Several Feature Selection Methods on Different Numbers of Subsets, compared with imbalanced classes and balanced classes using resampling and spread subsampling methods for the subsets from Chapter 3 101
Table 5-4: Classification of balanced classes using Random Forest after resampling data 104
Table 5-5: Classification of balanced classes using J48 after resampling data 105
Table 5-6: The Order List of Selecting 22 Features from Balanced Data Using Several Feature Selection Methods 106
Table 5-7: Classification Outputs using RF method for training datasets on the balanced data 107
Table 6-1: Encoder Parameters 117
Table 6-2: Decoder Parameters 118
Table 6-3: Autoencoder Parameters to Build the Network 121
Table 6-4: Performance Results of the Autoencoder in one Hidden Layer 125
Table 6-5: Likelihood Rates and Their Interpretations 126
Table 6-6: Likelihood Results for the Autoencoder with One Hidden Layer 127
Table 6-7: Performance Outcomes using Autoencoder with Two Hidden Layers 129
Table 6-8: Likelihood Results for Autoencoder with two Hidden Layers 130
Table 6-9: Performance Results for Autoencoder with Three Hidden Layer 131
Table 6-10: Performance Results for Autoencoder with Balanced data 132
Table 6-11: Performance results for Autoencoder training on balance data network 133

Chapter 1. Introduction

1.1 Background

A huge volume of data has been collected over many years from patients in hospitals and clinics (Li et al, 2005). The availability of such large datasets should allow for the development of decision support systems, which will improve efficiency in healthcare. These systems are often developed by extracting knowledge through the manipulation of data and its analysis. Decision support systems can be used not only to increase the efficiency in providing health care but also to improve and design personalised care through the development of predictive models (Moore, 2015). Such systems have to produce both consistent and accurate results, and this is done through the use of a systematic methodology which aims to reduce inconsistencies while increasing both the efficiency of computation and the accuracy of prediction. This has a number of stages including cleaning of data where data corruptions are identified and removed, and the identification of the relationships in order to identify the predictive model.

Data mining is an analysis process that analyses large quantities of data to identify patterns by finding correlations between variables (Batra et al, 2013; Hand et al, 2001; Potamias & Moustakis, 2001). This is done through the use of a variety of analytical tools, such as statistical and machine learning tools. Although statistical tools are simple and have less computational complexity, these tools are unable to manipulate all kinds of data without some prior knowledge about the data distributions and its characteristics. In contrast, machine learning tools are not simple to use but require no a priori knowledge of the data (its distributions or its characteristics) and act like black boxes. Machine learning tools are used in data mining to obtain information from a large dataset by learning the system. Machine learning algorithms can be supervised or unsupervised learning. In supervised learning, the training sets with the input and outputs are presented, then the goal is to map inputs to outputs by applying some rules. In unsupervised learning,

the labels (or outputs) are not known to the algorithm, and learning is done by the exploration of the data, and groups of patterns are extracted from it.

There are many frameworks for mining data that are available e.g CRISP-DM (Chapman et al, 2000), SEMMA (Cerrito, 2006) etc. These have been modified for clinical applications, resulting in both new frameworks and workflows (Potamias & Moustakis, 2001). These frameworks need to be comprehensive and able to deal with real datasets, which often contain missing values (both random and systematic), badly distributed and unbalanced. Thus the frameworks and workflows have to deal with datasets which pose challenges in obtaining outcomes. As a result, most frameworks consist of four or more interlocking stages (see Chapter 2) and these often consist of (a) pre-processing stage (b) reduction of dimensions, (c) extraction of knowledge and (d) development. In the first stage where data is pre-processed the dataset is analysed for defects and missing values. This stage plays a crucial role in improving a framework's performance and achieving accurate prediction. Then a model is developed to find the relationships and to select important attributes which reduce the dimensions. From the relations between data, knowledge will be extracted. The last step is to develop the model after evaluation, i.e. the testing of the designed model. These steps support the framework model to classify or cluster data in an effective manner.

Clinical data often poses a number of challenges such as missing values, high dimensionality, and class imbalance. Whatever the framework that is used, the data mining strategies must overcome these challenges seamlessly and predict class labels more accurately. With a framework, there are many techniques which can be used at each stage, in order to address these challenges. The key aspect here is that as the stages are interlocked, methods are used in different stages. Missing values can be addressed by removing data points, imputing missing values through the use of means, or employing machine learning models. Removing data points that have missing values leads to loss of

information within the data. The use of means to impute missing values is often not reliable (see chapter 3), as it introduces a bias in the data. Dimensionality reduction is performed to handle high dimensionality, by reducing the number of features in order to reduce computational complexity and make the dataset more understandable. Care should be taken, that in the process of reducing the dimensions, structural relationships within the data are not destroyed; that means no important features or data points will be deleted. Feature extraction and feature selection are two methods of dimensionality reduction. Feature extraction extracts significant features by finding derived values that are informative and not redundant, while feature selection, using a relevance metric, eliminates redundant and irrelevant features and keeps relevant features.

Classification and clustering can be used to evaluate the framework and to predict the class label. Classification uses supervised learning to classify data, whereas clustering is essentially an unsupervised process. There are many classification and clustering techniques used to build a model, such as neural networks and various variations of networked learning systems. More recently, there has been an interest in Deep Machine Learning (DML) methodologies. These provide a mechanism to extract more information from the data. Often these do not need a reduction in dimensions before learning, as they extract the relevant dimensions whilst producing the correct relationships. Thus, it is feasible that such learning paradigms will reduce the number of steps within a methodology. An example of this is autoencoder (see chapter 6).

1.2 Motivation and Scope

The wealth of electronic data available has made it difficult to collect, analyse and mining data from Electronic Health Records (EHRs). This makes it challenging for clinicians to capture a patient's entire clinical history. Analysis of large datasets is required to build an effective diagnosis strategy (Tripoliti et al, 2017). There are various possible solutions to

increase healthcare quality by the development of electronic medical records (EMR), computer-based physician order entry (CPOE), and clinical decision support system (CDSS) (Rajiv Wadhwa et al, 2008). Various frameworks exist to analyse and design models for disease management and diagnosis (Cleland, 1999).

Chronic heart failure is the most important issue in the medical field; it is the leading cause of death all over the world in the past decade (Shouman et al, 2012). The prevalence of death caused by heart disease is reported by the American Heart Association. The report mentions that in 190 countries, 17.3 million patients per year die of heart disease, by 2030 the number will increase to 23.6 million (Sharma et al, 2016).

The data mining methods are discovering relationships using mining tools to obtain a new knowledge from the quantity of data. Data mining assists clinicians in both decision support systems and in creating a framework for medical diagnosis, for example in detecting a particular disease through a small number of features. Thus healthcare can be fitted to the specific needs of patients with the following set of aims:

- To develop predictive models that will help in the planning of care.
- Improve treatments by discovering new knowledge and useful information.

The effective use of data mining methods for rapid clinical decision making requires the availability of high-quality clinical data (Lu & Su, 2010). In this thesis, the data is obtained from a cardiology clinic at the Hull Royal Infirmary Hospital (see next section 1.2.1). This data has a number of issues which create a problem for developing prediction algorithms and applicable classification. Therefore the motivation for the thesis is to develop a methodology for mining clinical data using an autoencoder model; also to investigate the clinical data challenges to improve the performance of classification algorithms.

1.2.1 Research dataset

Hull LifeLab is a large information-rich clinical dataset consisting of patients with possible heart failure referred to a cardiology outpatient clinic. The dataset is from a clinic which served a community of about 550,000 people in Kingston-Upon-Hull and East Riding of Yorkshire between 2000 and 2012. All referred patients were invited to participate in the study and 98% gave informed consent for their information to be retained and used for research purposes (Moore, 2015). Consenting patients received a comprehensive clinical assessment and those found to have heart failure were followed up with further outpatient assessments at regular intervals (typically every 4-6 months). The dataset is composed of 463 continuous and categorical variables and 2,032 patient records including quality of life. This thesis considers 61 important variables associated with blood chemistry. The reason for considering blood chemistry is that the other variables are either well understood or are essentially categorical, whose values are subjective and dependent on the interpretation of the patients' record by either the nurse or a clinician (Cleland et al, 2016). These additional variables are also prone to having missing data greater than 20%.

The reasons that there are missing values in the dataset are missing data and outliers during data recording or entry due to incorrect measures and mixed variable types (Weitschek et al, 2013). In addition, in cases where clinical data are collected as part of a clinical trial, the medical report pro forma allows certain variables to be left blank. This is usually due to the ailment being treated or perhaps the patient may not wish to disclose certain information, such as whether he or she is a smoker (Zhang et al, 2012).

Of the dataset, 1944 patient records were analysed, as the remainder had variables where more than 20% of values were missing (Acuna & Rodriguez, 2004). These 1944 patient records were collected at four different time points: 3, 6, 12 and 18 months. After 18 months, 1459 patients had no record of death and had attended an outpatient clinic and

were therefore classed as alive. The remainder were classed as dead (485) as there was a record of death present for each one of those patients. The classes will be referred to as the alive and dead classes in the following chapters.

The Hull LifeLab is a confidential dataset, which is being constantly updated. More details about the variables and their characteristics can be found in appendices I and II.

1.2.2 Clinical data challenges

As discussed earlier, real datasets have some challenges that need to be addressed before analysis. The challenges are missing values, high dimensionality, and class imbalance.

- Missing values

Missing values come from the different systematic ways that the clinical data are collected, and the transfer of data between systems; they constitute an important issue in medical data mining (Rahman & Davis, 2013). Missing values are categorised into three types: (a) missing completely at random (MCAR) (b) missing at random (MAR), and (c) missing not at random (MNAR); more details are given in Chapter 2.

Several strategies can be used to impute missing values: (a) removing the record, (b) mean/median, (c) imputation. Removing records that have missing values causes loss of the structure of the dataset and reduces the number of records, which may in consequence be insufficient to analyse data accurately. Finding the mean and median is a simple method, but takes no account of the consistency and the structure of the data. Imputation methods are based on data mining techniques, for example Most Common Imputation (MCI), and Concept Most Common Imputation (CMCI), K-nearest neighbour imputation (KNNI) (Batista & Monard, 2002), SVMI (Mallinson & Gammerman, 2003), Expectation Maximization imputation (EMI) (Barzi & Woodward, 2004), and K-mean imputation (Li et al, 2004).

- High dimensionality

Another challenge of clinical data is its high dimensionality. Clinical datasets often present too many features. Some features are redundant and irrelevant, whereas others are relevant. Dimensionality reduction will reduce computation complexity, and affect the achievement of data mining tools. High dimensionality must be addressed in the pre-processing phase of knowledge discovery so that features of low relevance are not sent to the learning process. Dimensions can be reduced by selecting features or by extracting features. Feature selection is a mechanism that eliminates irrelevant and redundant features. Feature extraction is a mechanism that reduces the number of features by identifying informative features, for example, principal component analysis (PCA) (Li et al, 2006), and independent component analysis (ICA) (Hoyer & Hyvärinen, 2000). Feature selection is categorized into three types: (a) ranking methods, by sorting the features adopted in the algorithm used, for example ReliefF (Guyon & Elisseeff, 2003), or chi-square (Zheng et al, 2004), (b) the Wrapper method, by evaluation of the feature during the learning algorithm (Cohen & Hirsh, 1994), and (c) the embedded method, by employing a classification algorithm (Zhang et al, 2015).

- **Class imbalance**

Clinical datasets often suffer from class imbalance. In a binary class, the class imbalance is that the number of data points of one class is greater than the other. The majority class is the class with the larger number of samples, whereas the minority class is the class with fewer samples, see Table 1-1 for the thesis dataset, which shows a class imbalance. The learning algorithm maps inputs to desired output; if there is an imbalance between samples for each class; then the minority class will get less chance to be trained. Thus the result of performance will be affected (Menardi & Torelli, 2014). Resampling is a simple strategy to deal with class imbalance, by increasing the minority sample or decreasing the majority sample, known as over-sampling and under-sampling, respectively (Cao et al, 2016).

Table 1-1: The Hull-Life-Lab Dataset Distribution of the Classes

No. of features	61	
No. of samples	1944	
Target output	Mortality	
Class	Alive	Dead
Frequency	1459	485

- **Classification**

The last step of knowledge discovery is classification, the result of which controls the decision support. Methods of classification of clinical data are affected by the above challenges. The models designed to classify data suppose that the data is complete has a low number of features and is balanced. The framework for mining real-life data should anticipate data with difficulties, by pre-processing of this data to be acceptable to the classification method used. Numerous classification methods can be used to evaluate the performance of the learning process, for example, random forest (RF), J48/C4.5, neural network and REPTree. A classification method can be evaluated and the results compared by criteria measures using a confusion matrix.

1.3 Research Questions

There is a wide range of research issues which are addressed in this thesis. These can be summarised at a high level as the following set of overarching questions:

1- How can pre-processing improve the outcomes of clinical data mining?
2- What combinations of techniques are useful for imputing missing value, selecting features, and classification?
3- Can we employ an efficient method which reduces the number of data mining steps?

The key objective of the thesis is to find out a suitable approach that can be used to improve the prediction of clinical data classification; so that this can be used in

decision support systems. The dataset used in this work is the Heart failure dataset provided by Hull York Medical School (HYMS). This data contains a set of issues such as missing values, high dimensionality, class imbalance and non-normal distribution. Data mining tools face challenges while working with data having these problems. This issue has not gone unnoticed, researchers have tried to solve these issues in this kind of dataset. Even though many researchers have worked on heart failure data, very few have reported about the dataset problems, and they did not solve all the problems of this data. The quality of data is the main criterion for designing an efficient decision support system. In this research, the clinical data issues are taken into consideration and found. As well, the methods used have been justified to by addressing all limitations of these methods and discussing how to improve these methods to enhance the outcomes of the prediction models. We used autoencoder to extract features that are most relevant to build a framework for classify data. This model can be used to decrease the data mining steps without loss of accuracy. The autoencoder method as a deep learning model has not previously been used for clinical datasets.

1.4 Aim and Objectives

The problem to be solved in this thesis is a clinical dataset, where we can employ data mining methodology. The aim is to investigate the issues in the clinical dataset and to implement machine learning techniques to solve these problems. Specifically, the main goal of this thesis is to define the data mining process and reduce the steps of the data mining methodologies by employing an autoencoder model, along with investigating the challenges of clinical data described in the previous section, including missing values, high dimensionality, and class imbalance. Three major steps are needed to solve the challenges of the clinical data, which form the key objectives of the thesis:

1- Investigating the missing values problem, by studying this issue and imputing the missing values using different imputation techniques.
2- Investigating feature selection methods to reduce high dimensionality by using data mining tools. Thus involves
    a. Finding the significant features from the heart failure dataset, which are the relevant features.
    b. Analysing the effect of class imbalance on classification performance and the output feature selection methods.
3- Developing and evaluating an autoencoder model to compress the data mining methodology by extracting features and classifying the heart failure dataset.

The main contribution of this thesis is improving the prediction performance for the heart failure dataset classification. Thus, we increase the accuracy of classification of data that has missing data 78% to more than 85% for the imputed dataset. Similarly, the accuracy of prediction has been increased to more than 88% using a small number of features rather than using all features of the dataset. Although the most techniques used are not novel and have been used before; our study analysis has been explaining the reasons for algorithms techniques that work better than others. As well as, interlinked between methods used for imputations and the methods used for feature selection and class imbalanced.

1.5 Thesis Structure

In Chapter 2 data mining methodologies are discussed. These methodologies set a well-structured framework in order to answer the questions posed above. The first stage is pre-processing, which can solve and impute missing values. Then, dimensionality is reduced by eliminating redundant and irrelevant features using feature selection or feature

extracting. Next, classification methods used to evaluate the model. This methodology has high computational complexity and would be compressed to reduce the complexity.

Having fixed the methodology and workflow, in Chapter 3, the various imputation methods are discussed. Missing values pose a crucial issue for using data mining algorithms and affect their performance, as imputation is the first step of data mining. Missing values affect the feature selection and extraction, since features with missing values may be not nominated because there is not enough information.

In later chapters, issues around selection of features (dimension reduction) (Chapter 4), Class Imbalance (Chapter 5) are discussed in detail. The focus in these chapters is not only on the overall performance but also issues around the complexity of the methods. This then leads to Chapter 6 where, based on the results from the previous chapters, an autoencoder is designed. The results are then compared with the other methodologies in terms of performance as well as computational complexity.

Chapter 2. Literature Review on Clinical Data Mining

2.1 Introduction

Data mining plays a crucial role nowadays in analysing and testing real-life datasets (Kausar et al, 2016; Sharma et al, 2016). Clinical data is an example of interesting real data due to the importance of this data to improve decision support systems in the medical field. However, to achieve the aim of implementing data mining algorithms, the data have to be prepared and edited to overcome some shortcomings such as missing data, high dimensionality, and class imbalance.

Several methodologies are used in data mining such as CRISP (Europe, 2017) and SEMMA (SAS-Institute, 2017), which are stages used to develop a framework for mining a large data volume. The methodologies start by understanding and pre-processing the dataset before manipulating the data to obtain knowledge. Pre-processing data starts with cleaning the dataset by deleting the data points that have too many missing values. Then missing values are imputed by implementing a suitable technique. One of the most appropriate techniques is imputation by calculating missing values through machine learning tools. The imputed dataset will be ready to extract latent information from raw data by dimensionality reduction. Features can be reduced by using a feature selection technique that selects the most relevant features and eliminates other features, which will reduce the complexity and identify the significant features. Classification is one of the most important processes in data mining, which helps to build decision support systems and prediction systems. Although these methodologies have many steps, we will follow the CRISP methodology in the thesis, and then reduce these steps using a neural network model.

An autoencoder is a neural network in which the outputs are almost equal to the inputs, where there is more than one hidden layer between the input and output layers. It is a deep architecture for transfer learning and other tasks. In addition, autoencoder can learn more

complicated relations between visible and hidden units. This model can be used to compress the data mining stages, by extracting features and classifying data in one model. This chapter discusses data mining methodologies, and the stages of obtaining the knowledge that helps in decision making. Also it explores data mining tools that can be implemented on clinical data to impute missing values and reduce dimensionality, and the classification techniques used to classify the dataset. Then the chapter will demonstrate neural network models, and how such a model can be employed on a clinical dataset.

2.2 Knowledge Discovery and Data Mining

Knowledge Discovery in Database (KDD) is a process for obtaining knowledge from a large dataset, by following an arranged sequence of steps (Poultney et al, 2006). The derived knowledge must represent in some respect the interpreted data and their relations (Ma et al, 2015). KDD describes the mechanism that can be applied to the result of data mining (Lobur et al, 2011). Thus, data mining is used to generate effective information acquisition by using different kinds of algorithms implemented on datasets (Hinton et al, 2006; Lobur et al, 2011). The knowledge discovery process includes data warehousing, cleaning, pre-processing and transformation; whereas data mining includes model selection, evaluation, and interpretation; for example, data mining classifies data and identifies patterns. Knowledge discovery refers to the comprehensive process of discovering useful knowledge, while data mining is the use of tools in particular steps in this process to extract information (Lobur et al, 2008). To define knowledge from pattern extraction, the KDD process steps involve understanding the domain and data, data preparation and association (Lavrač, 1999).

An easy approach would be to collect the data and run clustering, classification, model identification or evaluation algorithms (Poolsawad et al, 2014b). However, a dataset is

not easy to use before pre-processing steps, because data often come with problems like a large number of variables, missing values, and class imbalance. Thus, extracting information will not give satisfactory results. Clinical data is an example of the data that have problems that affect the use of data mining tools and their output (Balakrishnan et al, 2008). Thus, the challenges faced in mining the clinical data require a pre-processing stage for data mining algorithms to addres the data issues.

The first step in knowledge discovery is the selection of relevant prior knowledge to specify the application domain and goals of the application, as illustrated in figure 2-1. Next, the target data are selected and filtered by data cleaning. After that is a step called data reduction, which entails finding useful features using feature selection or extraction. The next step is to choose the mining algorithms such as decision trees, Support Vector Machine (SVM), Neural Network, etc. The mining algorithm is implemented to search for interesting patterns. Finally, the result is analysed by transformation, visualisation, or removing redundancy.

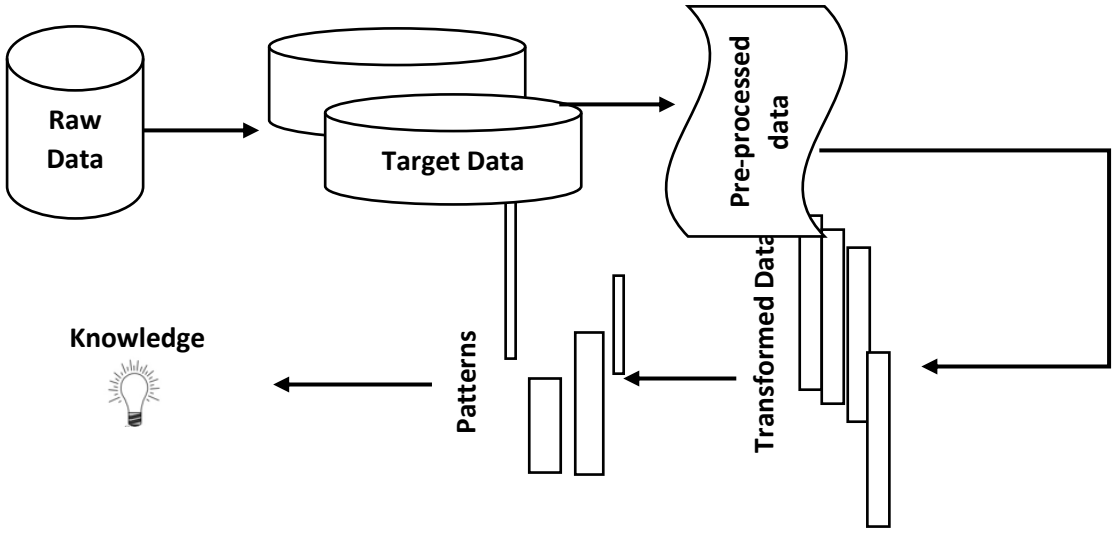

Figure 2-1 KDD, key steps in an iterative and interactive process

There are various approaches in data mining to generate information from the dataset, including classification, clustering, and regression (Bellazzi & Zupan, 2008). Classification is used when the output is presented with the input data, known as supervised learning. Clustering is a learning process, where there is no output data present, and the data are organised into similar groups; this is known as unsupervised learning. Regression is a supervised learning, but the outputs are not discrete, but continuous. Description and prediction are the two tasks of data mining (Bellazzi & Zupan, 2008). Description aims to find human-interpretable patterns and associations, after considering the data as a whole, whilst prediction seeks to predict some outcome of interest. Currently, a major aim of data mining is to discover association among variables that may be useful in future decision support (Mullins et al, 2006).

2.2.1 Data mining methodologies.

The knowledge discovery process involves interactive and iterative steps with many decisions made by the user (Fayyad et al, 1996), as shown in figure 2-1. Data mining methodologies can be categorized into the following:

2.2.1.1 **CRISP-DM**

CRoss Industry Standard Process for Data Mining (CRISP-DM) is widely used as a data mining model. As we see in figure 2-2, the model consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Development (Chapman et al, 2000). CRISP is a process defining the model in data mining to provide a framework for carrying out projects, independent of the industry sector and the technology used (Wirth & Hipp, 2000). The model aims to make large data mining projects less costly, more reliable, more repeatable, more manageable, and faster.

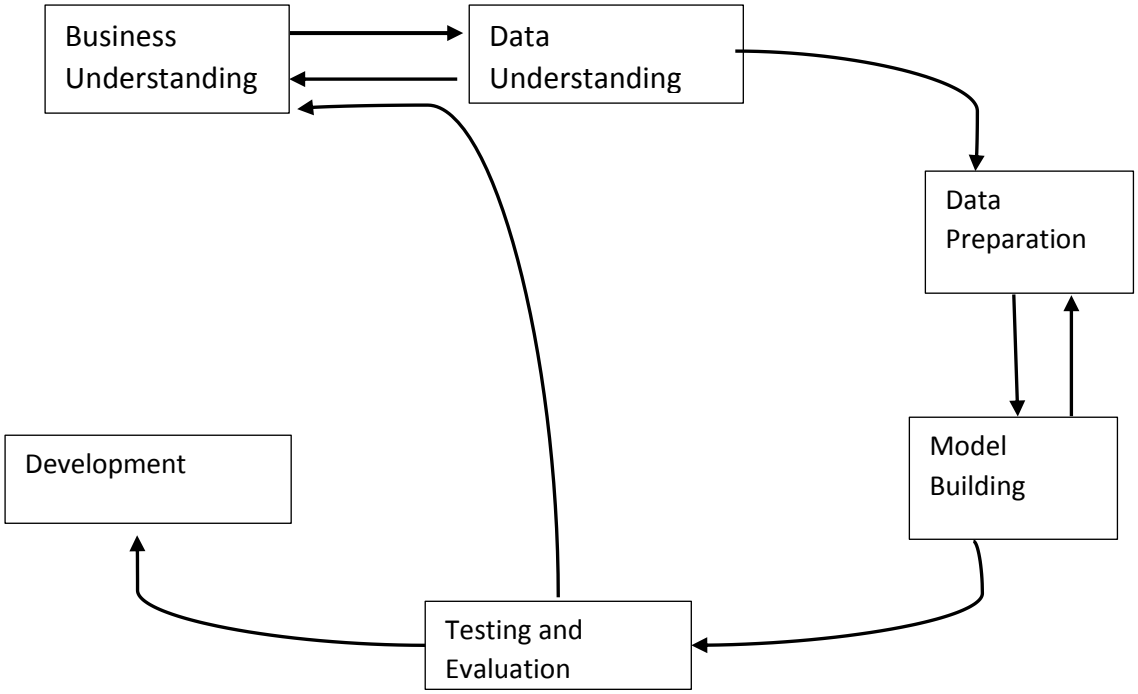

Figure 2-2 CRISP-DM Methodology

CRISP-DM consists of four levels: phases, generic tasks, specialised tasks, and process instances (Catley et al, 2009). At the top level, there are a small number of phases, each of which consists of several tasks in the second level. The second level is called generic because it is intended to be general enough to cover all possible data mining situations. To cover both the whole process of data mining and all possible applications, the generic tasks are designed to be as complete and stable as possible. The model's stability means it may be valid for yet unforeseen developments, like new modelling techniques. The specialized tasks in the third level describe the actions in the generic tasks to be carried out in specific situations. For example, at the second level, there is a generic task called build model. Build a response model in the third level is a task which contains specific activities for the problem and the data mining tools (Catley et al, 2009).

A specific sequence of discrete steps represents the description of phases and tasks. In practice, many of the tasks can be performed in a different order; for example, it may be necessary to backtrack to the previous task or repeat certain actions. Through the data

mining process, CRISP-DM will not possess all the possible routes because it would require an overly complex model with low expected benefits. The fourth level, the process instance, is where the actions, decisions, and results of an actual data mining engagement are recorded. The process instances are organised according to the tasks defined at the higher levels , but they represent what actually happened in a particular engagement, rather than what happens in general (Rahman & Davis, 2013). CRISP is widely used to solve clinical data mining problems.

2.2.1.2 **SEMMA**

Derived from the statistical analysis software institute (Matignon, 2007), SEMMA consists of five steps, as shown in figure 2-3.

a) Sample: the process starts with data sampling, i.e. selecting the data set for modelling.
b) Explore: discovering and anticipating the relationships between variables.
c) Modify: the methods to select, create and transform variables, in preparation for data modelling.
d) Model: depending on prepared variables, applying various modelling (Data Mining) techniques to create models that can provide the desired outcome.
e) Assess: To show the reliability and usefulness of the model, it should be evaluated.

In the sample step, the dataset is taken and portioned into training, validation, and test sets of samples. The dataset is visualized statistically and graphically in the explore step, to explore distributions, correlations, etc. Then the data is transformed in the modify step, to change the data for the suitable model, if possible, or deal with missing values. Next, comes the fitting of the dataset into the machine learning technique, for example when using neural network or decision trees. Eventually, in the assessment step, the data is partitioned into alternative sets to validate the model and to estimate the data mining process.

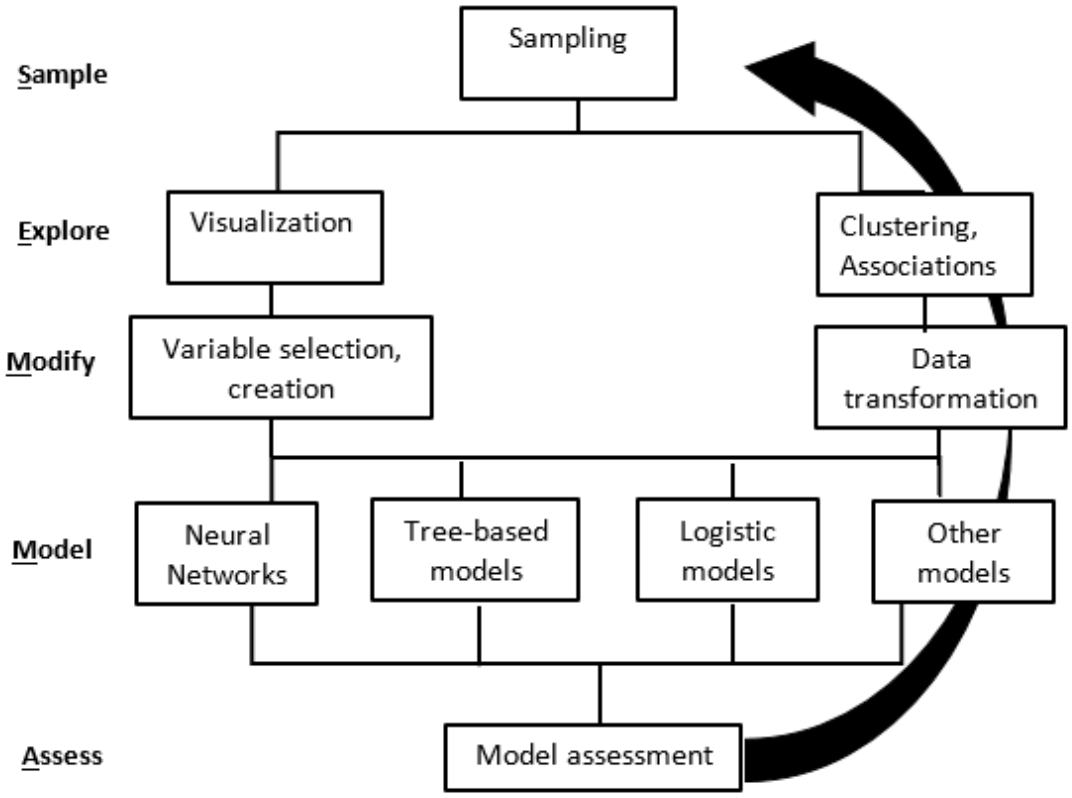

Figure 2-3 SEMMA Methodology (Original from SAS Institute)

The SEMMA process provides an easy to understand process, allowing adequate development and maintenance of data mining projects. In contrast to the CRISP-DM process, SEMMA also allows the user to return to previous steps in the process and focus mostly on the application to exploratory statistical and visualization-based data mining techniques (Bellazzi & Zupan, 2008).

2.2.2 Medical data mining

The increase of use of information and computer technology have led to an increase in the volume and complexity of data (Lobur et al, 2011). Patient records used in management and other medical information are stored electronically as medical data or hospital information systems; European Institute for Electronic Health Records (EuroRec) is an example of patients' records. This search has an interest in using the clinical data to develop frameworks and models which can improve healthcare. The field of data mining and management of clinical datasets has attracted interest in several

disciplines. Discovery of knowledge from the raw data is the goal of data mining techniques, and medical data offers a promise to identify clinical data patterns (Friedlin et al, 2011; Tsumoto, 2000). The SEMMA and CRISP-DM frameworks have been used for such clinical data prediction for clustering and classification (Jilani et al, 2016).

Clinical data mining entails extracting implicit and potential information from raw clinical data to benefit the decision support system and hence to improve healthcare systems (Esfandiari et al, 2014). Healthcare delivery benefits from the technologies used in the medical field (Masci et al, 2011). Moreover, the development of data mining for health monitoring systems has forced the provision of proactive decisions (Sow et al, 2013). Nowadays, there is an active interest in the advancement of diagnosis systems, which use clinical data to assist in recognition of diseases and tracking patients' conditions (Lin & Haug, 2008).

Data mining has been used in the medical domain for diagnosis and treatment analysis; it brings a set of tools that can be applied to clinical data to discover underlying patterns and support healthcare professionals (Lu et al, 2016). After pre-processing data, the data is further prepared for sequential patterns mining. The final step for mining data is evaluation; this is done by using evaluation approaches such as classification, clustering, and regression. Lokeswari & Jacob (Lokeswari & Jacob, 2015) propose a model that would enable mining meaningful medical information to be mined from a community dataset; they implement various parallel classification algorithms; decision trees, K-NN, and Naïve Bayesian classifiers. The model works by classifying data using the three techniques then the outputs are given as input to a "Bagging" algorithm which can improve the accuracy of the parallel classifier. The accuracy is evaluated using specificity and sensitivity metrics. Sharma and his colleagues (Sharma et al, 2016) provide a survey on heart and cancer disease. The study reviews the existing research to find out significant knowledge in this field and summaries of different approaches used in diagnosing disease,

in addition to discussing the tools available for processing and classification of data. They conclude that the selection of data mining approaches is not the same for all; it truly depends on the dataset type.

The Seattle Heart failure model (SHFM) is an example of a model for diagnosing heart failure (Levy et al, 2006). The model is designed to predict survival years in heart failure patients using Prospective Randomized Amlodipine Survival Evaluation (PRAISE1) (Pfeffer & Skali, 2013). It was developed primarily from clinical trial databases and the benefit of interventions extrapolated from published data. The SHFM was derived by retrospectively investigating predictors of survival among 1,125 patients in PRAISE1 (NYHA 3B-4, EF<30%, ACEI, diuretics, 403 deaths). This program uses data from six research studies on heart failure to calculate an individual patient's probability of survival with heart failure and the potential benefit of various medical and surgical treatments. It is intended for use by medical professionals who are trained in the treatment of heart conditions, to help inform decisions on treatment, and counsel the patient. A stepwise Cox proportional hazard model is used to develop a multivariate risk model, which identified age, gender, ischemic ethology, NYHA, ejection fraction, systolic blood pressure, K-sparing diuretic use, statin use, allopurinol use, haemoglobin, % lymphocyte count, uric acid, sodium, cholesterol, and diuretic dose/kg as significant predictors of survival. SHFM provides an accurate estimate of 1-, 2-, and 3-year survival with the use of easily obtained clinical, pharmacological, device, and laboratory characteristics (Washington, 2012). The researchers conclude that amlodipine would reduce the rate of death from all causes in heart failure patients with the same symptoms. The model is a well-validated prediction model of all-cause mortality in patients with heart failure, but its relationship with generic health status measures has not been evaluated (Li et al, 2013). It may also be used to assess relative risk and changes over time, but when assessing the absolute percentage of event-free survival, the overestimation of event-free

survival should be accounted for (Sartipy et al, 20014). The SHFM requires simple variables necessary to calculate risk and incorporates heart failure medications and devices. Addition of peak oxygen consumption to the model in 1,240 ambulatory heart failure patients increased the ROC from 0.758 to 0.766.22 When annual mortality is >20% the risk of pump failure death exceeds the risk of sudden death.

Patients at Risk of Re-hospitalisation (PARR-30) is another application for identifying inpatients at risk of re-admission within 30 days of discharge. The dataset is collected from hospitals in the UK within 30 days of discharge using information that can either be obtained from hospital information systems or from the patients. The results show that positive predictive value (the percentage of inpatients identified as high risk who were subsequently readmitted within 30 days) was 59.2%; also the area under the curve was 0.70%. The existing PARR tools that were first developed in 2005 are designed to identify patients who may be at risk of re-admission within a year, but they use data that may not be available while the patient is still in hospital (Nuffieldtrust, 2012). Predictive models need to be updated from time to time to reflect changes in clinical practice, and in epidemiology, demographics and clinical coding.

Another study on chronic heart failure is that of Shelton et al (Shelton et al, 2010), who investigate the effect of atrial fibrillation and its relationship with heart failure. A survey used a heart failure dataset to investigate current data mining techniques that discover knowledge from data, to compare prediction performance and decision tree outcomes (Soni et al, 2011). Classification of heart failure patients into four classes has been proposed in Saqlain et al (Saqlain et al, 2016), using data mining tools.

Using the PARR1 algorithm requires data on admitting diagnoses; however, obtaining diagnostic information on patients prior to discharge can be problematic with some hospitals and is likely to rely on the use of an admitting diagnosis field which many

consider less reliable than discharge data entered by medical records staff (Billings et al, 2006).

2.2.3 Data mining techniques

As the volume of information becomes larger, more extensive data mining techniques have appeared (IBM, 2012). The techniques could be used for classification, clustering, association, prediction, and pattern recognition, as shown in figure 2-4.

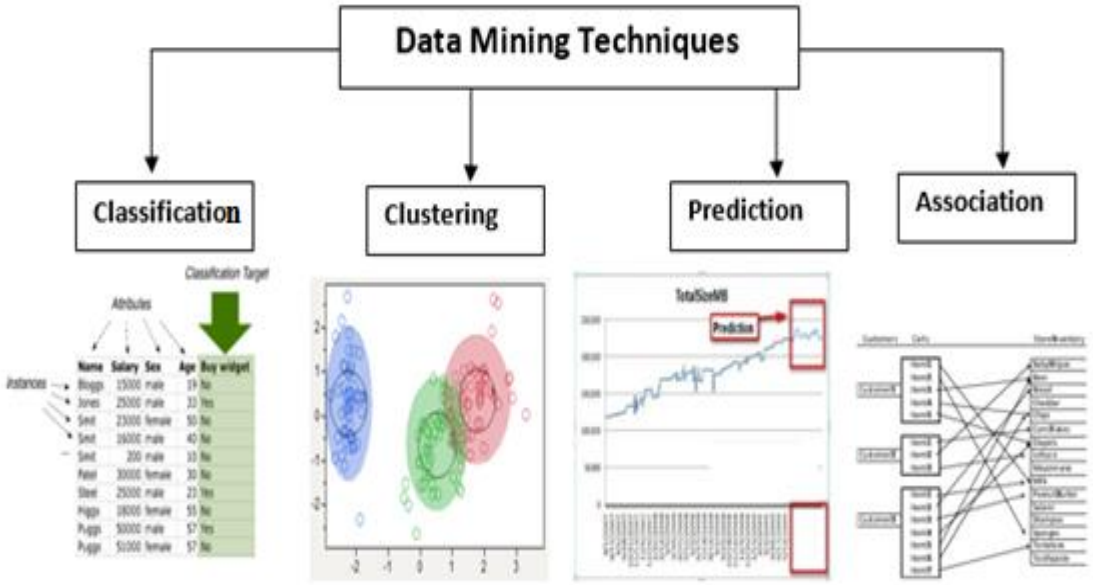

Figure 2-4 Types of Data Mining Techniques

Classification is a process where the learning function in the training data has to map the input instances into one of the predefined classes (Olson & Delen, 2008). The class is a specialised target represented by the value of a particular feature in the dataset (Aggarwal & Zhai, 2012). For example in clinical data, each record may be tagged by a particular class label that represents the mortality of the patients; thus the goal is to predict the patient class label.

Clustering is an automatic technique that takes the dataset as input and groups the data points into clusters. This is unsupervised learning that works without a class label (Olson

& Delen, 2008). It is an optimisation problem, where the variables represent the membership of data points, and the objective function maximises mathematical quantification of group similarity (Aggarwal & Zhai, 2012). An example of clustering is to determine patients that are similar to one another in the context of a variety of variables.

The association technique is used to predict patterns depending on the relationship between particular items in a data and other items in the same transaction. For example, the data matrix corresponds to an item in each column and association finds the relation between variables and the class or between the variables themselves.

In prediction, the analysis discovers the relationship between the independent and dependent variables. For example, in linear distributions, if the line is distributed in a stable trend, the next target value could be predicted for the requested data. Pattern recognition techniques search for similar patterns in the data transaction over a dataset; the patterns can help to find the relationships among data. They are generally categorised corresponding to the learning technique used to generate the output value.

2.3 Missing Values and Imputation Techniques

Missing values come from the different systematic ways that the clinical data are collected, and the transfer of data between systems; they constitute an important issue in medical data mining (Rahman & Davis, 2013). Missing values are categorised into three types, (a) missing completely at random (MCAR) (b) missing at random (MAR), and (c) missing not at random (MNAR). MCAR refers to a variable that has missing values that do not affect the data in other variables, this missing data is independent of other variables and of itself, so the value of a record does not depend on any other variable, even itself. MAR is present when the variable is independent of other variables but not itself, for example when a variable is left empty because it is affected by another

variable value. MNAR refers to a variable that is related to other variables, so many variables are not filled because such a variable is left blank.

Missing values or missing data occur when no information is available for some data points in the database. Missing values are often present in real data (Engels & Diehr, 2003) since as Carmona et al (Carmona et al, 2012) note, the collected data in real-life applications are not perfect. The data could be missing for a variety of reasons, like manual data entry procedures, and equipment errors. The existence of missing values in data mining produces several negative effects in the knowledge extraction process, such as (a) loss of efficiency, (b) difficulty in managing and analysing data, and (c) bias resulting from differences between missing and complete data (Carmona et al, 2012). Data mining algorithms are designed for quality data (Blake & Mangiameli, 2011). A number of techniques for data analysis have been developed in recent years. However, most of them do not deal sufficiently with missing values (Lobato et al, 2015).

Methods used to impute missing values are ignoring, single imputation, and multiple imputations. If the occurrence of the missing data is completely at random, we can simply remove it because it does not affect other data (Jing, 2012). Case deletion or ignoring missing data means eliminating all records from the dataset that have at least one missing value (Carmona et al, 2012). Removing missing data will lose some data which could be important for the learning process. Also, the distribution of data will change, while imputation can restore the maximum amount of information without losing any data (Jing, 2012).

2.3.1 Single imputation:

A variety of single imputation methods exist, they include:

- Concept Most Common (CMC): in this method, the missing value is imputed in respect of class value by the most common value in the attribute for symbolic

attributes; missing values of numerical attributes are imputed by the most common value. The concept refers to a set of all cases with the same outcome (Grzymala-Busse et al, 2005).

- Mean/Mode: missing values are imputed by computing the mean of the numeric data, or the mode value for discrete data. This method assumes that the data are MCAR, which is not true in most biological data (Rahman et al, 2014). The mean cannot be computed for dependent variables and will be affected by other values.
- K-Nearest Neighbour Imputation (KNNI): this algorithm is instance-based. Every time a missing value is found in a current instance, by calculating the distances for each data point, KNN can impute the missing value by the nearest data point value. KNNI can predict both quantitative (the mean among the KNN) and qualitative attributes (the most repeated value among the KNN). The main deficiency of the KNNI approach is that the algorithm searches through all the data and this is repeated each time to find nearest distances, which is very slow and is a limitation for large datasets (Batista & Monard, 2003). This limitation is critical for KDD since we have a massive dataset.
- Model-based methods: a model is designed to learn from complete data, and then the missing values are imputed based on the prior knowledge. Based on using classification learning for nominal attributes, and regression for continuing attribute, the model can then impute missing values for the rest of data (Rahman et al, 2014).
- Expectation Maximisation (EM): a statistical method based on maximum likelihood. EM consists of an E-step and an M-step The E-step calculates the conditional expectation of the parameter on missing data. The M-step estimates the parameters by maximising the complete data likelihood (Jing, 2012). EM is commonly used in data clustering and machine learning. According to Little & Rubin (Little & Rubin,

1989), in incomplete data problems, it is a general iterative algorithm for maximum likelihood estimation (Yuan, 2010).

The treatment of missing data is a very widespread broad statistical problem and one should consider that there is no universal imputation method performing best in every situation (Schmitt et al, 2015). We will use most of the single imputation methods investigate the best methods. The methods used will be differentiated by applying different scenario.

2.3.2 Multiple imputations MI:

Missing values can be imputed by finding more than one imputed value by different methods, then filling the missing value with the least estimation error (Ghoneim et al, 2011; Su et al, 2011b). MI appears to be the most attractive method for handling missing data (Allison, 2000), The basic idea is given by Rubin (Rubin, 1987):

- Use an appropriate model to impute missing values that incorporate random variation.
- Do this M times (usually 3-5 times), producing M complete sets.
    - Perform the desired analysis on each dataset using standard complete-data methods.
    - Average the values of the parameter estimate across the M samples to produce a single-point estimate.
    - Calculate the standard errors, by averaging the squared standard errors of M-estimates, then calculating the variance of the M parameter estimates across samples, and combining the two quantities.
    - Develop a Multivariate normal (MVN) model, which considers that the variables are continuous and normally distributed.

Multiple imputations make the model more complex, so we will not use this technique in this research.

2.3.3 Machine learning (ML)

It is possible to impute missing values based on machine learning algorithms (Farhangfar et al, 2008). Unlike statistical methods, ML algorithms generate a data model from incomplete data, then the model is used to perform classification that imputes the missing data (Farhangfar et al, 2007). Decision trees, probabilistic, and rule-based methods are some examples of ML algorithms used to impute missing data. Evolution of the imputation method simply applies a classification technique for the imputed data, then evaluates the method outcomes based on accuracy, sensitivity, and specificity.

2.4 High Dimensionality

The task of data mining is to extract information and knowledge for a large dataset. Data pre-processing is required to prepare the data for data mining and machine learning to increase the predictive accuracy (Selvakuberan et al, 2011). Often high dimensionality data cause problems for the learning algorithms in terms of efficiency and effectiveness. Thus it is important in mining medical data to manipulate the high dimensionality and data preparation, then to use data for other applications (Cios & Moore, 2002). Usually, clinical datasets are highly dimensional in nature, as large quantities of information about patients and their clinical history are accumulated (Balakrishnan et al, 2008). Dimensionality reduction is used for reducing the number of features to those that are more relevant for further analysis (Joshi & Machchhar, 2014). Therefore the reduction will reduce the dataset size, while maintaining much of the variance of the larger dataset without dropping the critical features. This also decreases computational complexity and makes it easier to use real-life datasets. A variety of models have been developed to obtain optimal extraction of patterns, including artificial intelligence and machine learning models (Abraham et al, 2007).

The goals of dimensionality reduction algorithms are (Gonen, 2013):

1) Improve prediction performance by removing redundant and inherent noise.

2) Explore data by getting low dimensional visualisations.

3) Reduce computational complexity.

With the input variables $\{x_1, \ldots x_n\} = X \in \chi$, and the response variable y, then the objective of dimensionality reduction is to find a subset $\{x_1, \ldots x_m\} = \hat{X} \subseteq X$ where $n > m$, with minimal dimensions d that satisfy a particular criterion (Fewzee & Karray, 2012). Two different approaches for dimensionality reduction are available: supervised and unsupervised. The supervised approach is when some discriminant analysis uses class information, and in unsupervised approaches, data samples are not accompanied by labels (Joshi & Machchhar, 2014). Examples of supervised approach methods are linear discriminant analysis (LDA), and autoencoder neural network. Unsupervised methods include principal component analysis (PCA), independent component analysis (ICA), single value decomposition (SVD), and kernel principal component analysis (KPCA).

2.4.1 Feature selection

Feature selection is important in clinical data because it determines the features that can diagnose the disease. Thus, a minimum number of features could determine patients' care activity (Gürbüz & Kılıç, 2011). A raw clinical dataset could provide useful knowledge for effective decision making, and data mining searches for the relationships and patterns of this kind of data. These datasets usually have a high dimensionality of variables, with irrelevant and redundant features. Feature selection is used to handle the above issues, and also can reduce the amount of data needed for learning, increasing the constructed models' precision, and improving algorithm predictive accuracy (Balakrishnan et al, 2008). By reducing the number of clinical data features, this may reduce the number of measures made or enhance classification accuracy, hence reducing false negatives (Balakrishnan et al, 2008). Much research in data mining has focused on improving classification accuracy by applying feature selection methods (Abraham et al, 2007).

Feature selection could be used to speed up the process of learning, since using limited features will simplify the representation of patterns and decrease the complexity of the classifiers. The performance of a classifier depends on the relationship between the sample size, the number of features, and the classifier complexity. To obtain good classification accuracy, the number of training samples must increase as the number of features increases (Guyon & Elisseeff, 2003). The prediction accuracy of the classifier in data mining is improved by applying feature selection techniques, since not all features used in descriptive data are important for all problems.

Feature selection methods:

1. **Ranking methods,** these are simple methods where only the best features are selected by measuring the relevance to the concept variable. They are categorized into
    - Univariate techniques: individual features are evaluated independently. The best m features that obtain the best accuracy will be selected. These methods are computationally simple, and sometimes fail if there are dependencies between features. Examples of univariate techniques are chi-square, information gain, and correlation. These methods use a calculation measure between each feature and the concept variable, then select the best features that can improve the classification performance.
    - Multivariate techniques take into account the relations between features, using different approaches to measure the correlation between features and the class. An example is ReliefF.
    - Subset features selection; these methods do not rank features depending on their measurements, but select the subset that obtains a high improvement, for example, correlation-based feature selection (Cfs) and consistency. These methods operate either by increasing or decreasing the number of features,

depending on the starting point. They may start with the empty set then add features one after the other to the evaluated set, or start with the full set and delete features from the set until the best subset is attained. The approaches used are backward and forward, sequential forward selection (SFS) and sequential backwards selection (SBS) (Uzer et al, 2013).

2. **Wrapper method**: in this method the evaluation of features uses a training algorithm to find the relevant subset of features to attained the best subset (Cohen & Hirsh, 1994). Thus the algorithm acts as a wrapper around the classifier, which selects subsets that best classify the dataset. Then the cross-validation will evaluate the learning scheme. This method has high computational complexity because it is necessary to train the classifier for each selected subset (Zhang et al, 2015). The heuristic search has space size 2^n for n features.
3. **Embedded methods:** in this technique, the feature selection is performed in the learning process. Therefore, the method employs a classification algorithm, then it will search for variables by adopting the structure of the classifier (Maldonado et al., 2011). For example, it can use support vector machine (SVM) as a classifier method to find the best features depending on the technique of SVM learning.

Feature selection is computationally complex since it needs many iterations to select a proper group of features. Also, this method needs to be evaluated each time features are selected.

2.4.2 Feature extraction

Feature extraction refers to the most outstanding information that can be used for data representation, classification, and visualisation. It involves building a deep hierarchy of features within unsupervised modelling (Rifai et al, 2011). Each level learns the representation, to become better than the one above. Principal component analysis (PCA) is a widely used technique for feature extraction to solve linear problems. As it is a linear

method, it can work with linear data only; it works with a structured and steady dataset (Choi et al, 2014). Using PCA, patterns are detected in the data and on the basis of these patterns, similarities and dissimilarities in the data are identified (Sun & Du, 2006). PCA helps in detecting patterns in the data that cannot be represented and analysed graphically. ICA is an unsupervised dimensionality reduction technique; with high computational complexity, that relates to data independence (Bashiri & Geranmayeh, 2011). The disadvantage of LDA is the lack of sample data per class, so the classification performance is decreased due to the generalisation of decisions for arbitrary data with noise regularisation (Zhu et al, 2009). Sang Jeen et al (Sang Jeen et al, 2003) compare between PCA and autoencoder neural network mechanisms for feature extraction to reduce the dimensionality of optical emission spectroscopy data. The results show that neural networks have more agreement with low errors between model predictions and measured data. Feature extraction is more efficient than feature selection because it has one iteration to extract the features, so a less computation and fewer steps are needed.

2.5 Pattern Recognition

Maintaining clinical datasets is an essential task to provide quality services in the healthcare industry (Assawamakin et al, 2013). The automatic medical analysis is an important application of pattern recognition. Pattern recognition is defined as the classification of data based on knowledge already achieved or on statistical information extracted from the patterns (Rutkowski et al, 2014). In the last decades, pattern recognition has attracted more interest from researchers in computer vision and machine learning (Chellappa, 2016; Karczmarek et al, 2017). It accurately detects objects when the system is efficient (Ahmad et al, 2011). The main computational steps of pattern recognition are (a) feature weights and calculation, (b) feature extraction or selection, and (c) classification (Caesarendra et al, 2015).

The motivations for development of pattern recognition techniques are (Khodaskar & Ladhake, 2014):

(a) It is an important part of the artificial intelligence field, which tries to give human intelligence to a machine.

(b) In real life problems, it can provide high quality, intelligent analysis, and classification of measurements,

(c) Data mining and knowledge discovery benefit from frameworks developed by pattern recognition techniques.

Data mining techniques applied to clinical datasets discover relationships and patterns that are helpful in studying the progression and management of diseases (Prather et al, 1997).

2.5.1 Patterns Measures

Distance measure:

The variation between pattern representations will be calculated to measure the distance. Similar patterns have minimum distances. One metric example is the Minkowski measure, which is the distance between two data points in the form;

$$dm(x, y) = \left(\sum_{k=1}^{n} |x_k - y_k|^m \right)^{\frac{1}{m}} \quad (2.1)$$

The Euclidean distance is an example of distance measure, where $m = 2$;

$$d(x, y) = \sqrt{\sum (x_i - y_i)^2} \quad (2.2)$$

This metric should have the following properties (Wang et al, 2005),

1. Positive reflexivity $d(x, y) \geq 0$

2. Symmetry $d(x, y) = d(y, x)$

3. Triangular inequality $d(x, y) \leq d(x, z) + d(z, y)$

Weighted distance measure:

For more precision when calculating the distance, a weight can be added to their values. The metric of weighted distance is of the form (Murty & Devi, 2011),

$$d(x, y) = \sum (w_k \times (x_k - y_k)^m)^{\frac{1}{m}} \quad (2.3)$$

Size of Patterns

The size of a pattern measured by the features being considered.

2.6 Classification

There are two steps to classify data, firstly, building a model that describes a predetermined class of dataset; secondly, the using the model data for training. Prediction is viewed as the construction and use of a model to evaluate the class of an unlabelled sample (Han & Kamber, 2011). The pre-processing steps help the classification or prediction process to improve accuracy, efficiency, and scalability (Han & Kamber, 2011). According to Han and Kamber the three pre-processing steps are:

1- Data cleaning: to remove the data with many missing values.

2- Treatment of missing values: this step will help reduce confusion during learning.

3- Relevance analysis: this step is known as feature selection, which eliminates features that are irrelevant or redundant.

Evaluation of Classifiers: the type of research and the target of the framework can determine the evaluation process for the classifier. Thus the evaluation is categorized into (Daskalaki et al, 2006; Margineantu & Dietterich, 2000)

- Prediction accuracy: To classify unknown patterns correctly, is the main aim of the classifier. The accuracy is measured by the percentage of a number of data points that are correctly classified for the test dataset.
- Speed or computation cost: the time spent to create the classifier for learning the dataset is a design time, then the time spent by using the designed classifier to classify a pattern is the computational time.
- Space required; is the space of the training set; if the dataset is huge, it could be reduced by dimensionality reduction or divided into portions.
- Interpretability: has easy it is to understand how the classifier chooses the class of patterns.

Methods of validation: classifiers need data for training and data for validation. Data validation methods can be categorized into (Murty & Devi, 2011)

- Holdout method: it divides the dataset into two sets one for training and one for testing; typically two-thirds are used for training.
- Random sub-sampling: different training and validation subsets are generated each time. Then the accuracy will be calculated by,

$$Acc = \frac{1}{k} \sum_{i=1}^{k} Acc_i \tag{2.4}$$

where k is the number of times to select the subsets, and the accuracy is calculated by the number of correctly classified data points from all data points.

- Cross-validation: each pattern is used the same number of times for training and testing. In k-fold cross-validation, the data is divided into k equal subsets. During

each run, one subset is used for testing and the rest of the subsets are used for training. To ensure the quality of the model, a cross-validation method is used to estimate the error rate of classifiers. In cross-validation, the dataset is partitioned randomly into N samples and evaluations are run for N iterations. At each iteration, $N-1$ samples are selected for training and the final sample is used to evaluate the accuracy of the classifier.

- Bootstrap procedure: selects a pattern randomly from the dataset without eliminating it, then selects another and so on. This is repeated n times to select N patterns.

Cross-validation will be used in this research where the data will be split into 70 per cent for training and 30 per cent for testing.

2.6.1 Classification methods

1- **The decision tree (DT)**; decision trees use a tree-graph to make predictions by decisions. Knowledge is extracted and represented in the form of classification IF-THEN rules. One rule is created for each path, from the root to a leaf node. Building a decision tree starts by selecting a feature as the root and then extending the tree for all possible features to reach the class label, and so on, as shown in figure 2-5. The aim of decision tree learning is to create a model that predicts the value of a target variable based on several input variables; it reaches the goal by determining a most likely strategy (Ravichandran et al, 2012).

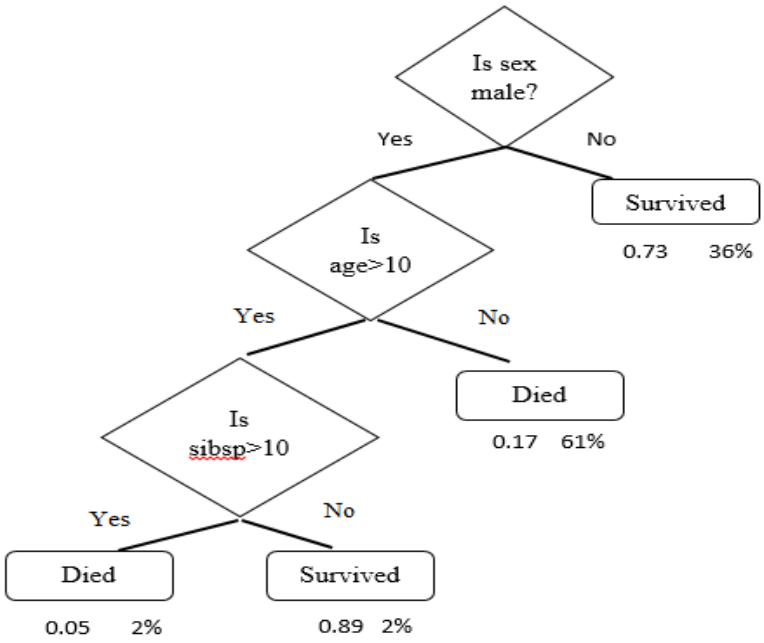

Figure 2-5 Example to Building a Decision Tree

2- **Random Forest (RF**); is a way to combine information across an ensemble, by a strategy of divide-and-conquer to generate many decision trees for the training set. For k classifiers in the ensemble there are decision trees for classifying a new element into one of m possible outcome groups. At each node, an individual decision tree determines the split on the basis of a smaller, random selection of attributes, and not from the set of all attributes, see figure 2-6. Each tree in the forest of trees then votes on the classification of a new item, and the most popular class is returned as the ensemble solution (Ledolter, 2013). The attributes of RF are that it works efficiently on large datasets, and it provides more consistent accuracy than other algorithms. Also, if we have missing values, this method estimates missing data and maintains the accuracy rate (Pal & Mather, 2003). Moreover, RF provides an estimate of important attributes in the classification (Ham et al, 2005). However, the weaknesses of RF

approach are the computation time cost, and the complex interpretability or difficulty of gaing insight.

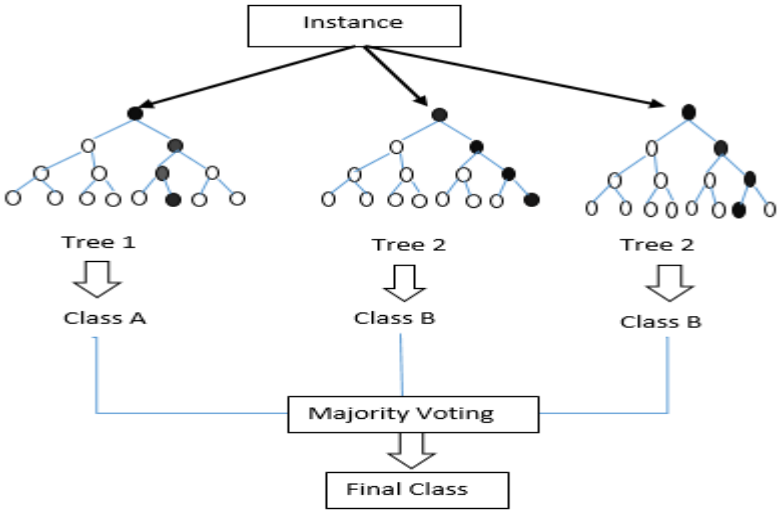

Figure 2-6 Example of Random Forest

3- J48 is a simplified version of a C4.5 decision tree, which is a development of the ID3 algorithm (Ravichandran et al, 2012). C4.5 is a divide-and-conquer technique for growing trees from the set *N* of cases by the following steps (Quinlan, 1996);

(a) The tree for *N* is a leaf for the most frequent class; if *N* has stopping criteria, then the reason for stopping is when *N* has cases for this class.

(b) Partitioning *N* into subsets N_1,N_2, …N_k for T test with outcomes T_1,T_2… T_k.

(c) The tree has evaluated for a subtree T_i that is constructed by the same procedure as the case N_i.

Therefore, J48 splits the data into ranges based on the attribute values for items that are identified in the training sample (Devasena, 2015). Given a set T of total instances the following steps are used to construct the tree structure (Gupta et al, 2012):

Step 1: If all the instances in *T* belong to the same group class, or *T* has fewer instances than the tree, then the tree is represented as a leaf labelled with the same class.

Step 2: If step 1 does not occur; then a test is selected based on a single class. This test is considered as a root node for the tree with one branch for each outcome of the test

according to the result of each respective case, and the same may be applied in a recursive way to each subnode.

Step 3: For every attribute information gain is calculated and ranked using two heuristic criteria (Kaur & Chhabra, 2014),

The J4.8 algorithm is fastest to train, it generally, gives good results although less so than random forest and its output is human readable. For the missing values, J48 eliminates missing values in the dataset while building a decision tree.

4- **REPTree (Reduced Error Pruning Tree)** is a fast decision tree learner. It builds a decision tree using information gain and prunes it using reduced-error pruning. At the beginning of the model preparation, it sorts the values of numeric attributes once (Han et al, 2011). The method creates multiple trees in many iterations, then selects the representative, which is the best tree. Pruning uses mean square error for the prediction (Kalmegh, 2015).

5- **Naïve Bayesian Classification** is a simple technique used for classifying from a subset of attributes and is a consistently performing probabilistic model. Data classification based on Naïve Bayes is the task of predicting the class of an instance from a set of attributes and describing the instance; it considers that all the variables are conditionally independent given the class (Domingos & Pazzani, 1997). Naive Bayesian provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypotheses, and it is robust to noise in input data. There are many uses of Bayesian classification, such as text

classification, spam filtering, and online applications. Naïve Bayesian probability is given by:

$$P(c|f) = \frac{P(fx|c)P(c)/P(x)}{P(f)} \quad (2.5)$$

where c is the class, f is the predictor variable, $P(c|f)$ is the posterior probability of class c given feature f , $P(c)$ a prior probability of the class, $P(f|c)$ is the probability of the variable given class also known as likelihood, and $P(f)$ is the prior probability of the variable. The algorithm starts by calculating probability for the likelihood table, then applies the Naïve Bayesian equation to find the posterior probability for the classes. Thus the prediction outcome is the highest posterior probability.

6- **K-nearest neighbours (K-NN)** is a classification technique that stores all available cases and classifies new cases based on a similarity measure, for example, distance functions. For the N training vector, K-NN identifies the k nearest neighbour of the vector C whose class we want to estimate, regardless of label. For example, if we want to predict vector C in a binary class sample, where $k = 3$, then the algorithm will calculate the distance for the three nearest neighbour vectors to C, as explained in figure 2-7. Next the algorithm will predict C to the class label that has maximum nearest labels, which is class A in our example. The main drawback of this algorithm is the complexity of search for large dataset, because it will search the nearest neighbour for each sample.

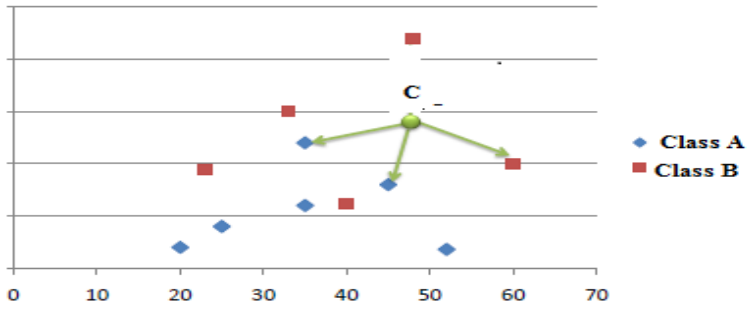

Figure 2-7 KNN algorithm example

7- **Support vector machine (SVM);** it is a supervised learning algorithm for linear classification and regression. The samples are represented by points in space. The model is built to identify and categorize these points into two groups for binary classification. These two categories have to be separated by a hyperplane (or margin), and each time the gap between categories has to be as wide as possible (Tong & Koller, 2001). The hyperplane is defined by

$$g(\vec{x}) = \vec{w}^T\vec{x} + b \tag{2.6}$$

where $\vec{x}$ is the input vector, $\vec{w}^T$ is the weight of the vector, and b is the bias. The data are separated depending on the value of threshold,

$$\vec{x} = \begin{cases} class\ A, & g(\vec{x}) \geq 1 \\ class\ B, & g(\vec{x}) \leq -1 \end{cases} \tag{2.7}$$

The margin is calculated by

$$z = \frac{|g(\vec{x})|}{\|\vec{w}\|} = \frac{1}{\|\vec{w}\|} \tag{2.8}$$

Hence, the total margin is computed by

$$z = \sum \frac{|g(\vec{x})|}{\|\vec{w}\|} \tag{2.9}$$

Minimizing $\|\vec{w}\|$ will maximize the gap between classes.

As we can see in figure 2-8, the algorithm first finds a hyperplane between two classes and then increases the gap to improve the outcome accuracy. Although in most cases, there is more than one hyper-plane that correctly separates classes, the method searches for the maximum margin between classes.

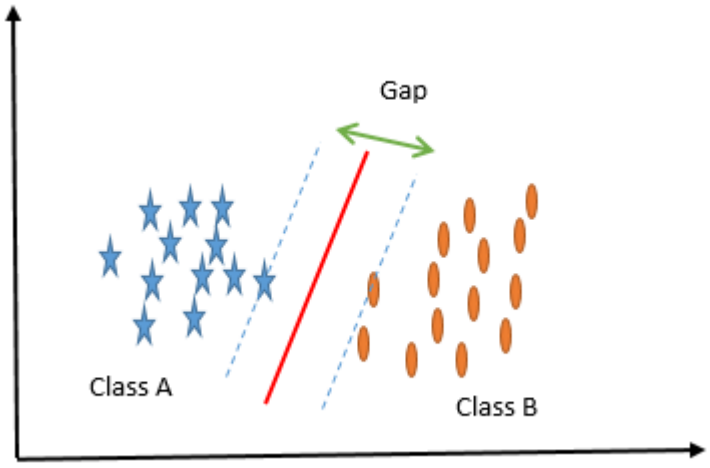

Figure 2-8 SVM algorithm example

The comparison between the classification methds is shown in table 2-2.

Table 2-1 Classification methods pros and cons

Classification method	Advantages	Disadvantages
DT	• Ability of selecting the most discriminatory feature comprehensibility so that can be used in Rule Generation problems. • Data classification without much calculation, dealing with noisy or incomplete data, handling both continuous and discrete data (it is necessary to choose the proper algorithm)	• A drawback of using decision trees is that the outcomes of decisions, subsequent decisions and payoffs may be based primarily on expectations. • The high classification error rate while training set is small in comparison with the number of classes.
RF	• It is one of the most accurate learning algorithms available. • It runs efficiently on large databases. • It gives estimates of what variables are important in the classification	• Random forests have been observed to overfit for some datasets with noisy classification/regression tasks. • Biased in favour of categorical variables with more levels
J48	• Easy to interpret and explain. • Easily handles feature interactions and is non-parametric, so it is not necessary to worry about outliers or whether the data is linearly separable.	• Has the highest sensitivity of all the other algorithms (Endo et al, 2008).

RepTree	• Reliable tools for real-time quality monitoring tasks under both network and human impairments (Yang et al, 2015). • Fast learning regression tree which is suitable for classifying numerical values (Namratha et al, 2013).	• Tree is unstable even for small changes in input data. • Large tree models are difficult to analyse (Namratha et al, 2013).
Naïve Baysian	• Tremendously appealing because of its simplicity, elegance, and robustness. • Light to train: no complicated optimisation required. • Easily updateable if new training data is received.	• Assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. • It is made to simplify the computations involved.
Knn	• Robust on noisy training data (especially if we use Inverse Square of weighted distance as the "distance"). • Effective if the training data is large.	• Need to determine value of parameter K (number of nearest neighbours). • Computing cost is quite high because we need to compute distance of each query instance for all training samples.
SVM	• By introducing the kernel, SVMs gain flexibility in the choice of the form of the threshold separation. • Since the kernel implicitly contains a non-linear transformation, no assumptions about the functional form of the transformation (Auria & Moro, 2008).	• Generally black boxes, it is not possible to read the acquired knowledge in a comprehensible way. • The lack of transparency in results. Since the dimensions might be very high, SVM might not be able to show the company's score as a parametric function based on financial ratios nor any other functional form (Karamizadeh et al, 2014).

2.7 Machine Learning Process

Machine learning (ML) refers to the question whether computers might be made to learn. This field, along with Artificial Intelligence (AI), is probably the most significant progress in the last decade (Kononenko & Kukar, 2007). Alpaydin in (Alpaydin, 2014) defines machine learning as programming a computer to optimise a performance criterion

by implementing example data or a training model. From the above, machine learning consists of two steps: first, training, which needs efficient algorithms to store and process the massive amount of data and to solve the optimisation problem. Secondly, the representation and algorithmic solution for inference need to be efficient. The ML objective is to minimise the model error through training data (Baker, 2014). These days machine learning is a cornerstone of many computing fields. The popularity of ML comes from the powerful new optimisation techniques, and the appearance of effective and user-friendly implementation tools (Kononenko & Kukar, 2007).

2.7.1 Machine learning models

The most popular techniques are:

1- Linear regression**,** a machine learning model for expressing the dependence of a response variable on several explanatory variables (Friedman et al, 2001). β_0 and β_j are unknown coefficients and X_j is the explanatory variable to predict a real value Y for the input vector $X^T = (X_1, X_2, \dots X_p)$. Where p is the number of features, the linear regression has the form:

$$f(x) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{2.10}$$

2- Multi-layer perceptron (MLP): it is a feedforward neural network, consisting of three layers, an input layer, a hidden layer, and an output layer (Wei, 2005). The hidden layer plays a filtering and synthesis role for the input, while the output presents the final response (Fusco et al, 2015). The output of one hidden neuron is:

$$z_j = g(\sum_{i=0}^{d} w_{i,j} x_i) \tag{2.11}$$

where d is the input neuron, x is a hidden neuron, g is a transfer function, and $w_{i,j}$ is the weight from the input neuron i to the hidden neuron j, x_i is the i^{th} input and x_0 is a bias which equals 1.

Artificial Neural Network (ANN) is a multi-layer perceptron; its self-adaptability, self-organization, and error tolerance make it suitable for nonlinear simulation. One of the most widely used is the back-propagation (BP) neural network. The BP consists of the input layer, one or more hidden layers, and an output layer. The network's response best matches the desired response by a training process to adjust the connection weights in the network (Li et al, 2014).

MLP is a technique that could be used for unsupervised learning by reducing the dimensionality.

2.7.2 Training methods

Machine learning processes consist of training, validation and testing subsets (Suthaharan, 2016). The attributes of machine learning employ the labelled set for estimating the dataset; also improving and resembling. Then, the second part of the dataset is used for testing; the algorithm tests some data to validate the effectiveness of the model. The training phase is an iteration process to ensure the model will obtain the best outcomes with minimum error. The training phase of machine learning is categorised into supervised learning and unsupervised learning

a- Supervised Learning

Parametrization and optimisation are the two main objectives; the response variables are used to define and differentiate these objectives (Suthaharan, 2016). If the response variable is continuous then the parametrization objective is defined as regression; on the other hand, it is defined as a classification if the response variable is discrete. The supervised learning algorithm is initiated and applied according to these main steps (Kile & Uhlen, 2012):

1. Analyse n_t operating states, and determine the response y for each, to generate a training data.
2. Predict $\hat{y}_i$ for an input x_i , by train a supervised model, as

$$\hat{y}_i = f(x_i) \quad (2.12)$$

3. Predict the response of the remaining operating states x_i , where $i = n_t + 1,\ n_t + 2, \ldots$

In reliability evaluation, the response y is a classification label, e.g. Dead vs. Alive, or Success vs. Fail (Pindoriya et al, 2011). Examples of supervised learning techniques are logistic regression and decision trees, while for a complex problem, more advanced algorithms like neural networks can be used (Kile & Uhlen, 2012).

b- Unsupervised learning

Unsupervised learning is based on the idea that there are underlying classes that are hidden within a dataset. The objective of a process is to identify a model that can partition the data into subgroups or multiple clusters (Asheibi et al, 2009). The general points of unsupervised learning are as follows (Kile & Uhlen, 2012):

1- Input dataset, and represent as data points.

2- Form n clusters of similar data points.

3- Find the centre of each cluster, i.e. the data points closest to the centre.

4- Evaluate the reliability indices of each data point closest to the centre.

The main disadvantage of unsupervised learning is that only heuristic arguments can be used, as opposed supervised learning, due to the absence of objective criteria for verifying the results. K-mean , spectral, and self-organizing clustering are the most popular clustering algorithms (Kile & Uhlen, 2012). Self-organizing maps are a suitable tool for high dimensional data. The algorithm is based on the principle that the distance between samples in the same cluster will be significantly less than the distance between samples in different clusters (Sapkal et al, 2007).

A supervised learning technique will be used in this research, where the class label is defined. We will investigate many machine learning techniques to find the best methods

that could be used for the heart failure dataset. Also, we will employ different evaluation techniques such as RF, Reptree, and J48 to compare between the outputs for these methods.

2.8 Neural Networks

A neural network is a collection of interlinked elements that are structured into layers and can be represented as a network of connections of neurones (Andreou et al, 2002). Neural networks consist of an input layer which takes in the input, a hidden layer and an output layer (Ojala, 2012). Each interlink corresponds to a numerical value called weight. The weight gets updated according to the problem being solved (Andreou et al, 2002). Each node in the hidden layer has n-inputs ($X_1, X_2 \ldots, X_n$) and a bias, figure 2-9.

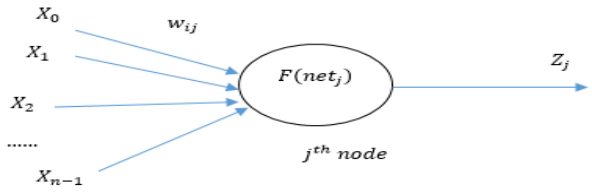

Figure 2-9 Single neuron architecture

The neurons are a summing container for the input neurons; then mapping forward the link for the layer ahead under a certain condition, see figure 2-10. There are different types of neural network architectures: feedforward, recurrent, and topological maps (Alkhasawneh & Hargraves, 2014).

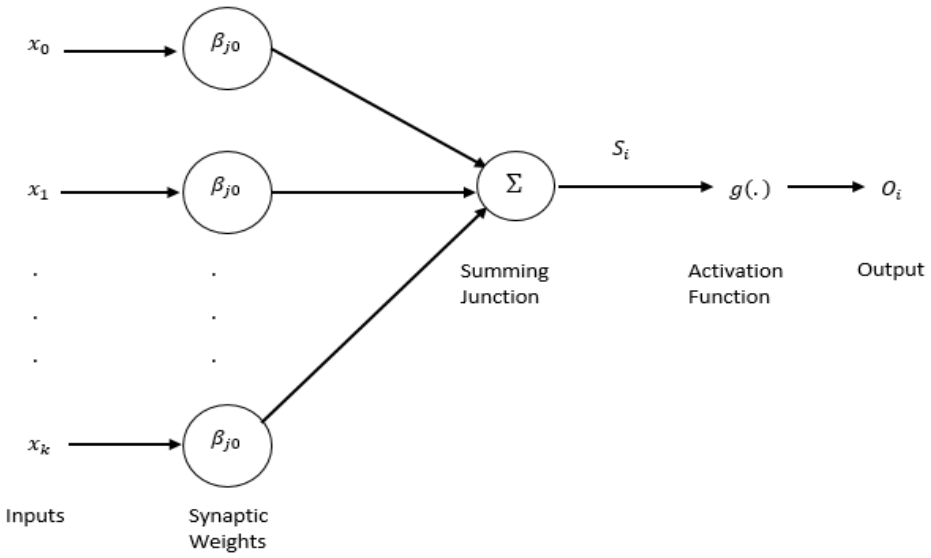

Figure 2-10 Model of a Single Neuron, see (Haykin [2009])

The neuron output is calculated by,

$$v_k = \sum w_k x(i,j) + b_k \tag{2.13}$$

where w is the weight of k-neurons, $x(i,j)$ the input value from matrix dataset for row i and column j, and b is the bias. The k^{th} processing element consists of p input values x_p, each value multiplied by synaptic weights w_{kp}, the summation over j for all values resulting in an output v_k. The output then becomes input to the activation function φ_k, which generates the output y_k, for the processing elements.

$$y_k = \varphi_k . v_k \tag{2.14}$$

There are different types of activation function, such as sigmoid or $tanh$ (tangent). The sigmoid function is defined by a logistic, where output ranges between [0, 1]. The $tanh$ is a rescaling of the logistic and its output range is [-1, 1]. The following are the equations for the sigmoid and $tanh$ functions respectively,

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.15}$$

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{2.16}$$

The sigmoid is sometimes preferred, for a number of reasons; its likely analytic tractability means it is more interpretable than the derivative of the $tanh$ function. For a

linear mapping, the linear activation functions are used in an autoencoder network like PCA.

2.8.1 Autoencoder Neural Network

An autoencoder is a feedforward neural network with one or more hidden layers, whose objective is to reconstruct the input data at the output layer (Hinton et al, 1997). The size of the output layer is the main difference between an autoencoder and traditional neural network; in an autoencoder, the size of the output layer is always the same as the size of the input layer (Tan & Eswaran, 2008). In contrast, the size of the deepest hidden layer is often smaller than the sizes of the input and output layers. As shown in figure 2-11, the autoencoder network consists of two components, namely "encoder" and "decoder". The encoder network transforms the high dimensional input data into a low-dimensional code, and the "decoder" network attempts to reconstruct the original high-dimensional data from the low-dimensional code.

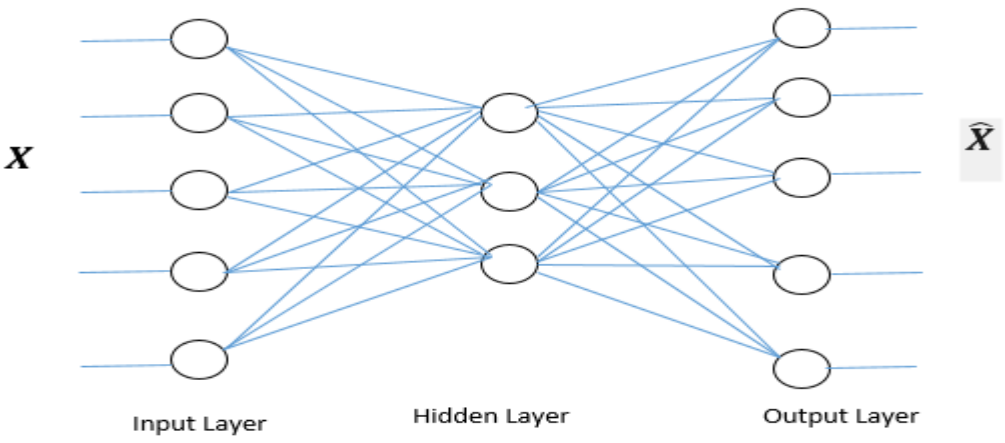

Figure 2-11 The Architecture of Autoencoder with One Hidden Layer

The networks can be trained by minimising the mean squared error (MSE) between the original and the reconstructed data. The required gradient is easily obtained by using the

chain rule to backpropagate the error derivatives first through the decoder network and then through the encoder network (Hinton & Salakhutdinov, 2006).

2.8.2 Feedforward backpropagation learning process

Backpropagation is the abbreviation for "backwards propagation of errors". Knowledge is acquired by the neural networks learning process. Knowledge is stored by the neuron weights w_{kp} Data is continuously iterated and these weights updated to make predictions (Alkhasawneh & Hargraves, 2014). Backpropagation is a supervised learning method in which the desired output for the given input must be known (Tan & Eswaran, 2009). In the backpropagation method, the activation function used in the neurons must be differentiable, since it is an implementation of the Delta rule. The Delta rule is a special case of the more general backpropagation algorithm. It is a gradient descent learning rule for updating the weights of the inputs to artificial neurons in a single layer neural network.

The output ($\hat{x}$) of a neuron is compared with the desired output (x) to derive the sum squared error. Connection weights of the nodes are adjusted according to the weight changes. For the output layer, the change in weight is given by Thompson et al (Thompson et al, 2002):

$$\Delta w_{ij} = \eta(x_l - \hat{x}_l) f'(net_l) z_j \tag{2.17}$$

where $\boldsymbol{\eta}$ is the learning rate. For the hidden layer, the change of weight is derived from (Thompson et al, 2002):

$$\Delta w_{ij} = \eta(\sum_{l=0}^{n}(x_l - \hat{x}_l) f'(net_l) w_{jl}) f'(net_l)\ z_j \tag{2.18}$$

The new weight is updated according to the Delta rule,

$$w_{new} = w_{curent} + \Delta w \tag{2.19}$$

2.8.3 Conjugate Gradient Backpropagation

Simply, this is like the backpropagation but this approach speeds up the training process compared to the traditional backpropagation with momentum (Tan & Eswaran, 2008). The line search function of the conjugate is used to locate the minimum point in the error function. The first search direction is the negative of the gradient of the performance. In the succeeding iterations, the search direction is computed according to the formula:

$$p_k = -g_k + \beta_k p_{k-1} \tag{2.20}$$

where g_k is the new gradient and P_{k-1} is the previous search direction. The parameter β_k can be computed in different ways, such as

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}\, g_{k-1}} \tag{2.21}$$

This is a gradient by the inner product of the previous change divided by the norm squared of the previous gradient.

2.8.4 Stacked autoencoder

An autoencoder with multiple hidden layers makes it difficult to optimise the weights. Training multilayer networks are implemented in phases as explained in Bengio et al (Bengio et al, 2007). During the first phase, the autoencoder is assumed to have three layers, namely, the input layer x, the output layer y, and a hidden layer h_1 , as shown in figure 2-12. Then the network is expanded by adding more hidden layers as in figure 2-13.

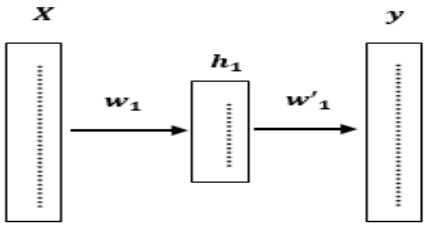

Figure 2-12 Deep Learning, One hidden layer

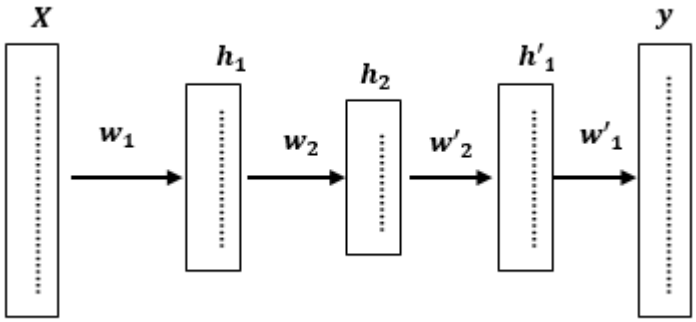

Figure 2-13 Deep Learning, Three hidden layer

The weights W_1 and W'_1 are trained using conjugate gradient backpropagation. The separate one-hidden-layer network consisting of the input layer h_1, output layer h'_1 and hidden layer h_2 is trained individually before being stacked onto the existing autoencoder. The bias and input to the hidden layer h_1 is $z_1, z_2, \ldots z_m$ where

$$z_j = f(net_j); j \neq 0; m < n \tag{2.22}$$

However, the size and the value of input and output layers are the same; the output layer h'_1 is equal to the input layer h_1. Hidden layer h_2 is a new hidden layer added onto the autoencoder. The weights of W_2 and W'_2 can be trained by using backpropagation. After the training of the separate one-hidden-layer network is completed, all the weights of the autoencoder are fine-tuned to cover a global minimum (Larochelle et al, 2007).

2.8.5 Restricted Boltzmann Machine (RBM)

It is a learning algorithm that allows optimal weights to be learnt from experience; the algorithm discovers important features that represent regularities of the training data (Salakhutdinov et al, 2007). The problem of the Boltzmann machine is that the learning stops working correctly when the machine is scaled up to non-trivial problems.

Despite the fact that learning is impractical, the General Boltzmann machine is quite efficient when the architecture is made conditional such that connections among input

units, as well as among hidden units are not allowed. The hidden units are independent, with only visible vector. Thus the learning is efficient to approximate the gradient descent in quantity called “contrastive divergence”, and works well in practice (Hinton, 2002). After the multilayer autoencoder is created, conjugate gradient backpropagation can be used to fine-tune the weights of autoencoders provided the initial weights and biases are close to the optimum solution. All the weights of the multilayer autoencoder can be fine-tuned in a single phase provided the weights are pre-trained with RBM (Tan & Eswaran, 2009).

2.8.6 Stacked RBM

It is an architecture that constitutes a stack of RBMs. After learning one hidden layer, the output of hidden units is used as the input for training the next RBM. The new RBM is stacked onto the previous RBM after training independently, as illustrated in figure 2-13. By repeating this process, a multilayer autoencoder can be created. The autoencoder created in this way has a smaller size hidden layer, which leads to dimensionality reduction. After stacking, conjugate gradient backpropagation is applied to fine-tune the overall weights. This process is repeated for stacking more hidden layers (Tan & Eswaran, 2009). It is proved that additional hidden layers improve the performance of the autoencoder, provided the number of hidden neurons per layer does not decrease, and the weights are initialized correctly.

2.9 Summary

This chapter provided a general background of data mining tools and their methodologies. CRISP-DM and SEMMA frameworks make data mining more effective and efficient for clinical data mining. The frameworks follow the intuitive stages shown in figure 2-1. However, what is different is the relationship between the three stages, which are often tailored for the application and nature of data. In contrast, there are two key challenges

involved in data mining steps: 1) agreeing on a data preparation method such as a data cleaning or pre-processing technique so that data mining methods are successfully implemented and 2) agreeing on a predictive model to predict future outcomes. In spite of this, the main goal of the frameworks consists of a particular course of action, to understand, evaluate and compare data which are mainly intended to achieve a result. There are many data mining tools used to analyze and classify datasets in different computation techniques. The outcomes are not just related to the method used but depend on the dataset itself and the preparation of data before classification. The challenges of real datasets, as discussed in chapter 1, need such tools to deal with these problems.

Data mining consists of a sequence of four steps: (a) pre-processing, (b) modelling, (c) prediction, and (d) evaluation. However these steps can be reduced; autoencoder is a model that classifies data after feature extraction so that this model can do two stages in one possible framework.

The next chapter will explore the pre-processing stage in mining data, by cleaning data and imputing missing values. Various imputation techniques will be employed to impute the missing values and these techniques will be evaluated in different classification methods.

Chapter 3. Effect of Imputation on Classification

3.1 Introduction

As discussed in Chapter 2, a data mining methodology starts with the pre-processing of the data (see figure 2-1). This stage includes the exploration of data, and its preparation looks at the characteristics of the dataset and a decision is made as to what records and variables need to be deleted, and how missing values are imputed. Data exploration allows for the gathering of information about the data itself. This would include the total number of variables, the number of records, and the percentages of missing values. Once this has been carried out, the next step is to start the process of imputing missing data, and on occasion determining the level of noise (or presence of spurious data). In clinical datasets, missing values are often present for many reasons, as discussed in the literature review (see section 2.3).

Missing values pose an important computational challenge, as these values do not enable development of algorithms for predicting the outcome or decision. Ignoring them is not an option; what can be done is to either eliminate the records with missing values, eliminate variables with a large percentage of missing values, or impute them. Thus a general rule of thumb is employed to remove records or variables. Where there is more than 30% missing, these records or variables are discarded.

In this chapter, a representative set of imputation methods that could be used is investigated. These methods range from using the mean/median of the data to using machine learning techniques for imputation. In this chapter, we will evaluate the performance of different imputation methods, and consider how the incomplete data affect the computation complexity for classification.

3.2 Missing values imputation

There are a variety of strategies to impute missing values. Some of these are:

(a) Listwise deletion strategy: here data points that contain more than a certain percentage of missing values will be removed.

(b) Remove variables that have more than 30% of values missing. Methods that discard variables or records are simple and easy, because they just delete data points with too many missing values.

(c) The mean strategy involves filling the missing values by the mean of the data for the same feature. The mean method imposes a bias on the data and is not compatible with other features such as class feature.

(d) Imputation of missing values by using machine learning techniques. The advantages of are to keep the data almost the same size, maintain the structure of the dataset and avoid bias. Imputation methods are varied and use different techniques. Some methods use the information within a variable to impute the missing values. Such methods often use the mean, median or most common value. These allow for the values to be replaced in different ways. The other alternative is to use machine learning techniques to determine the value which is missing. Some methods estimate the features data when calculating the missing values, for example, most common and expectation maximization. Other methods estimate the data points in the dataset, for example, K-mean and K-NN. Further, there are methods that impute missing values while paying attention to separation of classes for example SVM and CMC. In the research dataset, the missing values are distributed randomly. Hence, the records have variations in missing values, and also, the features have variation in missing values that appear in each attribute.

As discussed in the previous chapter the data mining methodology has a number of stages. One of these stages is a pre-processing; which includes preparing data and imputing

missing values. Imputing missing values has three steps, as shown in figure 3-1: removal of data with too many missing values, then imputation of the rest of the incomplete data, and finally, evaluating the imputation results.

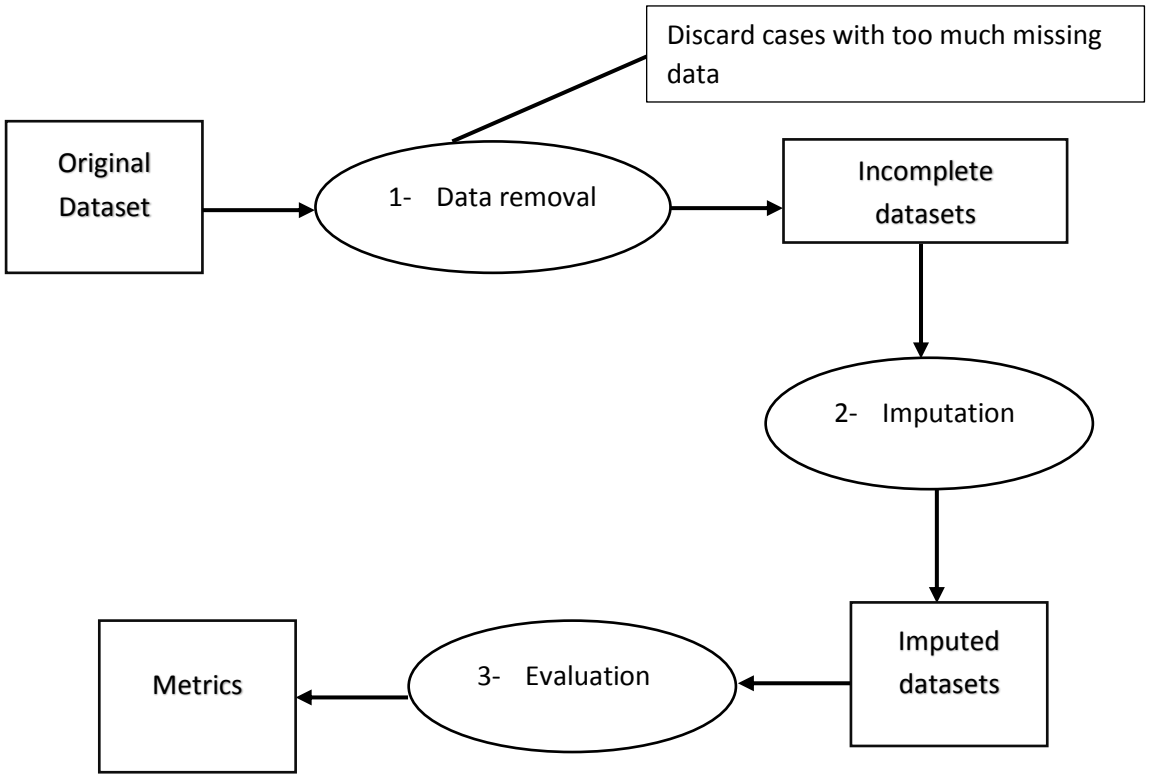

Figure 3-1 Process of Imputation of Missing Values in a Dataset

We will use six methods to impute missing values in this thesis. These methods cover all the different techniques used for imputation; they include feature based techniques, data point based techniques, and class-based techniques. The six methods are discussed in turn below:

3.2.1 Most Common Imputation (MCI)

This method works simply by finding the most common data points that occur in the feature to fill the missing value; this is when the data are symbolic. For numerical attributes, the missing values are filled by the most common of all values in the attribute (Grzymala-Busse et al, 2005). Therefore, all missing cases in one attribute will be filled by the same value. The drawback of this approach is that it can alter the distribution of

the data by overestimating the most frequent value, which often leads to improper interpretation (Su et al, 2011a). In addition, mean imputation distorts relationships between attributes by dragging estimates of the correlation toward zero (Pigott, 2001) .

3.2.2 Concept Most Common Imputation (CMCI)

The method is similar to the most common imputation, but it divides the data points into classes and finds the most common for each class (Kaiser, 2014). Thus in this approach, the algorithm replaces the missing value with the most common value that occurs in the attribute and for the same class label (Grzymala-Busse et al, 2005). Therefore, the missing values are filled by the most common value in an attribute for the same concept. This enhancement of most common imputation gives this method more power because most values of patients' records for the same class label could be similar. For example, if the most common value for the feature (Sodium) is 130 mmol/L, in MCI this value will be filled with any missing values in this feature, but in CMC it will find MC for the class Alive, which is, for example, 135, and MC for the class Dead, which is, for example 128, so if the missing values from the class Alive it will be filled by 135, otherwise by 128 mmol/L.

3.2.3 Expectation maximisation imputation (EM)

EM is an iterative procedure involving two steps, Expectation (E-step) and Maximization (M-step), adopting maximum likelihood estimates for analysing complete data (Dodge & Zoppe, 2004). The step uses the known attribute-value to estimate the parameters such as mean and covariance in the model for the data source (Karmaker & Kwek, 2005). Then for the repeated E-step, the M-step maximises the likelihood function to fill in the missing value (Kumdee et al, 2008). Thus, based on the unknown values, the E-step finds the distribution of the missing value, along with the current estimates of the parameter. Next, the M-step uses expected values to fill the missing values (Lin, 2010). The expected value

of x given the measurement y and based upon the current estimate of the parameter is computed using the current estimate of the parameter and the observed data;

$$x_i^{[k+1]} = E[x_i|y_i; p^{[k]}] \tag{3.1}$$

where x_i is the data to be estimated, and y_i is the observed data, p is the parameter such as mean (μ) and covariance (cov). New values of $\mu \; and \; cov$ will be obtained using the E-step, then the M-step applies log likelihood to maximize the estimated parameters. This will be repeated until the iteration converges or $(\mu_{k+1},\ cov_{k+1}) = (\mu_k,\ cov_k)$.

3.2.4 K-Nearest Neighbour Imputation (KNNI)

The method uses a metric to calculate the nearest neighbours between data points. The measure most commonly used is the Euclidian distance (Huang et al, 2015), which is the metric space between two points in Cartesian coordinates. The formula is

$$E(a,b) = \sqrt{\sum_{i=1}^{n}(x_{ai} - y_{bi})^2} \tag{3.2}$$

where $D(a,b) = D(b,a)$ a distance between two data points a and b, and $i \in D$ is the feature number in the set of features D. In missing value problems K-NN will divide the sample space into k-clusters; each cluster has nearest neighbour data points. The method then calculates the mean of the cluster to make a centre point to update the learning procedure and to decrease the error of classification as much as possible. There are two approaches to fill the missing values with the neighbour data point (Jonsson & Wohlin, 2004). The first approach is where there are no missing neighbours to the missing case, which allows the complete cases to be neighbours. In the second approach, the neighbours could be incomplete or complete cases.

The advantages of the KNN model are: (a) it can treat both numeric and categorical values, (b) accessibility to deal with multiple missing values, (c) no consideration of the correlations of the data (Jonsson & Wohlin, 2004; Song et al, 2008). However, K-NN

also has a drawback that this algorithm searches through all the dataset and updates the centre value while looking for the most similar instances, so it is a slow model, which is critical in the analysis of large datasets (Batista & Monard, 2003).

3.2.5 K-mean imputation

The dataset is divided into k groups based on similarity of objects and then missing value are filled by the mean of the group they belong to. The similarity depends on the distance scale between the centre of the k cluster and the objects (Li et al, 2004). This method finds the mean for a random k cluster; the distance between objects and cluster centroid is calculated with the help of Euclidean distance (Shankar et al, 2016). With k-mean, is the dataset D of data points x_i is divided into subsets S_1,S_2,…S_k, by following these four steps:

Step 1: select "k" instances randomly, from set stand for centre point.

Step 2: assign each data point to its closest chosen centre point.

Step 3: update the cluster centre.

Step 4: recalculate the positions of k-centroid.

The minimisation expression is calculated by the following equation

$$J(x,m) = \ arg \ \min_{s} \sum_{i=1}^{n} \sum_{j=1}^{k} (d(x_i, m_j))^2 \tag{3.3}$$

Let $S_j = \{x_i, i \in \{1,2,..,n\}: d(x_i, m_j \leq d(x_i, m_r), r = 1,2,\dots,k\}$

S_j is called the jth cluster and m_j is called its centroid. The disadvantage of this method is the need for determining the number of clusters, K, before analysis. Also the mean is not robust to outliers, so data far from the centroid may shift the centroid away from the real one.

3.2.6 Support Vector Machine (SVM) imputation

Although the SVM algorithm is used for classification by recognising patterns and analysis of data, it can be used to impute missing values. Adopting the chosen kernel, SVM defines the boundary between classes by selecting a set of support vectors (Sivapriya et al, 2012). As illustrated in figure 3-2, the algorithm aims to separate between the two classes by a margin and expand the gap as much as it can. The best line can divide the two classes and then classify instances based on which side of the line they appear. The value of the variable imputed then becomes a target value and avoids the original classification. However, the training algorithm is slow and requires many complex computation processes (Liu & Liu, 2010).

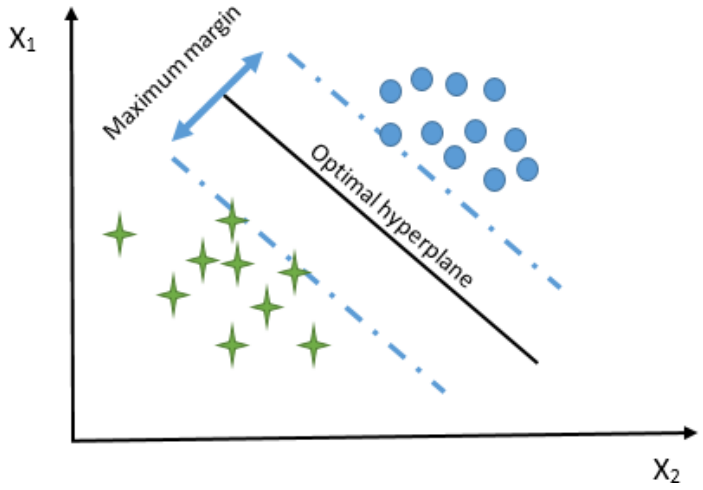

Figure 3-2 Support Vector Machine.

3.3 Experiments and Results

3.3.1 Chronic Heart Failure Dataset

The dataset called LifeLab used in this study is a dataset collected from patients with chronic heart failure; there are 1944 patients, each with up to 60 clinical variables (Cleland et al, 2016). There are many missing values in the research dataset, and most

imputation methods cannot work properly when the record has too many missing values. The missing values in the thesis dataset range from 0 to 20 per cent for the features, as shown in the appendix I. Only three features have no missing values. They are "*Age*", "*MR-proANP*", and "*CT-proAVP*". All the other 57 variables have missing values in from 13 to 396 samples. The features that have the most missing values are "Ferritin", "LVEDD (cm)", and LVEDD (High)" with 393, 380, and 396 missing, respectively. There are missing values for patients' records ranging from 0 to 60 per cent. We will exclude all instances with more than 15 per cent to 60 per cent missing values; this excludes 194 records, which amount to 10 per cent of data. when these records are eliminated from the dataset, the number of patients' records is decreased from 1944 to 1750 samples.

3.3.2 Evaluation of Classification Performance

The aggregate measure of performance is obtained using a classification algorithm and then generating a confusion matrix. This matrix in its raw form provides information on the number of correctly classified data points. However, this is further developed to provide a more sophisticated measure of performance. Table 3-1 illustrates the confusion matrix. The classification aims to classify the data as accurately as possible, and the performance measure is related to the percentage of data classified after learning on the training set (Sokolova & Lapalme, 2009). In the confusion matrix with two class labels, the classification outputs are for a binary class dataset. The two class labels are known as "positive" and "negative"; for these two labels, there are two possible results, "true" and "false" value. Thus the output will be in these categories:

- True Negative, refers to the number of samples that are in the Negative class and are correctly classified by the algorithm to the Negative class.

- True Positive refers to the number of samples in the Positive class that are correctly classified.
- False Negative refers to the number of samples that are in the Negative class but classified in the Positive class by the algorithm.
- False Positive refers to the number of samples that are in the Positive class but classified in the Negative class by the algorithm.

Table 3-1: Confusion Matrix

	Positives	**Negatives**
Classified as Positive	*True Positive (TP)*	*False Negative (FN)*
Classified as negative	*False Positive (FP)*	*True Negative (TN)*

Based on the confusion matrix, a number of crucial performance metrics such as accuracy, sensitivity, specificity, and precision are obtained. These metrics indicate how well the classification is being carried out, and in the case of selection of features, it enables the comparison of the various sets of features selected. Accuracy measures the ability of the algorithm to correctly predict the class, by finding the percentage of predictions that were correct (Chauhan et al, 2013). The formula is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

However, for clinical application, sensitivity and specificity are also crucial, in that they are measures of how well the individual classes are predicted. Sensitivity, also called true positive rate finds the percentage of the positive class that is correctly classified compared to those incorrectly classified as negative. This measures the actual positive samples which are correctly identified. Specificity, also known as the false positive rate, finds the percentage of the negative class that is correctly classified related to positives that are

incorrectly classified, so it measures the actual negative samples which are correctly identified. The positive class in our dataset refers to the "Alive" value, while the negative class refers to the "Dead" value. Low results of specificity mean the algorithm could not classify the negative class very well and vice versa.

$$Sensitivity\ (recall) = \frac{TP}{TP + FN} \tag{3.5}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.6}$$

Precision includes positive predictive value (PPV) is the percentage of the positive class that is correctly classified, i.e. the positive samples that were correctly identified. Likewise, the negative predictive value (NPV) is the percentage of the negative class that is correctly classified, i.e. the negative samples that were correctly identified.

$$Precision\ (PPV) = \frac{TP}{TP + FP} \tag{3.7}$$

$$NPV = \frac{TN}{TN + FN} \tag{3.8}$$

- The Receiver Operating Characteristic (ROC) curve, is a graphic plot evaluation of classification performance by plotting sensitivity against specificity. Thus, an increase in sensitivity will be accompanied by a decrease in specificity. Accuracy is measured in this curve by how close the curve is to the left-hand border, as illustrated in figure 3-3; conversely, the lowest accuracy is when the curve is close to the 45-degree diagonal.

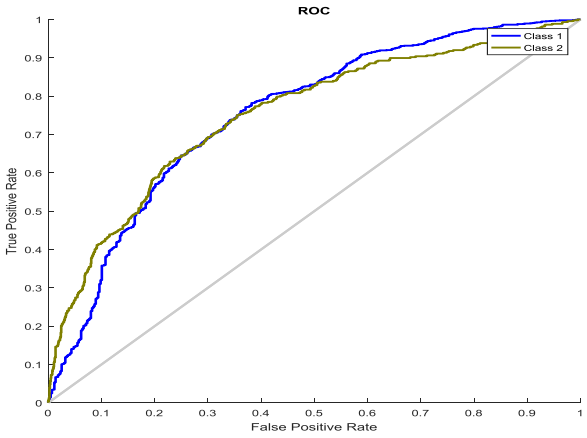

Figure 3-3 ROC curve example.

3.3.3 Analysis of Results

Imputing missing values is a pre-processing step to ensure that the knowledge discovery has suitable data, as the presence of missing data influences the tools used for data mining to obtain fair outcomes. There are various imputation methods used to impute missing values problems, as discussed in the previous section. Moreover, it is possible that different imputation methods may give different results, and also have different efficiency levels.

Although most of the imputation methods improve the classification outcomes, the most improvement is shown by SVMI and CMCI, whilst EMI and MCI show not much improvement (see Table 3-2). Results indicate CMCI and SVMI have the highest improvement outcomes with their accuracy when employing the RF algorithm being 85.71% and 84.97% respectively. These two methods follow a strategy that calculates missing values with respect to class. CMCI finds the most common value from data in the same class space, whereas SVMI separates the two classes and calculates the missing value in the same class space as well. Other methods, such as EMI and MCI, result in changing the basic nature of the properties of the data, and thus skewed distribution and shift the mean which then results in incorrect relationships being deduced (Zhang et al, 2012).

MCI finds the most common value in the feature without respect to the class or the distribution of the data points, and EMI also calculates the missing values depending on the feature variance and mean. The other two methods, which are KNNI and K-mean, provide little enhancement of the performance results, with 79.12% and 78.76% respectively. These two methods calculate the missing value without respect to the class label but divide the data into several parts in order to estimate the missing value for a part of a feature instead of all feature values. Therefore, it can be argued that an imputation method that respects the class label when calculating the missing values will provide more

accurate imputation than other methods. Indeed, the methods that calculate the missing values for the whole feature will result in incorrect imputation and thus will not improve the classification results. In figure 3-4 we can see how the methods perform, by looking at not only the accuracy but also the sensitivity and specificity. It can be seen that using any metric, both CMCI and SVMI yield improved outcomes. However for the other methods, the overall performance is mixed in terms of the various metrics. For example, sensitivity is decreased when implementing EMI, K-mean, MCI, and KNN, Also EMI and MCI do not increase the specificity measure.

From table 3-2 we can see that the learning algorithm RF consistently yield the best results compared to REPTree and J48. However, REPTree shows slight improvement over J48; RF gives greater enhancement than the other algorithms. However, RF generates too many trees; these cover all possible choices, but building a large number of trees will add to the computational complexity.

3.3.4 Numeric complexity imputation algorithms

For n-samples and d-dimensions the computational complexity is as follows (Melgani & Bruzzone, 2004; Thrun et al, 2004; Zhang et al, 2006{Melgani, 2004 #792):

-KNNI, $O(d)$ for distance of one sample, $O(n * d)$ for distance of all samples, and $O(n * k)$ is the closest samples; then the overall time is $O(nk + nd)$, because each destination $O(d)$; second step $O(n * d)$; then third step $O(n)$.

- K-mean solved the problem solved in time $O(nkId)$, where n is the number of data points, k is the number of clusters, I is the iterations, and d is the number of dimensions.

- SVM solved the problem in $O(n^2$*d), because of $O(max\ (n, d), min\ (n, d)^2)$.

- EM is the most complex as its complexity is $O(k^2 * n)$.

- MC is $O(n^2 * d)$.

- CMC is $O(\frac{n^2}{2} * d)$

As illustrated in figure 3-5, the computational time for learning algorithm RF is very long compared with REPTree and J48; being 60% more than J48 and 70% more than REPTree, because REPTree prunes the parts that make a mistake. Besides, the computational time is very long to learn classification for the incomplete dataset compared with imputed datasets. Imputed data need half the time compared to incomplete data. Also, we can see that the best methods that give high performance which are CMC and SVM, have shorter computational time than other methods, in terms of learning time.

Table 3-2: Performance Measures for Several Imputation Methods Implemented on Heart Failure Dataset, Using Different Classification Methods

Imputation Method	**Classification Algorithm**	**Accuracy**	**Sensitivity**	**Specificity**
EMI	RF	78.05%	96.42%	22.88%
	J48	71.20%	81.56%	40.04%
	REPTree	75.20%	90.55%	28.37%
K-mean Imputation	RF	78.76%	95.40%	28.66%
	J48	70.88%	80.26%	42.68%
	REPTree	75.66%	91.36%	28.45%
MC Imputation	RF	78.28%	96.26%	24.25%
	J48	68.97%	78.99%	37.44%
	REPTree	75.48%	91.24%	28.14%
CMCI	RF	**85.71%**	**97.56%**	**50.11%**
	J48	81.03%	87.43%	61.78%
	REPTree	82.57%	94.67%	46.22%
SVMI	RF	**84.97%**	**97.10%**	**48.51%**
	J48	78.57%	91.24%	55.60%
	REPTree	80.62%	92.76%	44.16%
K-NNI	RF	79.12%	95.68%	29.28%
	J48	71.14%	80.94%	41.65%
	REPTree	74.38%	90.81%	24.95%
Original Dataset	RF	78.18%	96.77%	22.26%
	J48	72.32%	84.37%	36.08%
	REPTree	74.69%	91.08%	25.36%

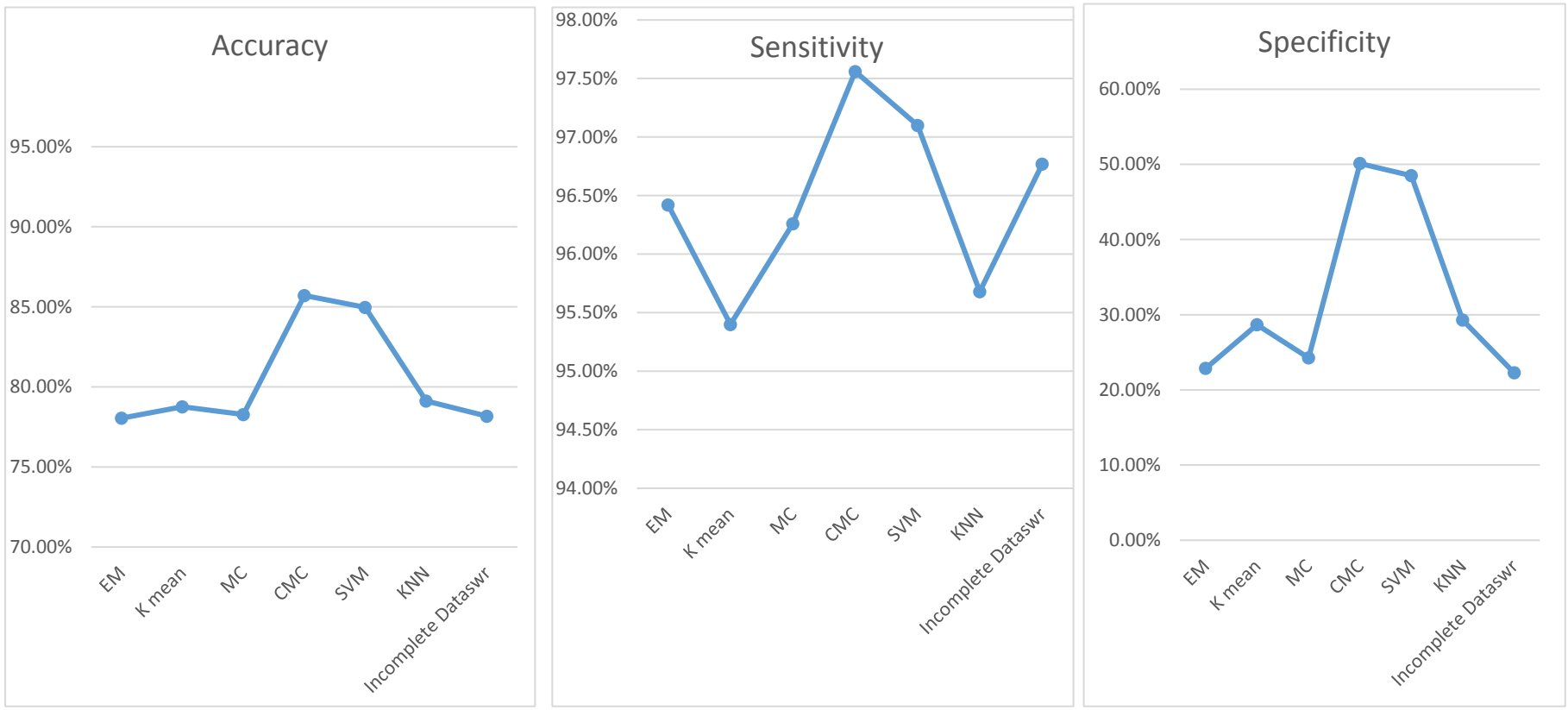

Figure 3-4 The line Charts that Shows the Outcomes of Imputation Methods Using RF method for Classification, (a) Accuracy, (b) Sensitivity, and (c) Specificity

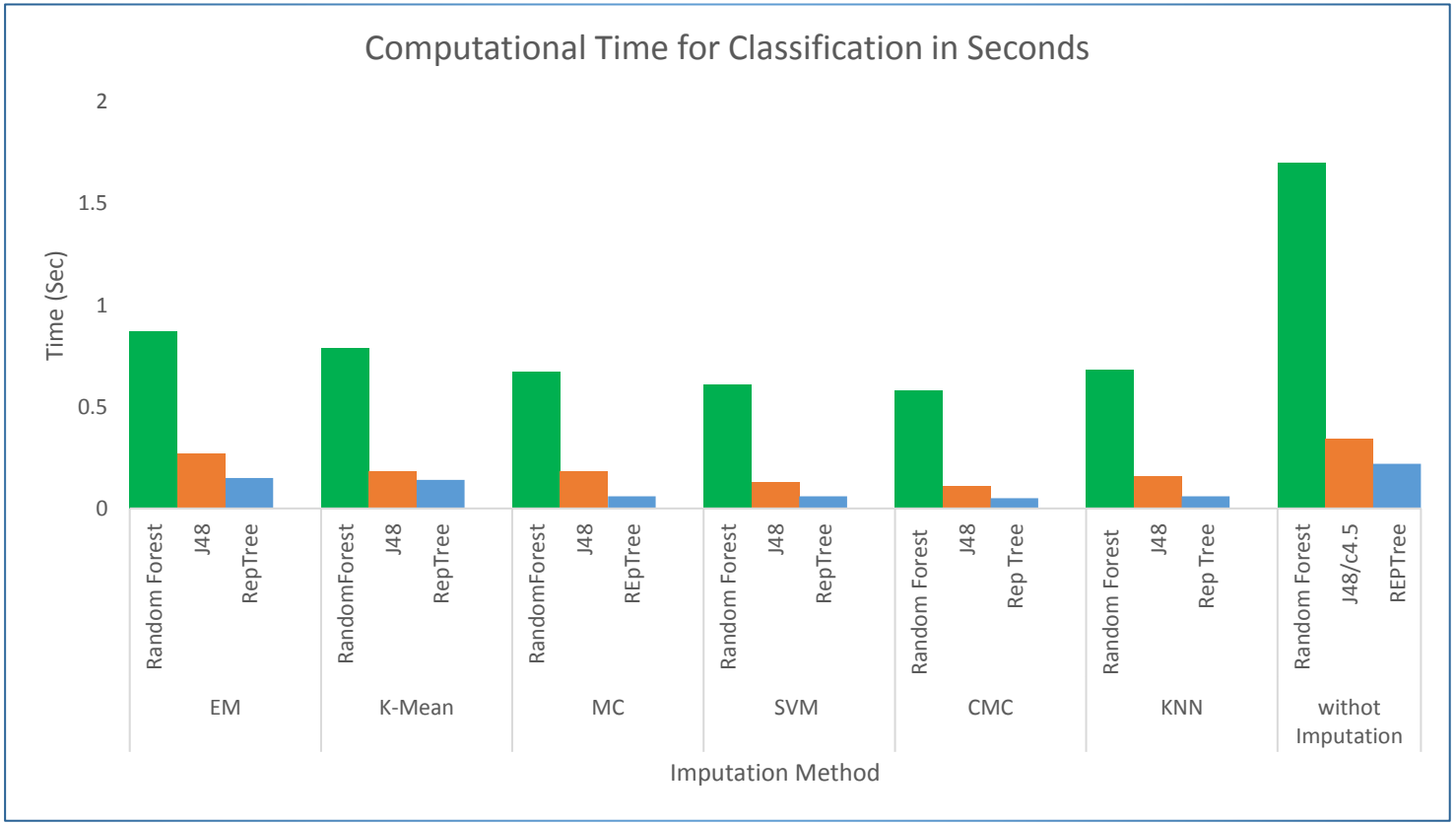

Figure 3-5 The Complexity Time of Different Classification Algorithms Used to Classify the Dataset Imputed By Different Imputation Methods

3.4 Conclusion

This chapter has investigated various imputation methods that are used for handling missing values. We illustrated that handling missing values is a significant issue in the data mining life cycle. Imputing missing values was preceded by discarding some patients' records with a large percentage of missing values, more than 15 per cent, as well as eliminating features with a large percentage of missing data. Although in the research dataset, not all features have large missing values, some patients' records have large missing values. Therefore, we discarded all records with more than 15 per cent missing data, which was 194 patients' records, so the number of patients remaining is 1750.

The imputation shown in this thesis was applied to a chronic heart failure dataset. Although many of the imputation methods used in this research gave an improvement in classification performance; some methods had a little improvement, whereas others had a significant improvement. The methods that impute missing values by separating the data points for each class, for example, SVMI and CMCI obtained significant results. In contrast the methods that evaluate data for the whole feature, for example, EMI and most common imputation did not yield an improvement , because the data in a patient's records will have close values rather than the data from two classes. Further, the methods that divided the data into different groups, for example, K-mean and K-NNI gave a little improvement.

For the computational time, we show that the RF is the most complex for learning compared with J48 and REPTree. Also, the learning process for the incomplete dataset has longer computational time compared with the imputed datasets.

Chapter 4. Dimensionality Reduction - feature selection

4.1 Introduction

High dimensionality is one of the obstacles faced in mining clinical data. A large number of features may influence the accuracy of classification and cause high computational costs. Two techniques are used to reduce high dimensionality; feature extraction or feature selection. Feature extraction transforms the existing features into a lower dimensional space, for example, PCA and LDA, whereas feature selection eliminates irrelevant features and selects the features that are most relevant. Feature selection can be categorized into (i) wrapper methods (ii) filter methods and (iii) embedded methods. In the filter methods, the algorithms measure the relevance of the feature and concept; some measures used are interclass distance, statistical dependence, and information measures. In the Wrapper method, the algorithm evaluates the prediction of the features by evolving the classifier algorithm with the dataset. The embedded method measures the interest of features by searching along with the learning algorithm by using cross-validation for assessment. The wrapper is slow in execution because it needs to train a classifier for each feature subset. Filter methods could rank features or a subset selection. Methods based on filters are simple and need little computational time. The first step of feature selection is to search for the space of a possible feature subset; the next step is to evaluation strategies to select the subset that is optimal or near-optimal with respect to objective function. Feature selection is important in the data mining process as the selected subset will reduce computational complexity and simplify interpretation. For example in a clinical dataset, the selected features can identify the main features that can determine the patient's status.

4.2 Feature Selection Problem

Understanding the feature selection problem starts with recognising the supervised learning process. Consider a dataset D that contains input space with f features $\{f_i^j; i = 1,2,3 \dots n, j = 1,2,3, \dots N\}$, where n is the number of features and N the number of data points. The output space is $\{y_i^j; i = 1,2,3, \dots m; j = 1,2,3, \dots N\}$ Then the function g is function map for each j (Rinaldi, 2009).

$$g: f_i^j \rightarrow y_i^j \tag{4.1}$$

For an input space with very large dimensions, it becomes numerically intractable. Therefore, there is a need for reducing the number of features such that the errors in predicting the outputs are below a threshold. Let S_i be the reduced set of features such that $S_i \subseteq f_i$ then, $|g - \hat{g}| \leq \varepsilon$ and $\hat{g}: S_i^j \rightarrow Y_i^j$. Thus the feature selection problem is one of finding the set S_i. A first step towards feature selection is to identify (a) irrelevant features, (b) redundant features. Once these have been determined, and removed from the set, the final selection process is started to rank the relevance of features.

The definition of relevance is as follows (Huang, 2015):

For a subset S and a set of features $F = \{f_1, f_2, \dots f_n\}$, where $S \subset F$, Y is the target concept to be learned, and r is a vector of values project to all features in S. Then, the feature f_i is relevant to Y given $S(f_i \notin S)$ iff there exists some $f_i, Y,$ and r for which

$$P(S = r, F_i = f_i) > 0 \tag{4.2}$$

such that

$$P(Y = y | S = r, F_i = f_i) \neq P(Y = y | S = r) \tag{4.3}$$

Let F be a full set of features, f_i a feature, and $S_i = F - \{f_i\}$, the categories of relevance can be formalized as follows (Yu & Liu, 2004):

- *Strong relevanc*e, input feature f_i is strongly relevant to the output class C iff

$$P(\ C|f_i\ ,\ S_i)\ \neq P(C|S_i) \tag{4.4}$$

• *Weak relevance,* input feature f_i is weakly relevant to the output class C iff

$$P(C|f_i, S_i) = P(C|S_i), \tag{4.5}$$

and $\quad \exists S_i'\ \subset\ S_i$, Such that $P(C|f_i\ , S_i') \neq P(C|S_i')$. (4.6)

• *Irrelevance, feature* f_i is irrelevant to the output class C iff

$$\forall\ S_i'\ \subseteq S_i, P(C|f_i, S_i')\ = P(C|S_i') \tag{4.7}$$

The relevance definition can be illustrated by the following $correlation\ XOR$ example (Kohavi & John, 1997). We have 5 Boolean features $\{f_1, f_2, f_3, f_4, f_5\}$ and the target concept $Y = f_1 \oplus f_2$, where $f_4 = \overline{f}_2$ and $f_5 = \bar{f}_3$, see the truth table 4-1. We can see that f_3 and f_5 are irrelevant because the target Y is dependent on the values of f_1 and f_2 . Note that Y is equivalent to $f_1 \oplus \bar{f}_4$; therefore f_2 and f_4 are weakly relevant, whereas f_1 is strongly relevant.

Table 4-1: The truth table of $\boldsymbol{Y = f_1 \oplus f_2}$ where $\boldsymbol{f_4\ = (\overline{f_2})}$ and $\boldsymbol{f_5\ = (\overline{f_3})}$

$\boldsymbol{f_1}$	$\boldsymbol{f_2}$	$\boldsymbol{f_3}$	$\boldsymbol{f_4}$	$\boldsymbol{f_5}$	$\boldsymbol{Y}$
0	0	0	1	1	**0**
0	0	1	1	0	**0**
0	1	0	0	1	**1**
0	1	1	0	0	**1**
1	0	0	1	1	**1**
1	0	1	1	0	**1**
1	1	0	0	1	**0**
1	1	1	0	0	**0**

4.3 Materials and Methods

4.3.1 Data pre-processing

The dataset called *LifeLab* used in this study is a dataset collected from patients with chronic heart failure; there are 1944 patients, each with up to 61 clinical variables (Cleland et al, 2016). Considering the data are incomplete, 1750 patients remained after those with more than 15 per cent missing values were excluded (as discussed in section 3.3.1). Data pre-processing is the most important step of data mining (Patel & Prajapati, 2016). For the purpose of this work Concept Most Common (CMC) was used to impute the missing values, because it yielded the best outcomes (as illustrated in section 3.3.3).

4.3.2 Feature Selection Methods

The dataset has large dimensionality, and some of these variables are not related to the class or the other variables. Also, some of these unwanted variables may reduce the classification performance. A variety of feature selection methods were used; a wrapper method, an embedded method by implementing a support vector machine (SVM) learning algorithm, and six filter methods; the categories for these methods are shown in table 4-2. Univariate methods mean that the relation exists between every single feature and the class label; whereas multivariate refers to the relation between groups of variables and the class. The filter methods are divided into subset methods, which are consistency and Cfs, whereas the others are ranking methods that use different analysis measures.

Table 4-2: Categorization of Feature Selection Methods

	Univariate method (Single feature evaluation- ranker)	**Multivariate (Ranker method)**	**Multivariate (Subset evaluation)**
Filter	Information gain, Chi-square, Correlation.	ReliefF	Consistency, Correlation-based feature selection (Cfs).
Embedded		Support Vector Machine (SVM).	
Wrapper	Ranking accuracy using a single feature.		

1- **Correlation-based Feature Selection** (Cfs) is a method that gives a high ranking to a subset that includes attributes which are highly correlated to the class feature with low correlation to other variables. Therefore, it evaluates a subset's interrelation with the concept rather than evaluating attributes individually (Hall & Holmes, 2003). The method considers the individual capability of each attribute according to the amount of redundancy between them (Guyon & Elisseeff, 2003). The relevant features are identified by the probability of

$$P(C = c|f_i = v_i) \neq P(C = c) \tag{4.8}$$

where f_i is the feature i, v_i feature's value where $p(f_i = v_i) > 0$, and C a class label. The method calculates the correlation between the features and the concept variable, by the Pearson coefficient:

$$r_{zc} = \frac{k\overline{r_{zi}}}{\sqrt{k + k(k-1)\overline{r_{ii}}}} \tag{4.9}$$

where k is the number of components, $\overline{r_{zi}}$ the average correlation of feature-class, and $\overline{r_{ii}}$ the average correlation of feature-feature (Hall, 1999). As a subset method, Cfs has a number of candidate subsets equal to 2^N, where N is the number of dataset features. The steps for subset selection are:

- Select a subset of attributes that have good predictive power.
- Add new features to the existing set if using sequential forward selection, otherwise, remove features from the existing set if using sequential backwards selection.
- Stop criteria, evaluate the subset by classification error or category distance measurement.

2- **Consistency Subset evaluation:** this is another subset evaluation method that selects a subset of attributes depending on the level of consistency in the class values when the training instances are projected onto the subset of variables (WEKA 3.6.9). A Consistency algorithm measures the smallest subset of features that can separate

classes i.e. the two equal values of instances must belong to the same class label (Dash & Liu, 2003). The consistent feature form a set $\{f_1, \dots, f_n\}$ (Shin et al, 2011):

$$\mathrm{P}(C = c \mid F_1 = f_1 \ , \dots \ , F_n = \ f_n) = 0 \ or \ 1 \tag{4.10}$$

Inconsistency rate (also known as consistency measure) is calculated as follows (Dash & Liu, 2003),

a. If the same two instances differ in the class label it is considered as an inconsistent pattern.

b. The number of times the inconsistent pattern appears minus the largest number of different classes is called the inconsistency count.

c. The inconsistency count over all patterns of the feature subset's divided by all instances; the outcome is called the inconsistency rate. The proportion of these inconsistent examples in the total number of examples is given by:

$$Inconsistency = \frac{Number\ of\ inconsistent\ examples}{number\ of\ examples} \tag{4.11}$$

Then the *consistency* is defined as:

$$Consistency = 1 - Inconsistency \tag{4.12}$$

3- **Chi-square Attribute Evaluation**; it is a simple filter approach that ranks features according to value chi-square for two-way tables; the two-way table for this case is a confusion matrix. The method tests the dependence of two variables, therefore it does not examine the redundancy between the attributes, as the variables are measured individually with respect to the class. It is used to test if the amount of a specific variable and the amount of a specific class are independent. Thus, the method calculates the chi-square between every feature and the target by observed data points. Then, if the relation is independent, we can exclude this variable. Chi-square is identified by:

$$\chi^2 = \sum \frac{(O - E)^2}{E} \tag{4.13}$$

where O is the observed value and E is the expected value; high scores mean that features f_i and f_j are dependent. Chi-square shares similarities with the coefficient of determination R^2, except that R^2 is applicable only to numeric data.

4- **Correlation Coefficient Evaluation:** one of the simple methods is Pearson's Correlation Coefficient which measures a linear correlation between two variables. The resulting value lies between $[-1, 1]$, where -1 refers to a negative relation, 1 refers to a positive relation, and 0 is no relation between the two variables. The Pearson correlation coefficient is defined as:

$$R(i) = \frac{cov(f_i, Y)}{\sqrt{var(f_i)var(Y)}} \tag{4.14}$$

where cov designates the covariance and var the variance, f_i an input feature and, and Y the output feature. The feature selection technique uses this algorithm to estimate the relationship between each variable and the concept variable, and chooses only the variables that have a high positive or negative relation. Figure 4-1 shows high positive correlations for some features in the thesis dataset.

5- **ReliefF Attribute Evaluation**: a supervised feature weight estimation, that measures attribute quality to select a feature subset (Demšar, 2010). The idea of a Relief algorithm is that it randomly selects features to estimate the feature's weights, using the difference in features value from the nearest sample (Jia et al, 2013). The algorithm selects a random data point j, then searches for the k nearest neighbour from the same class (called the nearest hit) and each different class (called nearest miss) as illustrated in figure 4-2 (Durgabai, 2014). The formula for updating the weight is as follows:

$$W^j = W^j - (f_i - H)^2 + (f_i - M)^2 \tag{4.15}$$

where W^j is the weight of feature f, j is a random data point, M is the nearest miss, H is the nearest hit.

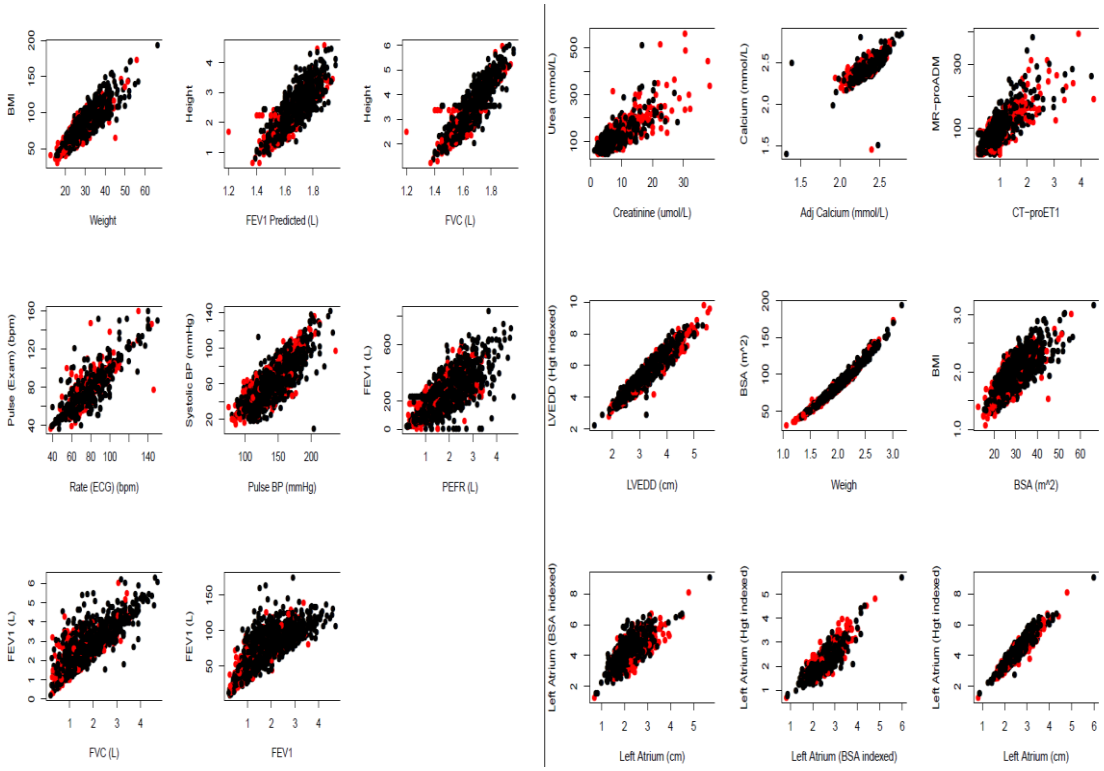

Figure 4-1 Pearson's Correlation between Features with High Positive Correlation in the Thesis Dataset.

set all weights W[A] = 0.0

$for\ i = 1\ to\ m$ do

begin

randomly select an instance R

find k nearest hits H_j

$for\ each\ class\ C \neq class(R)$

find k nearest misses $M_j(C)$

$for\ A = 1\ to\ \#attributes$ do

$$W[A] = W[A] - \sum_{j=1}^{k} \frac{diff(A,R,H_j)}{m \times k} +$$

$$\sum_{c \neq class(R)} \left[\frac{P(C)}{1-P(class(R))} \sum_{j=1}^{k} diff(A,R,M_j(C))\right]/(m \times k)$$

end

Figure 4-2 Relief algorithm

6- **Information Gain Attribute Evaluation** is a method using entropy to measure the relevance between the feature and the class label (Novakovic, 2009)**.** The entropy y is a conditional probability after observing f_n

$$H(y) = -\sum_{i=1}^{n} P(f_i) \sum p(y|f_i) \log_2 \frac{(P(y|f_i)}{P(y)} \tag{4.16}$$

where $H(y)$ is the entropy of the given dataset. It is a symmetric measure; the information gained by $H(y)$ after observing $H(f_i)$ is equivalent to the information about $H(f_i)$ after observing $H(y)$ (Novaković et al, 2011). Therefore, values vary from 0 (no information) to 1 (maximum information). If $IG(f, Y) = 0$ then the two variables are independent, otherwise they are dependent. Sui (2013) has some definitions of relevance by *IG;* one of them is using *IG* to measure the prediction power of the attribute, f_i is releveant iff

$$IG(F) > IG(F - f_i) \tag{4.17}$$

where $F - f_i$ is the feature subset resulting from removing feature f_i from feature set F. By this definition the feature is relevant if the information achieved by learning the feature subset $F - f_i$ is less than the information achieved by set F.

7- **Wrapper filter** the wrapper uses the induction algorithm itself as part of the evaluation function to search for a good subset (Cohen & Hirsh, 1994). The feature selection algorithm then occurs as a wrapper around the induction algorithm used as a "Black Box", as in figure 4-3. The Black Box is used to prompt a classifier that will be convenient to classify future instances, and the algorithm leads to a search for a high-performance subset in terms of classification (Kohavi & John, 1997). The evaluation involves the induction algorithm using cross-validation to evaluate the precision of the learning scheme for a set of features approximating the accuracy using estimation

accuracy techniques. The search space size for n features is 2^n with complexity $O(n)$, due to each state in the search space representing a subset. wrapper is unfeasible for computationally intensive methods as it must train a classifier for each feature subset (Zhang et al, 2015). The wrapper method relies on heuristic searching for all possible search subsets. An example of a heuristic search is hill climbing. The hill climbing algorithm shown in figure 4-4 displays addition of features one at a time until no further improvement can be achieved.

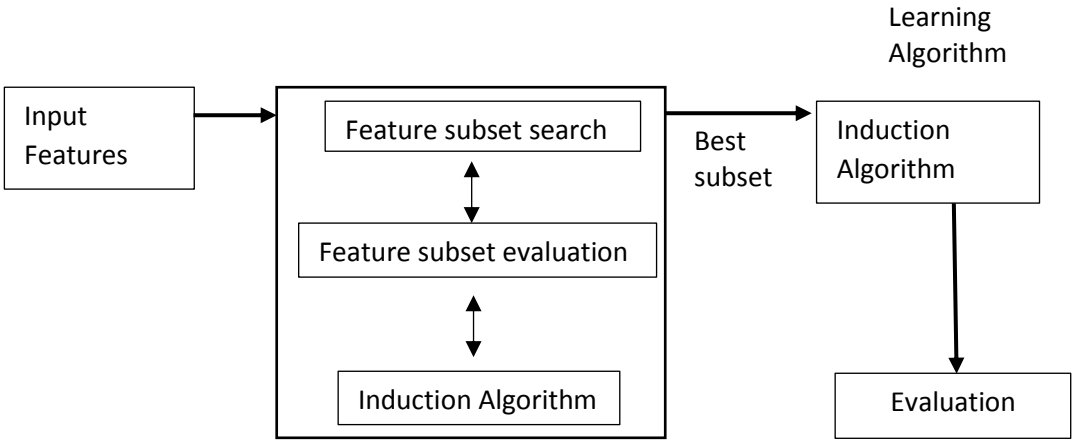

Figure 4-3 The Wrapper Method for Feature Selection

1: Let $S \leftarrow initial\ state.$

2: Expand S: apply all operators to S, given v's children.

3: Apply the evaluation function $Eval$ to each child w of S.

4: Let v' = the child w with highest evaluation $Eval(w)$.

5: If $f(v') > f(S)$ then $S \leftarrow v'; goto\ 2$

6: Return S.

Figure 4-4 Hill-Climbing Search Algorithm.

8- **Support Vector Machine (SVM)**, Embedded methods are different from the filter and wrapper methods as the feature selection is performed in the process of the

learning classifier (Hamed et al, 2014). The embedded method of searching for an optimal subset of variables adopts the structure of the classifier; embedded algorithms compromise the interaction with the classification model. The SVM algorithm uses a margin hyperplane to ensure the two patterns are separated linearly; the hyperplane maximises the sum of distances between the margin and hyperplane, see figure 4-5 (Wahed & Wahba, 2003). The model is trained with all features by setting the coefficients associated with the features to 0 and attempting to remove these features while preserving model performance. If the classes are not linear then a variant of SVM is used (Huang et al, 2015; Ozcift, 2012)*:*

1. Train a regular linear SVM.
2. Re-scale the input variables by multiplying them by the absolute values of the components of the weight vector w obtained.
3. Iterate the first 2 steps until convergence.

For n features the number of subsets tried is equal to n with computational complexity equal to $O(log\ n)$.

1: Converged: = FALSE, $\theta := 1$

2: **while** converged==FALSE **do**

3: $[f_l'\ , \alpha\ , b]$ = trainSVM($F'\ , Y'\ , \theta\ , C$)

4: θ^*= apply_Bundle_Method ($F'', Y'', f_l', \alpha, b, C$)

5: **if** $\theta^* == \theta$ **then**

6: converged=TRUE;

7: **end if**

8: $\theta\ =\ \theta^*$

9: **end while**

Figure 4-5 Algorithm Iterative Dimensionality Reduction for SVM

4.3.3 Numeric complexity of feature selection algorithms

Feature selection is an iterative process, as shown in figure 4-6. The subset generation is a heuristic search in which each state specifies a candidate subset for evaluation (Kumar & Minz, 2014).

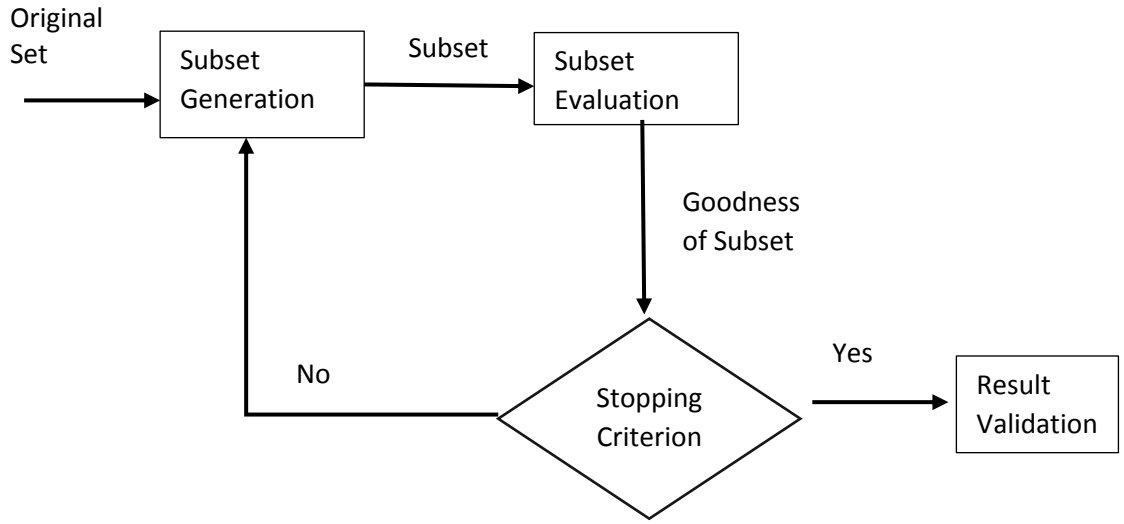

Figure 4-6 Feature selection process (Dash & Liu, 1997)

For a dataset D with number of features N, the number of candidate subsets is 2^N. The search strategy is exhaustive; the search space will be $O(N^2)$ and the number of possible steps is $O(N)$. In terms of time, for filter and embedded methods, complexity is $O(logn)$ for n number of subsets tried for feature ranking. However for the wrapper it is more computationally costly as the search space size for n features is 2^n with complexity $O(n)$. Thus in terms of complexity $filter < embedded < wrapper$; and if complexity is an important element, selecting a filter method would be ideal. However, if other considerations are to be included, then it is possible that another form of feature selection method would be selected (See section 4.4.1).

The evaluation process is another complexity added to the complexity of feature selection, whereas the validation step is not part of the feature selection process but must be performed by carrying out different tests and comparisons. The selected subset of features needs more iterations for classification for assessment.

4.4 Analysis Results

The number of variables selected by the feature is different and depends on the selection methods used. Methods such as Embedded, ReliefF, chi-square, and wrapper obtain the best performance results with 15 features, as illustrated in table 4-3. Other methods need a large number of features like correlation and information gain, obtaining the best results with 29 and 34 features, respectively. Moreover, the filter methods, which select subsets of features such as consistency and Cfs, selected 11 and 23 features, respectively. Further, the performance accuracy for the smallest set of features is 87.20% while it is 85.70% for the largest set of features selected, as shown in table 4-3. Thus, increasing the number of features selected does not ensure an increase in performance.

Table 4-3: The number of features that gain the best performance in the feature selection methods

Algorithms	**No. of Features with Best performance**	**Performance accuracy**		
Ranking methods		**RF**	**J48**	**REPTree**
Embedded (SVM)	15	87.14%	82.85%	84.28%
ReliefF	15	87.20%	83.8%	83.9%
Chi-square	15	88.50%	83.77%	84.50%
Wrapper	15	88.85%	84%	84.45%
Correlation	29	86.11%	82.68%	81.25%
Information Gain	34	85.70%	82.70%	82.0%
Subset methods				
Consistency	11	87.20%	82.97%	83.70%
Cfs	23	87.82%	83.42%	84.11

Each feature selection method chooses a different number of features that attained best classification performance; some of the algorithms improved the classification performance with a large set of features, while others worked well with a small set of features. The accuracies of embedded, ReliefF, chi-square, and wrapper methods increased as the set of features expanded until reaching a maximum of 15 features, then decreased again, as shown in figure 4-7.

All performance results using RF as a learning algorithm are shown in table 4-4.In general wrapper and chi-square obtained the best results with a number of features lower than 30 variables, then, the results decreased with the same minimum consequence at 85.48%. The wrapper method generally achieved better recognition rates than the other methods because it is tuned to the specific interaction between the classifier and dataset. In addition, the wrapper is successful for identifying strong feature relations because this method takes into account feature dependencies. The second two methods that had almost similar results are ReliefF and embedded SVM, which returned the highest results with 15 features. The methods that produced the worst results were Correlation and Information gain; they started with 78% and 79% respectively by selecting 11 variables.

This is because the information gained by entropy and Pearson's correlation is not focused on redundant variables, but can exclude variables that may contribute to prediction more than other features that have more information but can be swapped with one another. The two subset selection methods, Consistency and Cfs selected 11 and 23 attributes from 60 variables, with performance results of 87.20% and 87.82% respectively. Consequently, the performance accuracy of the full dataset is 85.45%, which indicates that all feature selection approaches can improve the performance of classification, on the condition of finding an appropriate number of attributes. Moreover, it can be seen that the chi-square and wrapper methods consistently performed better than full data where any number of features could be used. The differences in performance accuracy are illustrated in figure 4-9, where we can see a big difference in accuracy, with the Correlation and Information gain methods yielded of 4% and 3%, respectively, while the least difference resulted from using the wrapper method with less than 1% change.

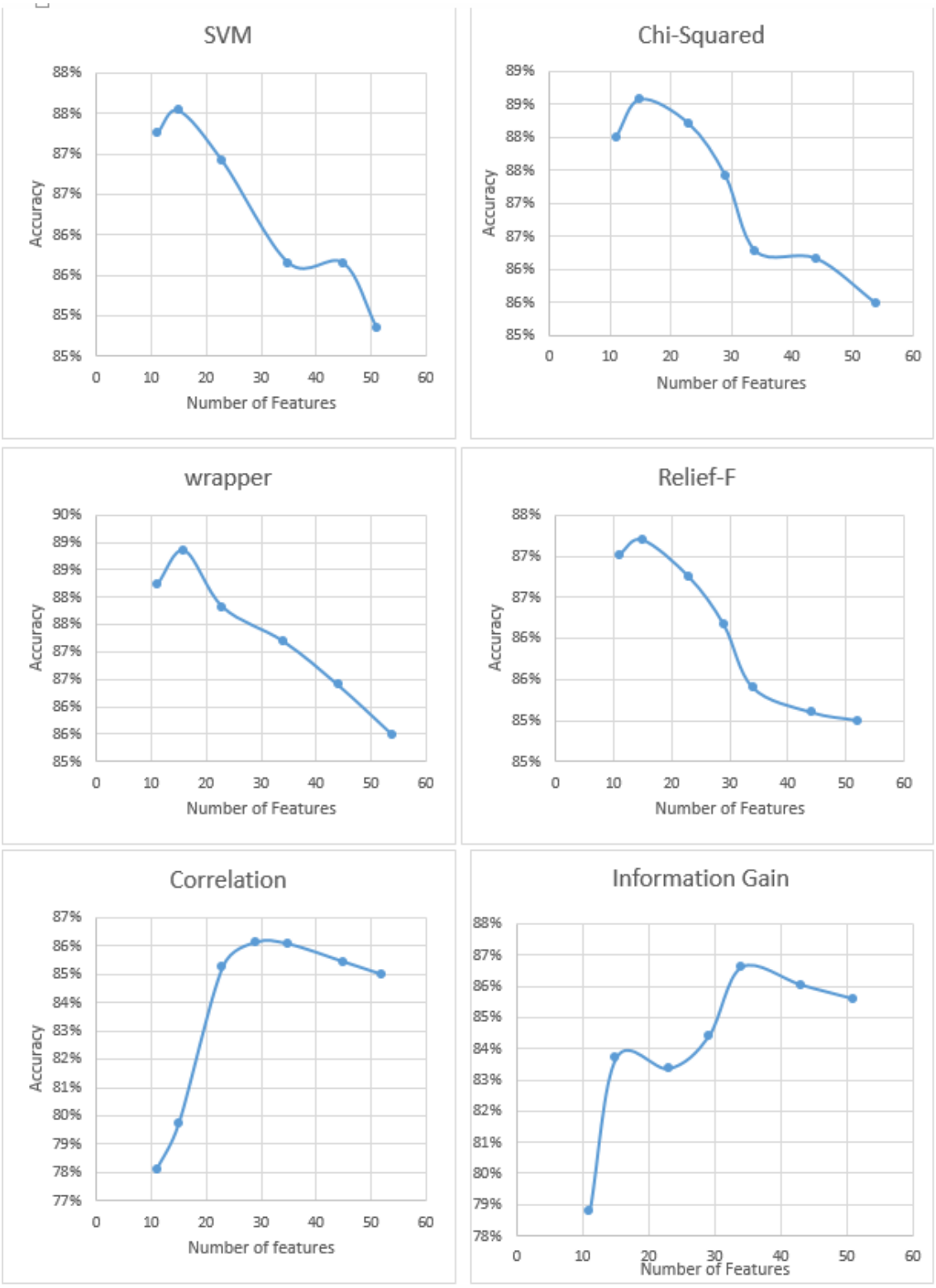

Figure 4-7 The Accuracy Obtained by Ranking Feature Selection Methods for Different Numbers of Features Used

Sensitivity and specificity outcomes for different classification methods are explained in figure 4-8. The results with the highest level of accuracy were obtained when using the

RF classification method, followed by REPTree; the lowest accuracy outcomes came from using the J48 classification method. In the sensitivity diagram, we see that the wrapper and chi-square methods have the highest sensitivity with 98%. In contrast, the highest sensitivity using REPTree comes from the Information gain method followed by chi-square and Cfs, while Consistency gets high sensitivity using the J48 classification method. RF has the highest specificity results, followed by J48 then REPTree. The wrapper method has the best specificity results with 62% using the RF classification method, followed by Information gain then Cfs. Information gain gives the highest specificity results using J48, followed by chi-square.

Table 4-5 displays the attributes' ranking for each feature selection method, and the subset attributes designated from a subset selection method. The best methods rank almost equal in the ordered list. For example, in the chi-square and wrapper methods, the 15 best performing features are {*LVEDD (cm), MCV, Ferritin, LVEDD (High Indexed), E, Iron, Hct, CT-proET1, Left Atrium, Aortic Velocity, Uric Acid, MR-proADM, Aortic Root, VitaminB12}* . These attributes contribute to prediction, such that if the selected subset contains most of the above list, it will enhance the predictive model.

Therefore, we can say that the subsets gained by chi-square and wrapper contain the variables with the strongest relevance; since these two methods have the highest performance and high similarity. SVM and ReliefF have different subsets although they obtained good results; this is because in some cases combining two weak variables can produce a high related variable. As for the results obtained from Cfs technique, as we know, the Cfs method tries to find a subset that has less correlation with other variables and high relevance to the concept variable. This method shows good results with 23 variables, which means some variables with weak relevance are added to those with strong relevance; the added weakly relevant variables can work together to create strong variables.

Table 4-4: Classification Performance for Selected Number of Features from Different Feature Selection Methods using RF Algorithm

Algorithm	No. of Features	Accuracy	Sensitivity	Specificity	PPV
Chi-square	11	88%	0.968	0.622	0.885
	15	**88.57%**	**0.980**	**0.602**	**0.881**
	23	88.20%	0.979	0.593	0.878
	29	87.40%	0.973	0.577	0.874
	34	86.28%	0.966	0.551	0.866
	44	86.17%	0.971	0.533	0.862
	52	85.37%	0.971	0.594	0.854
Correlation	11	78.11%	0.919	0.368	0.814
	15	79.71%	0.929	0.400	0.823
	23	85.25%	0.955	0.545	0.863
	29	86.11%	0.959	0.571	0.871
	34	86.05%	0.960	0.563	0.868
	44	85.42%	0.965	0.522	0.858
	52	84.97%	0.968	0.494	0.852
Information	11	78.97%	0.928	0.373	0.816
	15	83.71%	0.945	0.513	0.854
	23	83.37%	0.949	0.487	0.848
	29	84.40%	0.958	0.501	0.852
	34	86.62%	0.964	0.572	0.871
	44	86.05%	0.967	0.540	0.863
	52	85.60%	0.970	0.515	0.857
ReliefF	11	87.02%	0.967	0.579	0.873
	15	**87.20%**	**0.969**	**0.581**	**0.874**
	23	86.74%	0.970	0.558	0.868
	29	86.17%	0.967	0.545	0.865
	34	85.42%	0.958	0.542	0.863
	44	85.08%	0.971	0.490	0.851
	52	85.02%	0.969	0.494	0.852
SVM	11	87.25%	0.961	0.606	0.880
	15	**87.54%**	**0.969**	**0.595**	**0.878**
	23	86.91%	0.964	0.584	0.874
	29	85.88%	0.966	0.538	0.863
	34	86.28%	0.965	0.556	0.867
	44	85.65%	0.971	0.513	0.857
	52	84.68%	0.969	0.481	0.849
Wrapper	11	88.22%	0.977	0.597	0.879
	15	**88.85%**	**0.977**	**0.622**	**0.886**
	23	87.82%	0.976	0.584	0.876
	29	87.25%	0.977	0.558	0.869
	34	87.20%	0.971	0.574	0.873
	44	86.40%	0.975	0.531	0.862
	52	86.45%	0.978	0.524	0.861
Consistency	11	87.20%	0.966	0.588	0.876
CFS	23	87.82%	0.978	0.579	0.875
Full Data	60	85.48%	0.975	0.494	0.853

Table 4-5: The Order List of the First 22 Features from All Features Selection Methods

*hint: where the number before the feature name is the result of the employed method.

	Consistency	CFS	Chi-square	Wrapper	SVM	ReliefF	Correlation	Information Gain
1	Age(yrs)	Chloride	433 LVEDD(cm)	0.192 LVEDD(cm)	CT-proET1	0.03 LVEDD(HgtIn)	0.281 Urea	0.055 MR-proANP
2	Phosphate()	Urea	426 LVEDD(HgtInd)	0.170 MCV	MR-proADM	0.03 LVEDD(cm)	0.271 MR-proANP	0.051 Urea
3	UricAcid(mm)	Albumin	394 MCV(fL)	0.169 Ferritin	Urea(mmol/L	0.018 MR-proANP	0.252 LeftAtrium(BSAInd	0.050 CRP
4	MCV(fL)	UricAcid(mm)	371 Ferritin(ug/L)	0.168 LVEDD(HgtInd)	LeftAtrium(BSA Ind)	0.016 FEV1(L)	0.251 Creatinine	0.047 PCT
5	Iron(umol/L)	Glucose	368 Hct(fraction)	0.164 E	AlkalinePhosp	0.015 Age	0.233 Albumin	0.046 Creatinine
6	Ferritin(ug/L)	Triglycerides	364 Iron(umol/L)	0.160 Iron	MCV(fL)	0.013 FEV1	0.226 CT-proAVP	0.039 Age
7	CRP(mg/L)	WhiteCellCoun	362 E	0.160 HCT	Albumin(g/L)	0.013 CT-proET1	0.224 FEV1(L)	0.039 FEV1(L)
8	CT-proET1	MCV	348 CT-proET1	0.156 CT-proET1	LVEDD(HgtIn	0.013 Iron	0.223 Age	0.037 Albumin
9	CT-proAVP	HCT	330 MR-proADM	0.142 LeftAtrium(BSAIn	Creatinine()	0.013 Vitamin	0.216 LVEDD(HgtInd)	0.037 LeftAtrium(BSAind
10	LVEDD(HgtIn	Iron	315 LeftAtrium(BSAInd)	0.141 AorticVelocity	WhiteCellCnt	0.013 PEFR(L)	0.199 FEV1	0.036 CT-proAVP
11	E	VitaminB12	303 AorticVelocity(m/s)	0.140 UricAcid	BSA(m^2)	0.013FVC-Predicd(L)	0.195 Chloride	0.033 FVC (L)
12		Ferritin	295 UricAcid(mmol/L)	0.128 MR-proADM	Iron(umol/L)	0.011 FVC(L)	0.192 CRP	0.032 Chloride
13		TSH	219 AorticRoot(cm)	0.101 AorticRoot	MR-proANP	0.011 MR-proADM	0.192 MR-proADM	0.029 CT-proET1
14		MR-proANP	214 VitaminB12(ng/L)	0.094 VitaminB12	DiastolicBP(m mHg)	0.010 FEV1 Predictd(L)	0.185 FVC(L)	0.028 DiastolicBP
15		CT-proET1	205 LeftAtrium(cm)	0.093 Glucose	FEV1	0.010 FVC	0.184 leftAtrium(Hgt)	0.027 FVC
16		LVEDD(cm)	200 Triglycerides(mmol/L)	0.091 Triglycerides	Age(yrs)	0.010 PulseBP(mmHg)	0.184 FVC	0.026 Haemoglobin
17		LVEDD(HgtInd)	196 Glucose(mmol/L)	0.083 TSH	FVCPredicted(L)	0.008 Pulse(EXAM)(bpm)	0.184 Iron	0.023 Weight
18		AorticRoot	192 TSH(mU/L)	0.081 Cholesterol	Height(Exam	0.008 Rate(ECG)	0.181 DiastolicBP	0.021 SystolicBP
19		LeftAtrium(cm)	174 Cholesterol(mmol/L)	0.080 LeftAtrium(cm)	Chloride(mmo	0.008 BSA(m^2)	0.178 AlkalinePhosp	0.021 FEV1

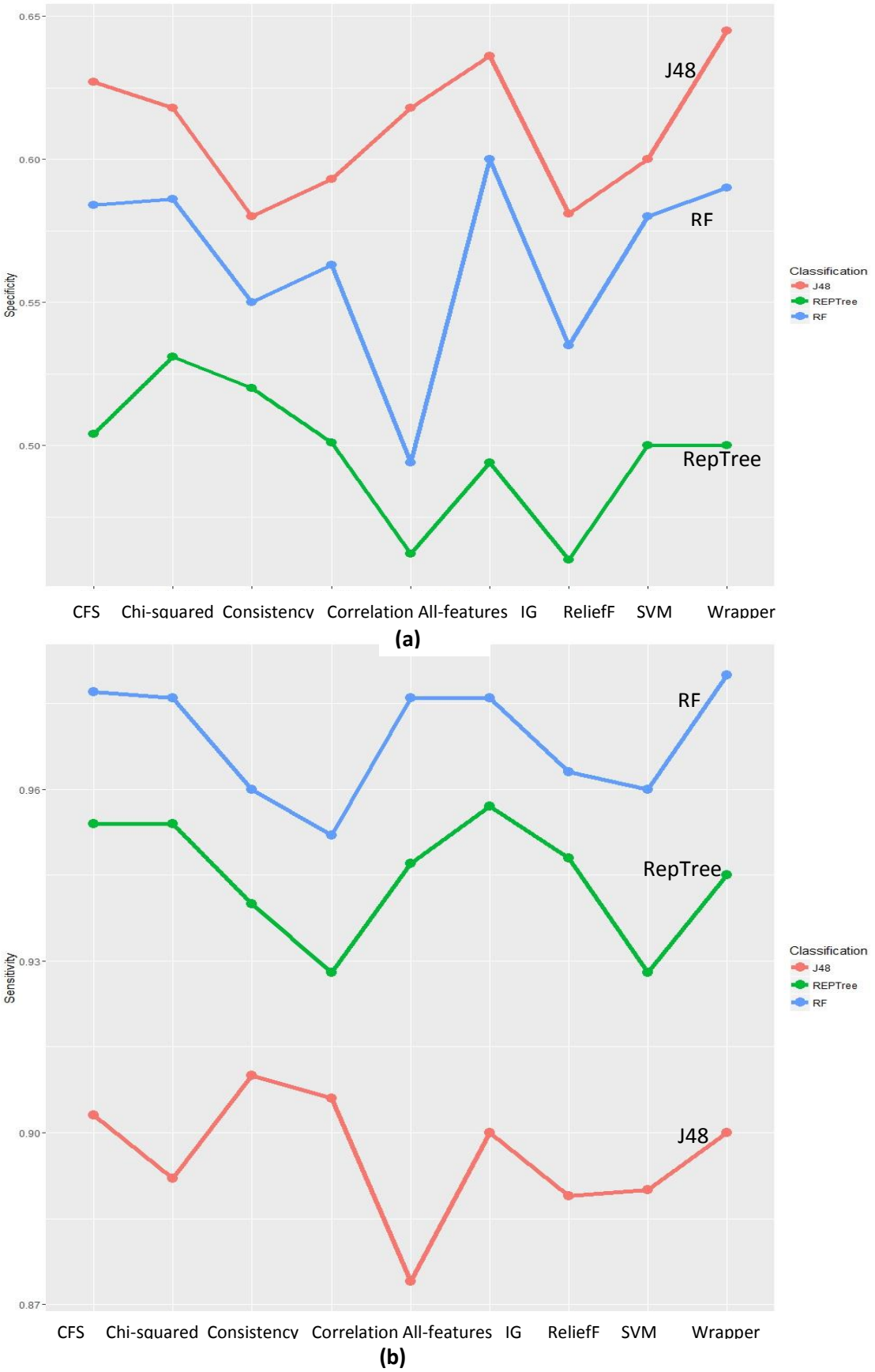

Figure 4-8. Performance measures using three different classification methods; (a) Specificity (b) Sensitivity

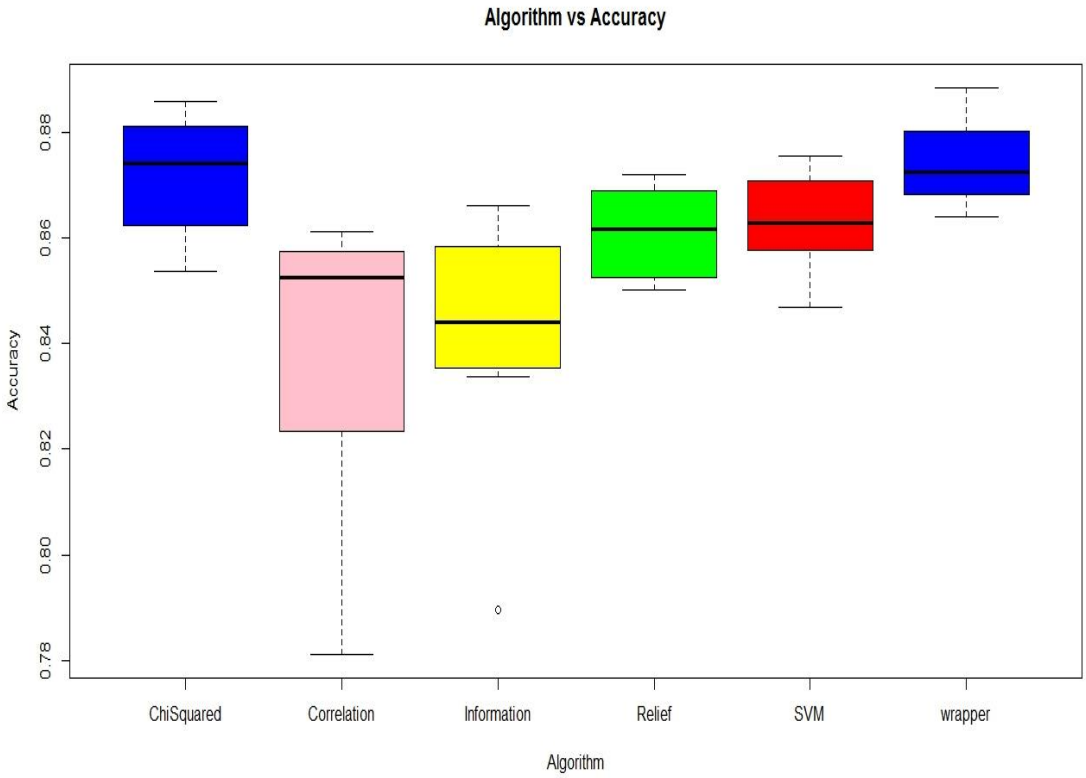

Figure 4-9 Box plot whiskers for different feature ranking methods in feature selection

4.4.1 The common features selected

a- Similarity results between methods

For the best 15 features selected, the intersection between the feature selections methods used for these variables is shown in table 4-6. As we can see, the first 14 variables appear in four methods or more, whereas only one feature appears in all methods, which is "*CT-proET1*". We eliminate variables that appear in only three or fewer methods, although some of them appear in the best 15 features for the best two methods such as "*Aortic Velocity*" and "*Aortic Root*". Thus, the result obtained from the most common variables that appear above is less than for the wrapper and chi-square methods when we use 15 variables to classify, as shown in table 4-7. This is because, as noted above, we eliminate the features that appear in three or fewer methods. By doing this, we eliminate features that appear in chi-square and wrapper only. Therefore, features like "*Aortic Velocity"* and

"Aortic Root" are more important and have stronger relevance than other variables that appear in more than four methods, such as "*Age"* and "*Albumin"*.

b- The set of features common to 3 methods and more.

The subset has expanded common features selected by adding features that appear in three methods and more. Table 4-8 shows the results for employing 25 variables which are picked from the table of common variables that appear in the best 16 features from the method used. As we can see, compared with the results in Table 44, the outcomes are improved but still less than those of the wrapper and chi-square methods. For 15 variables chosen by the wrapper and chi-square methods compared by the set of common variables, we found that all 15 features chosen by wrapper and chi-square are the same, except *Aortic Root* and *Aortic Velocity*. Almost 90% of features selected in this subset are in the 15 features subset of wrapper and chi-square.

Table 4-6: The Most Common Features and the Number of Methods

Features	No. of Methods	Features	No. of Methods
CT-proET1	*7*	*fev1*	*3*
Iron(umol/L)	*6*	*FEV1(L)*	*3*
LVEDD(HgtInd)	*6*	*FVC*	*3*
age	*5*	*FVC(L)*	*3*
MCV(fL)	*5*	*Glucose*	*2*
MR-proANP	*5*	*WhiteCellCoun*	*2*
LeftAtrium(BSAInd)	*5*	*AorticRoot(cm)*	*2*
MR-proADM	*5*	*AorticVelocity*	*2*
Albumin	*4*	*DiastolicBP(mmHg)*	*2*
Ferritin	*4*	*Phosphate(mmol/L)*	*1*
UricAcid	*4*	*TSH*	*1*
LVEDD(cm)	*4*	*LeftAtrium(cm)*	*1*
Urea	*4*	*AlkalinePhophatase*	*1*
VitaminB12	*4*	*BSA(m^2)*	*1*
CRP	*3*	*FEV1 Predicted(L)*	*1*
Chloride	*3*	*FVC-Predicted(L)*	*1*
CT-proAVP	*3*	*FVC-Predicted(L)*	*1*
E	*3*	*PulseBP*	*1*
HCT	*3*	*Haemoglobin*	*1*
Triglycerides	*3*	*PCT*	*1*
Creatinine()	*3*		

Table 4-7: Performance result of different classification methods for 14 features with common variables that appear in four methods and more

No. of Variables	Classification Method	Accuracy	Sensitivity	Specificity	PPV
14	J48	83.14%	90%	62%	87%
14	Random Forest	87.20%	97%	58%	87%
14	REPTree	84.80%	94%	56%	86%

Table 4-8: Performance result of different classification methods for 25 features with common variables that appear in three methods and more

No. of Variables	Classification Method	Accuracy	Sensitivity	Specificity	PPV
25	J48	83.25%	91%	60%	87%
25	Random Forest	87.48%	96.9%	59%	87.6%
25	REPTree	83.60%	94%	52%	85.5%

4.5 Conclusion

Feature selection is a tool that reduces high dimensionality in order to reduce computational complexity and ensure accuracy in predictive analysis. All feature selection methods aim to remove irrelevant and redundant variables while selecting the most relevant features. The only criterion is that the predictive performance of selected features is equal to the original set of features present in the dataset.

This chapter investigated several feature selection methods and used a representative set of classification methods for evaluating the features selected. These methods enabled the identification of a core set of features, from this dataset. Feature selection techniques include a wrapper method, an embedded-SVM method, together with six filter methods. In the category of filter methods were two which based on the creation of sets of subsets of features; the Cfs and consistency approaches, and four ranking methods; ReliefF, Information gain, chi-square, and Correlation attributes. In filter methods, selection procedures are independent of the learning algorithm. Once the ranking has been obtained, the best features are selected to evaluate their performance. In addition, subsets are also created and the aim is to find the subset that obtains the best performance, for

example, Cfs or Consistency approaches. In contrast, in wrapper the evaluation of features is a part of the learning algorithm used to train the model itself. Thus, thewrapper method is generally more computationally intensive. However, in the embedded method, a top subset of features is built into the classifier construction.

We found that wrapper and chi-square are the two methods with the highest classification performance. Similarly, embedded-SVM and ReliefF give good performance with 15 features. With all these methods, increasing the number of variables does not improve the predictive performance. In contrast, the Information gain and Correlation methods perform poorly with fewer features, but there is a performance enhancement by adding more features to the selected subgroup. Amongst the subset methods, we used Cfs and Consistency, which selected groups of 23 and 11 variables, respectively. Cfs performed better than Consistency, embedded, and ReliefF. We found that all the feature selection approaches improved the classification performance compared with the full dataset, but only if we chose an appropriate number of features.

In a large dataset relating to patients with chronic heart failure, the wrapper method was best for reducing the complexity of variables when trying to construct prognostic models. The chi-square method was better when computational power was limited. Both methods selected approximately 25% of the variables in the main dataset.

Chapter 5. Effect of Class Imbalance on Feature Selection and Classification

5.1 Introduction

The third challenge of the clinical datasets, as discussed in Chapter 1, is unbalanced class distribution. Class imbalance means that one class (majority) is represented by a larger number of data points than another (minority) one in binary classification (Poolsawad et al, 2014a). Data mining is generally prone to unbalanced data because most standard algorithms expect balanced class distributions, so learning classification techniques achieve poorly with class imbalance (He & Garcia, 2009). Class imbalance is often present in clinical data because the data collected do not take into consideration the class label; the data collected from "Alive" patients will be more than that from "Dead" patients. Thus, class imbalance is critical for real-world applications such as medical diagnosis, pattern recognition, and fraud detection (Cao et al, 2016). Methods used to manipulate class imbalance can be categorised into pre-processing approaches and algorithmic approaches. The pre-processing approaches are the handling obtained by resampling the class distribution, by under-sampling the majority class, or over-sampling the minority class in the training set (He & Garcia, 2009; Kirshners et al, 2017). In contrast, boosting is an example of an algorithmic approach that recalculates weights with each iteration to place different weights on the training examples (Mahdiyah et al, 2015).

In this chapter, we will show the effect of class imbalance on the training data; also we will find the outcome of this issue on selection of features using feature selection methods.

5.2 Class Imbalance

In a binary classification, the data are divided into two parts, where each part belongs to a class label. The part with more data points for a class label is named positive class instances or majority; the smaller part is named as negative class instances or minority. It is essential in processing a large volume of data to come up with small random samples,

rather than to process all data (Witten & Frank, 2011). Random sampling means each instance has an equal chance of being included in the dataset; this could be with replacement, or without replacement. Sampling with replacement refers to selecting an instance more than once; this is used for the bootstrap algorithm. On the other hand, sampling without replacement, for each instance selected, simply rejects the second copy. Bootstrapping is a mechanism whereby every time a sample is taken from a dataset to form a test or training set, it is drawn without replacement (Witten & Frank, 2011).

The imbalance ratio (IR) is measured by dividing the number of samples of the minority class by the number of samples of the majority class, as:

$$IR = \frac{Number\ of\ negative\ class\ instances}{Number\ of\ positive\ class\ instances} \tag{5.1}$$

In the research dataset, the class imbalance ratio is 1 to 3 for the Dead to Alive classes, and the imbalance ratio which is 33%, as shown in table 5-1.

Table 5-1: Target Classes Distribution on Hull-LifeLab

No. of features	60	
No. of data points	1750	
Target output	Mortality	
Class	Alive	Dead
Frequency	1313	437
Percentage	3	1
Imbalance ratio (IR)	0.33	

Several approaches can be used to balance class:

1- Data level: these methods create balanced data from the training dataset (Loyola-González et al, 2017). Methods in this level are called resampling methods, which can be divided into:

a- Resampling (external); basically it is a method that can balance classes by changing the size of data points to be trained. It provides a convenient and effective way to deal with imbalance related learning problems using standard classifiers because it alters the original training set rather than modifying the learning algorithm. The two approaches of resampling are oversampling and undersampling (Cao et al, 2016). Oversampling increases the frequency of samples in the training set for the minority class, as shown in figure 5-1(a). The drawback of oversampling is that it results in overfitting of the data due to it making exact copies of the minority class (Al-Shahib et al, 2005). Moreover, the size of the training set increases, which then increases the time taken to build a classifier. Undersampling decreases the frequency of samples in the training set for the majority class, as shown in figure 5-1(b). Undersampling can remove a lot of informative examples which could be useful in the development of the classifiers (Batuwita & Palade, 2010).

b- Active learning (internal) improves learner performance by selecting the more relevant data points to learn and leaving the irrelevant ones (Branco et al, 2015). This approach is like feature selection but selects data points, not features.

c- Weigh the data space: to avoid costly errors the training set distribution is modified using information concerning misclassification costs (Branco et al, 2015). It is a generalized cost-sensitive learning method to deal with unbalanced data distributions, where weights are assigned to every training instance based on users‘ needs.

2- Algorithm level (Cost-sensitive learning), by changing the classifier algorithm so it is more precise with the minority class (Loyola-González et al, 2017). The cost-sensitive method has to learn more characteristics of minority samples, in order to minimise higher error cost by considering higher costs for the misclassification of

positive class examples with respect to the negative class (Cao et al, 2016; SCI2S, 2007).

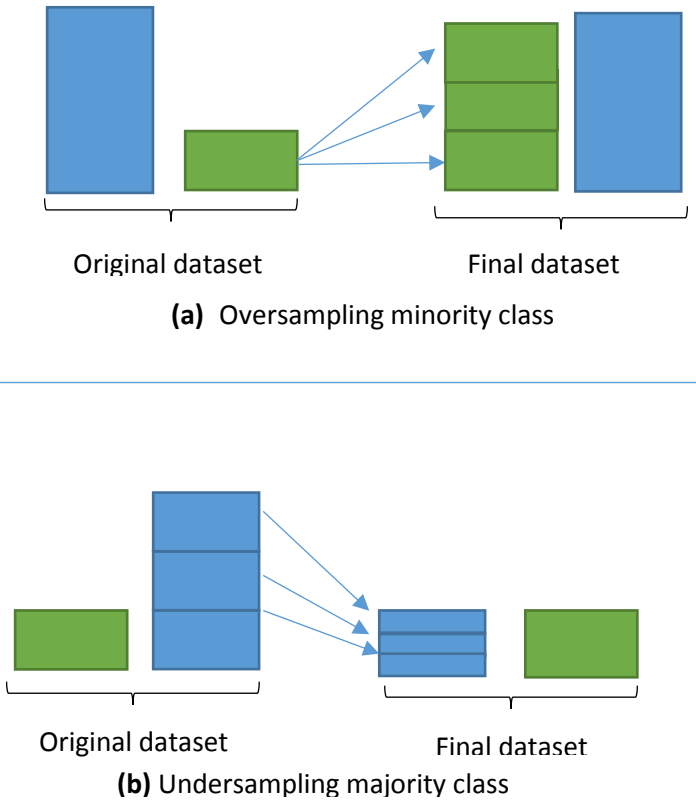

Figure 5-1 Oversampling increases the minority class by copying instances. Under-sampling removes instances from the majority class

5.3 Evaluation Measures

A confusion matrix is an object model for evaluating and assessing the performance of classification (as discussed in section 3.3.2). Based on the confusion matrix (Table 3-1); different measures are used to evaluate the classification. Each measures such a particular state; these measures include sensitivity, specificity, accuracy, PPV, and NPV. Sensitivity measures the percentage of cases correctly detected, in the major class; as this percentage increases, those incorrectly detected decreases. Specificity measures the percentage of cases correctly detected, in the minor class; this percentage increases when the number incorrectly assigned to the other class are decreased, as

illustrated in figure 5-2. Here, sensitivity and specificity are similar measures but for different class labels. Accuracy can be determined from sensitivity and specificity, because it measures the number of instances correctly identified in both class labels by a diagnostic test. PPV and NPV are influenced by the prevalence of the class that is being tested; PPV determines how to proceed after a patient gets a positive result, whereas NPV determines how to proceed after a patient gets a negative result.

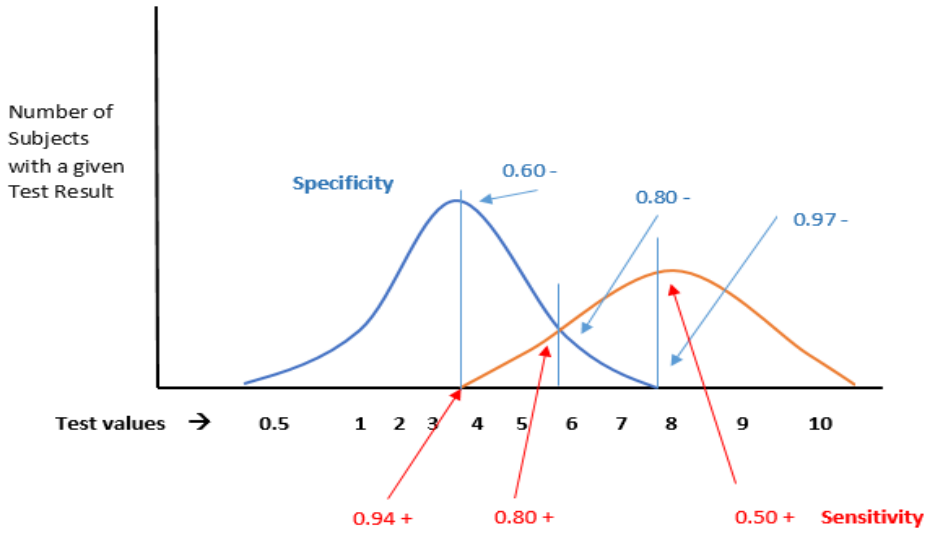

Figure 5-2 The trade-off between Sensitivity and Specificity

5.4 Class Balancing Techniques

Methods used in the thesis to handle class imbalance are:

1- **Resampling**: Produces a random subsample of a dataset using sampling either with replacement or without replacement. The original dataset must fit entirely in memory, and the number of instances in the generated dataset may be specified. The filter can be made to maintain the class distribution in the subsample, or to bias the class distribution toward a uniform distribution.

2- **Spread Subsampling**: Produces a random subsample of a dataset. The original dataset must fit entirely in memory. This filter allows specifying the maximum

"spread" between the rarest and most common class. For example, it may specify that there must be, at most, a 2:1 difference in class frequencies.

5.5 Analysis Results

In this section we investigate the class imbalance for the selected subsets obtained in Chapter 4 by employing the chi-squared, Information gain, ReliefF, embedded-SVM, and wrapper methods, then, we find the performance for the subsets using random forest and J48 methods for classification. The sets of data selected by different feature selection methods have improved the performance when applying RF and J8 as learning algorithms. Tables 5-2 and 5-3 show the results of applying the RF and J48 algorithms respectively, for different subsets sizes.

As we can see, the accuracy of data balanced using the resampling method has significant enhancement since resampling increases the minority class by copying instances, called oversampling. In contrast, using the spread subsample method does not show significant improvement and has almost the same results as the imbalanced classes. The most interesting aspect to note is the specificity results, which we can see doubled in some cases, such as with Information gain for 11 variables, and in all other cases, there are high increases. Specificity, also called true negative rate – section 3.3.2, measures the number of negatives correctly identified. After resampling, the classification of the new dataset has increased the number of true negatives (TN), and in many cases, this has doubled. TN means the true classification of "Dead" instances, where the "Dead" value refers to the minority class meaning that resampling can reinforce the minority class. The false positive (FP) measures the incorrect classification of the positive (Alive) class, which is the majority class; therefore, resampling decreases the error of the positives classified. The specificity of spread sub-sampling is equal to the specificity with class imbalance.

The specificity and accuracy with 11 and 23 variables are differentiated depending on the feature selection methods in class imbalance and balanced classes. However, by increasing the number of variables to 34 and 44, specificity and accuracy become equal for all feature selection methods, at around 79% and 94% respectively, using RF. The same indication is seen in PPV for both classification methods used, where with 44 features it outperforms (around 93%) all feature selection methods used. From the sensitivity and PPV for the balanced classes employed by RF and J48, respectively, it can be seen that the sensitivity becomes more than 99% or near for all feature selection methods with different numbers of variables. The increase in the percentage of sensitivity is because of the increase in the number of true positives (TP). Therefore, positive predicted values (PPV) are increased in the balanced classes in the same way as accuracy, with significant aggregate.

Table 5-2: Accuracy, Specificity, Sensitivity, and PPV Results in Implementing Random Forest Classification for Several Feature Selection Methods on Different Numbers of Subsets compared with class imbalance, and balanced classes using resampling and spread subsampling for the subsets from chapter 3

		Accuracy			**Specificity**			**Sensitivity**			**PPV**		
No. of Features	**Feature selection method**	**Imbalanced data**	**Resample Data**	**Spread Sub Sample**	**Imbalanced data**	**Resample Data**	**Spread Sub Sample**	**Imbalanced data**	**Resample Data**	**Spread Sub Sample**	**Imbalanced data**	**Resample Data**	**Spread Sub Sample**
11	Chi-squared	*88.00%*	97.02%	88.91%	*62.20%*	90.62%	62.47%	*96.80%*	99.16%	97.72%	*88.50%*	96.95%	88.67%
11	Information gain	*78.97%*	90.97%	79.31%	*37.30%*	73.23%	38.90%	*92.80%*	96.88%	92.76%	*81.60%*	91.58%	82.02%
11	Relief	*87.02%*	95.25%	88.11%	*57.90%*	85.35%	59.04%	*96.70%*	98.55%	97.79%	*87.30%*	95.29%	87.76%
11	Embedded-SVM	*87.25%*	94.91%	86.74%	*60.60%*	83.52%	58.58%	*96.10%*	98.71%	96.12%	*88.00%*	94.74%	87.64%
11	Wrapper	*88.22%*	95.42%	87.88%	*59.70%*	83.75%	57.89%	*97.70%*	99.31%	97.87%	*87.90%*	94.84%	87.47%
23	Chi-squared	*88.20%*	95.08%	87.88%	*59.30%*	82.84%	59.50%	*97.90%*	99.16%	97.33%	*87.80%*	94.55%	87.84%
23	Information gain	*83.37%*	92.91%	83.54%	*48.70%*	76.66%	49.20%	*94.90%*	98.32%	94.97%	*84.80%*	92.68%	84.89%
23	Relief	*86.74%*	94.11%	86.74%	*55.80%*	80.09%	56.06%	*97.00%*	98.78%	96.95%	*86.80%*	93.71%	86.89%
23	Embedded-SVM	*86.91%*	94.51%	86.90%	*58.40%*	81.46%	58.35%	*96.40%*	98.86%	96.42%	*87.40%*	94.13%	87.43%
23	Wrapper	*87.82%*	95.14%	87.77%	*58.40%*	82.84%	59.04%	*97.60%*	99.24%	97.33%	*87.60%*	94.56%	87.71%
34	Chi-squared	*86.28%*	94.57%	86.74%	*55.10%*	81.24%	54.69%	*96.60%*	99.01%	97.41%	*86.60%*	94.07%	86.59%
34	Information gain	*86.62%*	94.40%	85.94%	*57.20%*	80.78%	54.69%	*96.40%*	98.93%	96.43%	*87.10%*	93.93%	86.47%
34	Relief	*85.42%*	96.29%	85.54%	*54.20%*	86.50%	52.63%	*95.80%*	99.54%	96.50%	*86.30%*	95.68%	85.96%
34	Embedded-SVM	*86.28%*	94.05%	86.34%	*55.60%*	79.18%	55.38%	*96.50%*	99.01%	96.65%	*86.70%*	93.46%	86.68%
34	Wrapper	*87.20%*	94.97%	87.25%	*57.40%*	81.92%	57.21%	*97.10%*	99.31%	97.26%	*87.30%*	94.29%	87.23%
44	Chi-squared	*86.17%*	94.22%	85.82%	*53.30%*	79.86%	52.17%	*97.10%*	99.01%	97.03%	*86.20%*	93.66%	85.91%
44	Information gain	*86.05%*	94.00%	85.82%	*54.00%*	78.49%	53.09%	*96.70%*	99.16%	96.73%	*86.30%*	93.27%	86.10%
44	Relief	*85.08%*	94.11%	85.02%	*49.00%*	78.72%	48.74%	*97.10%*	99.24%	97.11%	*85.10%*	93.34%	85.06%
44	Embedded-SVM	*85.65%*	93.88%	85.82%	*51.30%*	78.49%	51.72%	*97.10%*	99.01%	97.18%	*84.90%*	92.65%	84.94%
44	Wrapper	*86.40%*	93.94%	86.57%	*53.10%*	78.72%	54.92%	*97.50%*	99.01%	97.11%	*86.20%*	93.32%	86.62%

Table 5-3: Accuracy, SPEC, SEN, and PPV Results in Implementing J48 Learning Algorithm for Several Feature Selection Methods on Different Numbers of Subsets, compared with imbalanced classes and balanced classes using resampling and spread subsampling methods for the subsets from Chapter 3

		Accuracy			Specificity			Sensitivity			PPV		
No. of Features	Feature selection method	Imbalanced class	Resample Data	Spread SubSample	Imbalanced data	Resample Data	Spread SubSample	Imbalanced Class	Resample Data	Spread Sub sample	Imbalanced class	Resample data	Spread SubSample
11	Chi-squared	85.02%	92.00%	84%	62.24%	89.39%	63.16%	92.61%	95.05%	91.01%	88.05%	94.33%	88.13%
11	Information gain	77.25%	86.17%	76.74%	37.76%	66.36%	33.41%	90.40%	92.76%	91.17%	81.36%	89.23%	80.44%
11	Relief	85.71%	90.85%	83.94%	57.89%	79.86%	58.35%	94.97%	94.59%	92.46%	87.14%	93.38%	86.96%
11	Embedded-SVM	82.11%	89.60%	83.25%	52.86%	75.51%	53.78%	91.85%	94.29%	93.07%	85.41%	92.04%	85.81%
11	Wrapper	84.05%	91.25%	83.94%	60.87%	78.72%	58.58%	91.77%	95.43%	92.38%	87.57%	93.09%	87.02%
23	Chi-squared	82.40%	91.08%	83.25%	60.64%	74.85%	62.24%	89.64%	94.59%	90.25%	87.25%	93.59%	87.78%
23	Information gain	79.02%	87.71%	77.60%	55.15%	72.77%	52.40%	86.98%	92.69%	85.99%	85.35%	91.09%	84.44%
23	Relief	82.00%	90.11%	81.77%	61.33%	77.35%	58.81%	88.88%	94.36%	89.41%	87.35%	92.60%	86.71%
23	Embedded-SVM	82.71%	91.25%	82.85%	60.41%	81.01%	62.70%	89.41%	94.67%	89.57%	87.15%	93.74%	87.83%
23	Wrapper	82.34%	90.68%	82.85%	61.56%	79.63%	62.01%	89.26%	94.36%	89.79%	87.46%	93.30%	87.66%
34	Chi-squared	82.68%	91.42%	82.05	60.41%	80.78%	63.16%	90.10%	94.97%	88.35%	87.24%	93.69%	87.81%
34	Information gain	82.91%	90.22%	81.65%	61.56%	81.24%	58.58%	90.02%	93.22%	89.34%	87.56%	93.72%	86.63%
34	Relief	81.82%	89.88%	80.68%	61.10%	78.49%	59.95%	88.73%	93.68%	87.59%	87.27%	92.90%	86.79%
34	Embedded-SVM	82.57%	90.91%	81.77%	60.87%	80.09%	62.70%	89.79%	94.52%	88.12%	87.33%	93.45%	87.65%
34	Wrapper	83.00%	91.54%	81.48%	60.87%	81.24%	64.07%	90.48%	94.97%	87.28%	87.42%	93.83%	87.95%
44	Chi-squared	81.48%	90.85%	82.28%	60.18%	80.32%	61.56%	88.58%	94.36%	89.19%	86.99%	93.51%	87.45%
44	Information gain	81.31%	90.22%	82.17%	61.33%	81.24%	60.87%	87.97%	93.22%	89.26%	87.24%	93.72%	87.27%
44	Relief	80.74%	90.68%	80.51%	59.95%	79.18%	59.04%	87.59%	94.52%	87.66%	86.79%	93.17%	86.54%
44	Embedded-SVM	82.57%	91.48%	81.94%	61.78%	81.01%	62.70%	89.49%	94.97%	88.35%	87.56%	93.76%	87.68%
44	Wrapper	81.25%	90.85%	82.05%	60.41%	78.95%	62.01%	88.18%	94.82%	88.73%	87.00%	93.12%	87.53%

5.5.1 Feature selection with balanced data

In the previous section, results using the features obtained from unbalanced data were presented. However, what needs to be looked into is the implicit assumption that features selected from the original data set would be the same with the balanced data, using the same feature selection algorithms. In fact after resampling data, the classification performance was not improved appreciably for the selected subsets. It can be seen from the tables 5-4 and 5-5; that with the resampled dataset, there was not much of a change in the performance of most of the feature selection algorithms, apart from the Information gain algorithm. It should be noted that this algorithm gave a better performance even when the number of features was small compared to the others. The performance of feature selection using the chi-square, wrapper, embedded, and ReliefF methods decreased. On the other hand, features obtained using the Information gain method provided an increase in accuracy from 78.97% to 86.90% for a subset of 11 attributes, when employed with balanced classes. With 44 variables, all methods including Information gain, had lower performance for balanced data than the original dataset. Implementing the learning algorithm J48 for subsets with 34 and 44 features yielded almost the same results, with and without resampling.

The explanation for these observations is that the true positive values (TP) of the confusion matrix show little variation between the classification of the balanced and imbalanced classes, whereas the changes come in false classification, especially false positive (FP). As seen in Table 5-6 the list features selected from the balanced dataset is slightly different from the list of features selected from the original data (refer to Table 4-5), while in some methods, including Cfs and Chi square, there are noticeable changes in the selected features between balanced and unbalanced classes. Thus, resampling by increasing the samples of the minority class will allow some features to be more relevant to predict the minority class rather than the majority class. Conversely undersampling the

majority class can discard data potentially important for the classification process. Hence there does not exist a control to remove patterns of the majority class. Resampling may cause overfitting, especially when using oversampling (Guo et al, 2008), which is due to selection of irrelevant features by the feature selection algorithm.

5.6 Training Datasets on Balance Data

Training the datasets selected by different feature selection methods on balanced data gave the outcomes shown in Table 5-7. The table illustrates the performance outcomes of training a feature subset on an RF classifier learned on balanced data, the selected features subset from Chapter 4. There is an impact of class imbalanced training data on the performance of a classifier, because in all feature selection methods the output is enhanced. The outcomes show that building a classifier on balanced data will produce better results, since the learning algorithm will build on accurate decision tree, since the samples have equal chance to represent the class label. The class corrected classification includes increased TP and TN compared with the classification outputs for class imbalance, whereas FN is noticeably decreased because now we have enough minority samples. In all methods the sensitivity because almost 97% because the number of FN has decreased. Also the specificity is enhanced, because the number of FP has deceased. In general, a class imbalance in the training data has an effect on the classifier performance. The minority class effects also the variability of the classifiers' performance, due to the random sampling from the population and random factors present in the training neural network.

Table 5-4: Classification of balanced classes using Random Forest after resampling data

		Accuracy		Specificity		Sensitivity		PPV	
No. of Features	Feature selection method	Imbalanced data	Resample before classification	Imbalanced data	Resample before classification	Imbalanced data	Resample before classification	Imbalanced data	Resample before classification
11	Chi-squared	88.00%	86.40%	62.20%	55.15%	96.80%	96.80%	88.50%	86.64%
11	Information gain	78.97%	86.90%	37.30%	56.75%	92.80%	96.95%	81.60%	87.07%
11	Relief	87.02%	80.97%	57.90%	39.36%	96.70%	94.82%	87.30%	82.45%
11	Embedded-SVM	87.25%	82.17%	60.60%	46.22%	96.10%	94.14%	88.00%	84.02%
11	Wrapper	88.22%	86.91%	59.70%	56.75%	97.70%	96.95%	87.90%	87.07%
23	Chi-squared	88.20%	84.80%	59.30%	49.43%	97.90%	96.57%	87.80%	85.16%
23	Information gain	83.37%	85.08%	48.70%	50.11%	94.90%	96.73%	84.80%	85.35%
23	Relief	86.74%	83.71%	55.80%	46.00%	97.00%	96.27%	86.80%	84.27%
23	Embedded-SVM	86.91%	83.20%	58.40%	47.83%	96.40%	94.97%	87.40%	84.54%
23	Wrapper	87.82%	86.17%	58.40%	52.86%	97.60%	97.26%	87.60%	86.11%
34	Chi-squared	86.28%	84.68%	55.10%	48.97%	96.60%	96.57%	86.60%	85.04%
34	Information gain	86.62%	84.40%	57.20%	48.51%	96.40%	96.34%	87.10%	84.90%
34	Relief	85.42%	83.08%	54.20%	45.31%	95.80%	95.66%	86.30%	84.01%
34	Embedded-SVM	86.28%	82.91%	55.60%	43.71%	96.50%	95.96%	86.70%	83.67%
34	Wrapper	87.20%	85.14%	57.40%	50.57%	97.10%	96.65%	87.30%	85.45%
44	Chi-squared	86.17%	83.31%	53.30%	44.85%	97.10%	96.12%	86.20%	83.97%
44	Information gain	86.05%	83.88%	54.00%	46.45%	96.70%	96.34%	86.30%	84.39%
44	Relief	85.08%	83.48%	49.00%	45.54%	97.10%	96.12%	85.10%	84.13%
44	Embedded-SVM	85.65%	82.74%	51.30%	44.85%	97.10%	95.35%	84.90%	83.86%
44	Wrapper	86.40%	83.02%	53.10%	42.56%	97.50%	96.50%	97.11%	83.47%

Table 5-5: Classification of balanced classes using J48 after resampling data

		Accuracy		Specificity		Sensitivity		PPV	
No. of Features	**Feature selection method**	**Imbalanced data**	**Resample before classificatio**	**Imbalanced data**	**Resample before classificatio**	**Imbalanced data**	**Resample before classificatio**	**Imbalanced data**	**Resample before classificatio**
11	Chi-squared	85.02%	*85.02%*	62.24%	*62.24%*	92.61%	*92.61%*	88.05%	*88.05%*
11	Information gain	77.25%	*84.11%*	37.76%	*59.95%*	90.40%	*92.16%*	81.36%	*87.36%*
11	Relief	85.71%	*79.42%*	57.89%	*34.10%*	94.97%	*94.52%*	87.14%	*81.16%*
11	Embedded-SVM	82.11%	*79.60%*	52.86%	*47.14%*	91.85%	*90.40%*	85.41%	*83.71%*
11	Wrapper	84.05%	*84.11%*	60.87%	*59.95%*	91.77%	*92.16%*	87.57%	*87.36%*
23	Chi-squared	82.40%	*82.91%*	60.64%	*63.16%*	89.64%	*89.49%*	87.25%	*87.95%*
23	Information gain	79.02%	*83.65%*	55.15%	*63.39%*	86.98%	*90.40%*	85.35%	*88.12%*
23	Relief	82.00%	*81.48%*	61.33%	*56.75%*	88.88%	*89.72%*	87.35%	*86.17%*
23	Embedded-SVM	82.71%	*82.22%*	60.41%	*61.10%*	89.41%	*89.26%*	87.15%	*87.33%*
23	Wrapper	82.34%	*83.37%*	61.56%	*62.24%*	89.26%	*90.40%*	87.46%	*87.80%*
34	Chi-squared	82.68%	*82.45%*	60.41%	*61.78%*	90.10%	*89.34%*	87.24%	*87.54%*
34	Information gain	82.91%	*82.97%*	61.56%	*61.78%*	90.02%	*90.02%*	87.56%	*87.62%*
34	Relief	81.82%	*81.31%*	61.10%	*64.07%*	88.73%	*87.05%*	87.27%	*87.92%*
34	Embedded-SVM	82.57%	*82.05%*	60.87%	*60.41%*	89.79%	*89.26%*	87.33%	*87.14%*
34	Wrapper	83.00%	*82.91%*	60.87%	*62.24%*	90.48%	*89.79%*	87.42%	*87.72%*
44	Chi-squared	81.48%	*81.48%*	60.18%	*61.10%*	88.58%	*88.27%*	86.99%	*87.21%*
44	Information gain	81.31%	*81.54%*	61.33%	*61.33%*	87.97%	*88.27%*	87.24%	*87.27%*
44	Relief	80.74%	*80.91%*	59.95%	*61.56%*	87.59%	*87.36%*	86.79%	*87.22%*
44	Embedded-SVM	82.57%	*81.94%*	61.78%	*60.87%*	89.49%	*88.96%*	87.56%	*87.23%*
44	Wrapper	81.25%	*81.54%*	60.41%	*61.10%*	88.18%	*88.35%*	87.00%	*87.22%*

Table 5-6: The Order List of Selecting 22 Features from Balanced Data Using Several Feature Selection Methods

	CFS	Chi square	wrapper	Svm	relief
1	Age(y)	Iron(umol/L)	LVEDD(HgtIndexed)	LeftAtrium(BSAIndexed)	PEFR(L)
2	Creatinine(umol/L)	Albumin(g/L)	MCV(fL)	MR-proADM	Age(yrs)
3	AdjCalcium(mmol/L)	LVEDD(HgtIndexed)	Ferritin(ug/L)	CT-proET1	LVEDD(HgtIndexed)
4	Albumin(g/L)	LVEDD(cm)	LVEDD(cm)	Urea(mmol/L)	FVCPredicted(L)
5	UricAcid(mmol/L)	TotalProtein(g/L)	UricAcid(mmol/L)	AlkalinePhophatase(iu/L)	LVEDD(cm)
6	Glucose(mmol/L)	Bicarbonate(mmol/L)	MR-proADM	MCV(fL)	Potassium(mmol/L)
7	Cholesterol(mmol/L)	E	CT-proET1	Iron(umol/L)	FEV1Predicted(L)
8	MCV(fL)	Ferritin(ug/L)	Hct(fraction)	LVEDD(HgtIndexed)	CT-proET1
9	Hct(fraction)	UricAcid(mmol/L)	AorticVelocity(m/s)	Creatinine(umol/L)	Hct(fraction)
10	Iron(umol/L)	Hct(fraction)	Iron(umol/L)	Chloride(mmol/L)	SystolicBP(mmHg)
11	VitaminB12(ng/L)	LeftAtrium(BSAIndexed)	E	Albumin(g/L)	Iron(umol/L)
12	Ferritin(ug/L)	MCV(fL)	Glucose(mmol/L)	WhiteCellCount(10^9/L)	FEV1(L)
13	CRP(mg/L)	AorticVelocity(m/s)	LeftAtrium(BSAIndexed)	BSA(m^2)	FEV1
14	TSH(mU/L)	CRP(mg/L)	LeftAtrium(HgtIndexed)	Age(yrs)	PulseBP(mmHg)
15	MR-proADM	Sodium(mmol/L)	Triglycerides(mmol/L)	FVC	Albumin(g/L)
16	AorticRoot(cm)	CT-proET1	VitaminB12(ng/L)	TSH(mU/L)	FVC(L)
17	LeftAtrium(BSAIndexed)	MR-proADM	AorticRoot(cm)	E	Urea(mmol/L)
18	AorticVelocity(m/s)	Triglycerides(mmol/L)	LeftAtrium(cm)	Ferritin(ug/L)	Pulse(Exam)(bpm)
19	E	PCT	Cholesterol(mmol/L)	FVCPredicted(L)	Chloride(mmol/L)
20	Weight(Exam)(kg)	AorticRoot(cm)	Urea(mmol/L)	Height(Exam)(m)	QT
21	SystolicBP(mmHg)	Bilirubin(umol/L)	PEFR(L)	BMI	LeftAtrium(BSAIndexed)
22	PEFR(L)	Chloride(mmol/L)	Chloride(mmol/L)	Platelets(10^9/L)	LeftAtrium(cm)

Table 5-7: Classification Outputs using RF method for training datasets on the balanced data

Feature selection methods	Number of features	TP	FN	FP	TN	Accuracy	sensitivity	Specificity
Chi square	11	1285	28	103	334	92.51%	97.87%	76.43%
Chi square	34	1282	31	150	287	89.65%	97.64%	65.68%
Chi square	44	1286	27	146	291	90.11%	97.94%	66.59%
Information gain	11	1245	68	168	269	86.51%	94.82%	61.56%
Information gain	34	1278	35	152	285	89.31%	97.33%	65.22%
Information gain	44	1278	35	138	299	90.11%	97.33%	68.42%
Relief F	11	1295	18	130	307	91.54%	98.63%	70.25%
Relief F	34	1287	36	153	284	89.77%	97.28%	64.99%
Relief F	44	1285	28	166	271	88.91%	97.87%	62.01%
SVM	11	1281	32	117	320	91.48%	97.56%	73.23%
SVM	34	1286	27	155	282	89.60%	97.94%	64.53%
SVM	44	1283	30	152	285	89.60%	97.72%	65.22%
Wrapper	11	1289	24	109	328	92.40%	98.17%	75.06%
Wrapper	34	1286	27	149	288	89.94%	97.94%	65.90%
Wrapper	44	1279	34	151	286	89.42%	97.41%	65.45%

5.7 Conclusion

In this chapter, the impact of unbalanced class distributions for a large dataset is demonstrated. The imbalance issue is presented in clinical data, where the number of people living has to be much more than the number of dead. Moreover, the implementation of five different feature selection methods was considered. The feature selection techniques used were information gain, chi-squared, ReliefF, embedded-SVM, and wrapper. Class imbalance was addressed by resampling and spread subsampling, and the learning tests used were the J48 and RF algorithms. The results presented in this chapter indicate that determining features from original data, and then training a classifier on balanced data, produces good results, often outperforming the original method. The outcomes show that resampling of the unbalanced classes generates a good enhancement in performance results for all measurements such as accuracy, specificity, sensitivity, and PPV, because there are enough samples for the minority class, which reduces the number classed incorrectly. In contrast, using spread sub-samples to balance the class distributions yields no different results compared with the classification for the unbalanced classes. Specificity had a great increase using the resampling method because it raised the number of the negative class (minority class). All other performance measures were improved using resampling to balance class, although the improvement was less compared with the specificity measure. The most performance improvement was for the information gain method, considering that the resampling added more information to the features, which increased the prediction results. Resampling data has greatly improved performance only with the information gain method. All other methods resulted in reduced performance or produced the same performance as the original data.

Another aspect considered was training the data on the model built on a class balance. The outcomes show that this will yield better results. Henceforth, a classification method built on balanced data will have an equal chance for all features, which leading to better

choices of the root. Also, classification algorithms show a bias towards the majority class, so the size of samples for both classes should be enough to contain the significant information to represent the data.

Chapter 6. Autoencoder Framework for Dimensionality Reduction and Classification

6.1 Introduction

An important aspect of feature selection is that at its very core lies a combinatorial problem, and all algorithms reduce the search space and make their selection efficient. Thus when looking at these algorithms, it is important to understand the degree of complexity they have. On the other hand extraction of features does not have this combinatorial problem, as they are often projections of higher dimensions on lower dimensions. However, when designing clinical decision support systems, feature extraction does not provide dimensions with original labels or meanings, which is a big drawback. However, if the process of feature extraction (and/or selection) are combined into one single process with learning classifiers, this problem of labels does not immediately become transparent. Thus the use of autoencoders and deep machine learning techniques are used in this context. Therefore, feature extraction would be a suitable solution for dimensionality reduction in terms of computational complexity. Deep learning is an automatic model that can be used for feature extraction because it is trained to do so. Autoencoder network is an example of deep machine learning that can compress the data mining methodologies by performing feature extraction and classification at the same time, so that result validation is done in the same model process.

Deep networks based on autoencoders are created by stacking pre-trained autoencoders layer by layer. An autoencoder is an automated model that can classify data, without explicitly providing information on how dimensions are reduced, or what projections are used to extract the data. There is a seamless integration of the two steps. This overcomes the immediate difficulty of lack of labels for the variables. Therefore it is an option that can reduce the data mining methodology steps and make it more efficient. As seen in the previous chapter, feature selection algorithms are iterative, and often the number of

iterations far exceeds the number of features selected. This is the combinatorial problem mentioned earlier.

6.2 Artificial Neural Network (ANN)

Neural networks are composed of a large number of artificial neurons. The neuron can have varying numbers of inputs from 1 to n. Each input to a neuron has its own weight associated with it, as illustrated in figure 6-2, by the small circle. The weight is simply a floating point number; the training process of the network concentrates a fine-tuning the weights. These inputs represented as $x_1, x_2, x_3 \ldots x_n$ are transformed into a single output O, via three basic elements (González, 2009):

- Two parameters, bias b and weights (w_1,w_2, ... w_n).
- The combination function h, which combines the input signals and the two parameters to produce a single input.
- A transfer function or activation function g that produces the output O by taking as argument the net input signal.

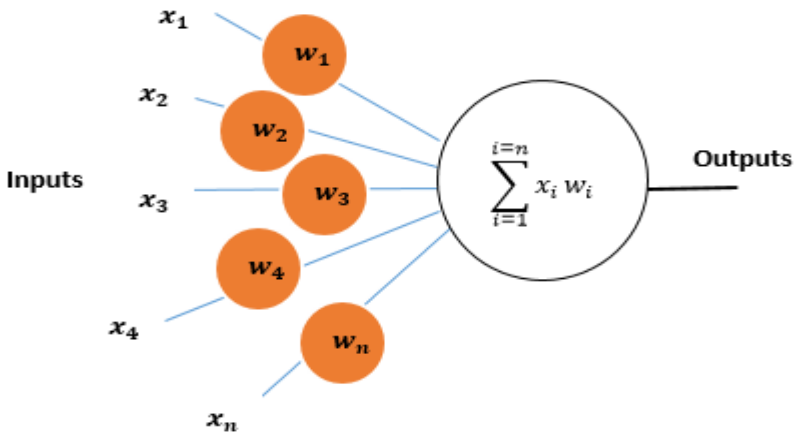

Figure 6-1 Neuron architecture

The matrix is representation introduced for the neural network, then the combination function h computes the inner product of the input vector and weights, which also includes a bias b, represented as a synaptic weight connected to a fixed input,

$$h = (\sum_{i=1}^{n} w_i\, x_i) + b \tag{6.1}$$

6.2.1 Activation Functions

The output is a function of the weighted sum $y = f(x)$ which is an activation function. There are different types of activation functions, see figure 6-3:

1- Linear function,

$$f(x) = \left(\sum w_i x_i\right) + b \tag{6.2}$$

2- Heaviside step function,

$$f(x) = \begin{cases} 1, & x \geq t \\ 0, & otherwise \end{cases} \tag{6.3}$$

Where t is a threshold

3- Sigmoid function,

$$f(wx) = \frac{1}{1 + e^{-\mathrm{w}x}} \tag{6.4}$$

4- Tanh function,

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2\mathrm{w}x}} - 1 \tag{6.5}$$

6.2.2 The Perceptron Function Space

The model perceptron mathematically may be viewed as a parameterized function space f from an input $\mathrm{X} \subset \mathbb{R}^n$ to an output $\mathrm{y} \subset \mathbb{R}$ where f consists of parameterized bias and weight, with $n + 1$ dimensions due to n inputs and 1 bias. Then the network input to the neuron is obtained first by linear combination of inputs and weights, in equation 6.1. By

adding the activation function g to the linear combination, the output of the neuron is given by

$$y = g(b + \sum_{i=1}^{n} w_i x_i) \tag{6.6}$$

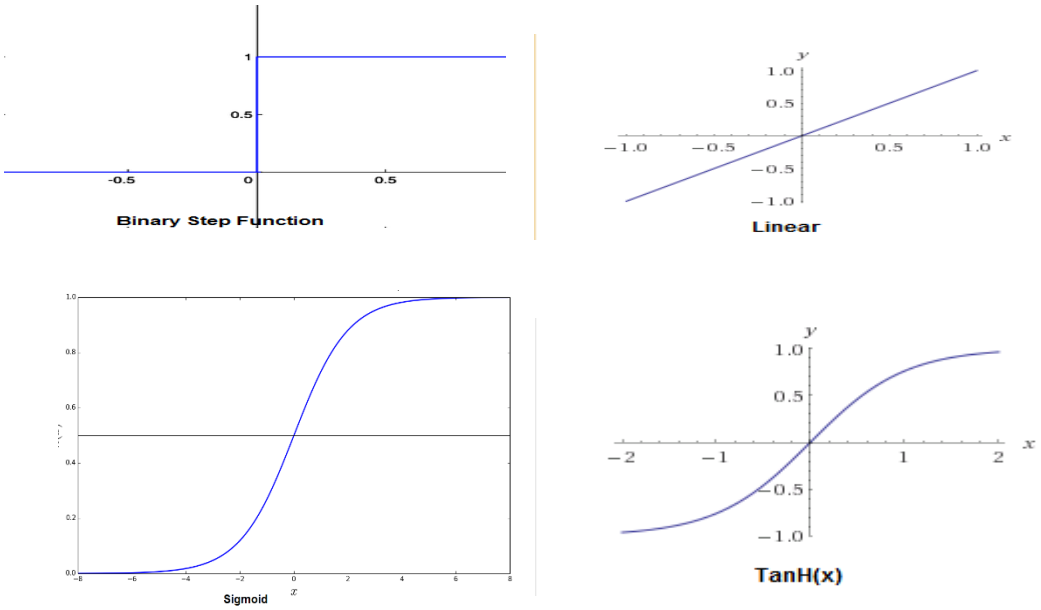

Figure 6-2 Activation function types

This is simply the architecture and mathematical notations for a single perceptron that can solve simple learning tasks, where g is a sigmoid function. However, connecting many neurons in a network architecture gives more power of neural computation to solve complex learning tasks.

6.3 Multilayer Neural Network (MLP)

6.3.1 Feed-forward architecture

A collection of neurons connected together in a network can be represented by a directed graph, where the nodes represent the neurons and arrows represent the links between

them. In a feedforward network, the neurons in each layer feed their output forward to the next layer until we get the final output from the neural network. The number of hidden layers within a feedforward network is differs, and the number of neurons in each layer can be adapted to the input and output nodes. The input layer redistributes input signals to all nodes in the first hidden layer. The output layer stimulates patterns from the hidden layer and creates the output patterns. Neurons in the hidden layer can detect hidden features; the hidden layer is so called because it 'hides' the desired output. Figure 6-4 shows a multilayer perceptron with n inputs, one hidden layer with m neurons, and k neurons at the output layer.

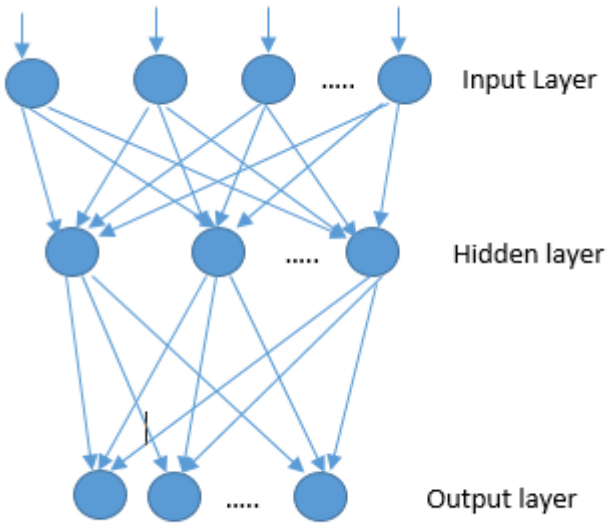

Figure 6-3 Feedforward neural network

6.3.2 Learning process

The learning step is the most important part of ANN creation and development for an appropriate performance (Bataineh, 2012). There are two categories of learning: supervised learning and unsupervised learning. In supervised learning, the output and input data are known, which can be used for classification and regression problems.

Unsupervised learning means that the output is not known, and the network is trained to groups the output in a proper way. This kind of training is used for clustering problems.

Training phases:

1- **Normalisation:** because of size and variance, the inputs of the training samples should be normalised. Normalisation is to decrease the variance and compress inputs to a small range (Bataineh, 2012). One of the ways it is done is by sigmoidal function to compress inputs to be handled by the network.

2- **Optimisation:** the network will be optimised and updated during the training process to reach the best prediction level through the training. The optimisation of the network is performed by the loss (minimization) function, which is the difference between the predicted outputs and the desired outputs. The cost function is the mean square error (MSE), defined as:

$$MSE = \sum_{i}^{n} (y_i - \hat{y}_i)^2 \tag{6.7}$$

where y is the predicted output and $\hat{y}$ the actual outputs. Updating neuron weights until reach the minimal value of MSE defined by the user.

6.4 Deep Neural Networks

The standard neural network consists of a 3-layer neural network involving one input layer, one hidden layer, and one output layer. In a deep neural network, there are multiple hidden layers to compute much more complex features (Ng et al, 2013). Although having more layers increases performance and is more beneficial, backpropagation and gradient descent are complex mathematically, as discussed in the previous section. Therefore, for deep layers, the chain becomes too long and derivatives are very hard to estimate consistently.

There are two conditions that make a neural network deep (Cho, 2014):

1- The network is extendable by adding more layers, and the activation of the neurons is shared. Consequently, the nodes in the added layer use the activations of the units in the existing lower layer for their own activations.
2- In every layer, it is possible to train the parameters.

6.4.1 Autoencoder Architecture

Autoencoder is a neural network that is trained to reproduce the inputs at the output layer, by learning a deep neural network. Autoencoder has a hidden layer h that describes a code used to represent the input. This network leads to generative modelling by theoretical connection with latent variable models. Moreover, it is based on comparing the activations of the network on the original input to the reconstructed output (Hinton & Salakhutdinov, 2006).

Consider a training set for n sample of inputs x where $x \in \mathbb{R}^d$ a real value and targets t. $D_n = \{(x^1, t^1), .., (x^n, t^n)\}$. The goal is to conjecture a new representation y, where $y \in \mathbb{R}^{d\prime}$. If x a binary representation $x \in [0,1]^d$ then $y \in [0,1]^{d'}$. We assume supervised learning.

Optimisation of the cost function is the basis of the learning process, which measures the difference between the inputs x at its reconstruction at the output y. The architecture of autoencoder is in two parts, consisting of the network encoder and decoder, where the encoder function $x = f(x)$ and the decoder $r = g(h)$ produces a reconstruction, figure 6-5.

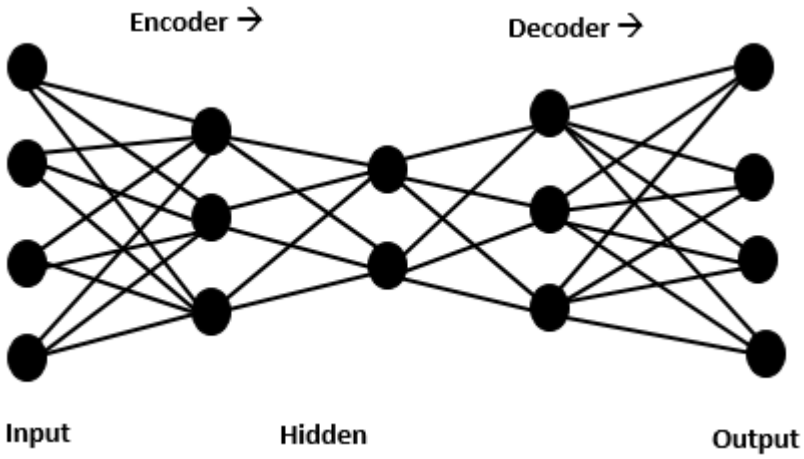

Figure 6-4 Autoencoder Architecture

The encoder maps the vector x to another vector $x \in \mathbb{R}^{D^{(1)}}$, when the input is a vector $x \in \mathbb{R}^{D_x}$, as follows (Inc, 2016)

$$z^{(1)} = h^{(1)}(W^{(1)}x + b^{(1)}) \tag{6.8}$$

Table 6-1 below shows the details of the encoder equation.

Table 6-1: Encoder Parameters

The subscript (1)	The first layer.
$h^{(1)}$: $\mathbb{R}^{D^{(1)}} \rightarrow \mathbb{R}^{D^{(1)}}$	A transfer function for the encoder.
$W^{(1)} \in \mathbb{R}^{D^{(1)} \times D_x}$	Weight matrix
$b^{(1)} \in \mathbb{R}^{D^{(1)}}$	Bias vector

The encoded representation z maps back into an estimate of the original input vector x, via decoder as follows,

$$\hat{x} = h^{(2)}(W^{(2)}x + b^{(2)}) \tag{6.9}$$

Table 6-2 below shows the details of the decoder equation.

Table 6-2: Decoder Parameters

The subscript (2)	The second layer.
$h^{(2)}: \mathbb{R}^{D_x} \rightarrow \mathbb{R}^{D_x}$	A transfer function for the decoder.
$W^{(2)} \in \mathbb{R}^{D_x \times D^{(1)}}$	Weight matrix
$b^{(2)} \in \mathbb{R}^{D_x}$	Bias vector

- **Autoencoder Structures**

1. Under completed autoencoder.

Where the h (hidden layers) have fewer dimension than x (input layer), then the training forces the autoencoder to obtain the most relevant features of the data (Deng et al, 2010). The learning process is described as

$$L(x, g(f(x)) \tag{6.10}$$

where L is the loses function. This structure is performed similarly to PCA when the decoder is linear and L is the mean squared error. However, it will perform more powerfully when the encoder and decoder functions are nonlinear.

2. Regularized Autoencoder.

A weakness of the under complete autoencoder is excess capacity of the encoder and decoder. Moreover, if the hidden layer has dimensions greater than or equal to the input layer, this will cause the over-complete case (Vincent et al, 2010). A regularised autoencoder is used in which encoder and decoder can learn, depending on the data distribution. Sparse and denoising autoencoder are two examples of Regularized Autoencoder.

2.a Sparse Autoencoder.

In this method, the training measure involves a sparsity penalty $\Omega(h)$ to the hidden layer h. Hence, the learning process will be described as

$$L\left(x, g\left(f(x)\right)\right) + \Omega(h) \tag{6.11}$$

where $g(h)$ is the decoder output. The penalty $\Omega(h)$ is simply like a term added to a feedforward network. The sparse autoencoder framework is like approximating maximum likelihood training of a generative model that has hidden features. Adding a regularizer to the cost function will boost the sparsity of an autoencoder (Deng et al, 2013). The mean squared error function is the cost function for training an autoencoder,

$$E = \frac{1}{N}\sum_{n=1}^{N}\sum_{K=1}^{K}(X_{kn} - \acute{X}_{kn})^2 + \lambda * \Omega_{weights} + \beta * \Omega_{sparsity} \tag{6.12}$$

where the first part is the mean squared error, $\Omega_{weights}$ is L_2 regularization, $\Omega_{sparsity}$ is sparsity regularization, λ is the coefficient for the L_2 regularization, and β is the coefficient for the sparsity regularization term. Then the function of the average output activation value of a neuron, which is the regulairzer, is given by

$$\widehat{p_\iota} = \frac{1}{n}\sum_{j=1}^{n} z_i^{(1)}(X_j) = \frac{1}{n}\sum_{i=1}^{n} h(w_i^{(1)T} X_j + b_i^{(1)}) \tag{6.13}$$

this is the average output activation measure of a neuron i , where n is the total number of training samples. Here the autoencoder can learn a representation by constraining the value of $\widehat{p_\iota}$ by adding a term to the cost function.

2.b Denoising Autoencoder.

This autoencoder network adds a penalty Ω to the cost function and changes the restriction error term as well (Lu et al, 2013). Denoising autoencoder must undo the corruption of x

$$L(x, g(f(\hat{x}))) \tag{6.14}$$

where $\hat{x}$ is a copy of input that has been corrupted by some noise.

6.4.2 Autoencoder Parameters

The proposed model uses particular parameters, such as the number of inputs and outputs and a number of hidden layers (Lukoševičius & Jaeger, 2009; Teoh et al, 2006):

1- A number of input neurons: as every input variable is treated by a single input neuron, the number of input neurons is determined by the size of the input vector.
2- A number of hidden layers: since the optimal number of hidden layers strongly depends on the data, then the number of hidden layers cannot be clearly defined, and may differ from case to case (Lawrence et al, 1997). In most cases, one or two hidden layers as enough, and increasing the number of those layers also increases the danger of overfitting (Chen, 2007). Therefore, the number of hidden layers cannot be predicted easily, and training of networks with different numbers of hidden layers may be required for the following comparison.
3- A number of hidden neurons: no rules exist for how to select the right number of hidden neurons. Experimentation phase is the most common method to determine the proper number. Nevertheless, it is also necessary to keep in mind that too big a number is demanding on resources, and too small a number may not reflect all the variety of the input data (Teoh et al, 2006). The network that performs well on the testing set and has the smallest number of hidden neurons is preferred (Kaastra & Boyd, 1996).
4- A number of output neurons. Most neural networks use only one output neuron for forecasting the next value.
5- A number of neural networks. Different networks have different strengths and weaknesses. The data decide how many layers a deep neural network needs (Cho, 2014).
6- Network parameters, see Table 6-3.

Table 6-3: Autoencoder Parameters to Build the Network

Parameter	**Value**	**Description**
L2 Weight Regularization	0.001	Controls the impact of an L2 regularizer for the weights of the network.
Sparsity Regulation	4	Controls the impact of a sparsity regularizer, which attempts to enforce a constraint on the sparsity of the output from the hidden layer.
Sparsity Proportion	0.05	Controls the sparsity of the output from the hidden layer.
Decoder Transfer Function	Logsig	$f(z) = \frac{1}{1 + e^{-z}}$
noF	60	Number of Features
noClass	2	Number of Classes

The L2 weight regularisation parameter is quite a small number. Increasing the values of the weights $w(l)$ and reducing the values of $z(l)$ can make the sparsity regulizer too small to train a sparse autoencoder (Olshausen & Field, 1997). Sparsity regulation is the second parameter, which controls the impact of a sparsity regularizer. Therefore, it attempts to enforce a constraint on the sparsity of the hidden layers' output. Adding a regularization term encourages sparsity by adding a large number, when the average activation value are not close in value between neuron i and its desired value (Olshausen & Field, 1997). Kullback-Leibler divergence can be a sparsity regularization term because it is a function for measuring the difference between two distributions;

$$\Omega_{sparsity} = \sum_{i=1}^{D^{(1)}} KL(\rho||\hat{\rho}_i) = \sum_{i=1}^{D^{(1)}} \rho \log(\frac{\rho}{\hat{\rho}_i}) + (1-\rho) \log(\frac{1-\rho}{1-\hat{\rho}_i}) \quad (6.\ 15)$$

Here, it takes a small value when ρ and $\hat{\rho}_i$ are close to each other, if they are equal it takes the value zero, otherwise it become larger. Thus minimizing of the cost function forces this term to be small. We will define the desired value of the average activation value using the *sparsity proportion*, which is the next parameter of the sparsity regularizer. It controls the sparsity of the output from the hidden layer. A low value for *sparsity proportion* usually leads to each neuron in the hidden layer "specializing" by only giving

a high output for a small number of training examples. For example, if the *sparsity proportion* is set to 0.05, this is equivalent to saying that each neuron in the hidden layer should have an average output of 0.05 over the training examples. This value must be between 0 and 1. The ideal value varies depending on the nature of the problem. The transfer function could be one of four types, as discussed in section 6.3.1; here we chose logistic function (Sigmoid) as a transfer function. NoF refers to the number of features as input, where in our implementation is 60 nodes with binary class that indicate the last parameter.

6.4.3 Autoencoder Algorithm

The pseudo code for the algorithm used is shown in figure 6-6. The algorithm is divided into four parts, starting by initializing the network, then defining the encoder function, after that defining the decoder function and lastly training the network. The initialize function will randomly weigh the neurons in each layer, as well as add a bias neuron for every neuron. In the encoder function, the nodes in the first visible layer will multiply the weights by the input values, and calculate the sigmoid function for every neuron. Then the decoding procedure reverses the calculation of the encoder to the next hidden layer. The training function takes the weights with such parameters as learning rate to train the network many times until it reaches a least MSE. In the main function, we will select the number of hidden layers and neurons in each hidden layer, then evaluate the network to decide the best network architecture.

Initialize

```
Initialize_AutoEncoder(int n_v, int n_h, double ** w, double * hb, double
                * vb) {
  for (int i = 0; i < n_hidden; i++)
          W[i] = new double[n_visible];
  double a = 1.0 / n_visible;
  for (int i = 0; i < n_hidden; i++) {
      for (int j = 0; j < n_visible; j++) {
          W[i][j] = uniform(-a, a); }}
  for (int i = 0; i <  n_hidden; i++)
          hbias[i] = 0;
  for (int i = 0; i < n_visible; i++)
          vbias[i] = 0;
```

Encoder

```
get_hidden_values(int * x, double * y) {
  for (int i = 0; i < n_hidden; i++) {
```

```
        y[i] = 0
      for (int j = 0; j < n_visible; j++) {
            y[i] += W[i][j] * x[j];
      }
          y[i] = hbias[i];
          y[i] = sigmoid(y[i]);
   } }
```

Decoder

```
get_hidden_values(int * y, double * z) {
   for (int i = 0; i < n_visible; i++) {
         z[i] = 0
       for (int j = 0; j < n_Hidden; j++) {
              z[i] += W[i][j] * y[j];
          }
        z[i] = hbias[i];
        z[i] = sigmoid(y[i]);
   } }
```

Train

```
get_hidden_values(int
            * x, double learning_rate lr, double corruption_level) {
      tilde_x = new [n_visible]
      //vbias
      For ( i = 0; i < n_visible; i++) {
          v_error[i] = x[i] – z[i];
          vbias[i] = vbias[i] + lr * v_error[i]/N;
      }
      //hbias
      For(i = 0; i < n_hidden; i++) {
          h_error[i] = 0;
          for (j = 0; j < n_visible; j++) {
              h_error[i] = h_error[i] + W[i][j] * v_error[j];
           }
          h_error[i] = h_error[i] * y[i] * (1 – y[i]);
           hbias[i] = hbias[i] + lr * h_error[i]/N;
      }
      //W
     For( i = 0; i < n_hidden; i++) {
        For (j = 0; j < n_visible; j++) {
            W[i][j] = W[i][j] + lr * (h_error[i] * tilde_x + v_error[j] *
y[i]/N
```

Figure 6-5 Pseudo Code to Programming Autoencoder Model

Autoencoder allows automatic generation of features, reducing human intervention in this process. Hence, the training is to perform the input-copying task that can be useful to extract meaningful features. This automatic feature extraction can be performed using an error function that encourages the model encoder to have specific

characteristics, including sparsity of the representation and robustness to noise. This autoencoder can be stacked to create a deep structure to increase the level of abstraction of the features learned. However, the lack of interpretability is the drawback to this model, which means it is difficult to understand how the features arrive at the prediction. The time complexity of the worst case of back-propagation model is $O(w\text{^}3)$; where w is the count of the weights in the network. This is because of the number of passes through each connection weight required to update that weight. The first pass is to compute the error at the output, the second pass is the backward propagation of the error to the lowest most weights, and the final pass is the update of each weight.

6.5 Analysis Results

6.5.1 One Hidden Layer

One hidden layer is the simplest autoencoder network. The autoencoder with one layer obtained the results shown in Table 6-4. We can see that the best performance and greatest accuracy can be obtained with 30 neurons in the hidden layer, although all the results are not very good and less than many classification methods. Therefore, one hidden layer is not enough for autoencoder to learn to extract the best features. We can see how the classification outcomes (TP, FN, FP, and TN) look. Although the prediction of positive class value is good, that of negative class value is poor. The positive class here is the "Alive" instances, whereas the negative is the "Dead" instances, referred to as the majority and minority classes, respectively. The positive samples are 1313 instances, equal to three-quarters (75%) of all the data, and the remaining, quarter (25%) are the 'Dead' instances. For example, in the case of a hidden layer with 30 neurons the TP is 1130 and FP are 183, which means 86% of instances were classified correctly for the positive class (majority class), which is the sensitivity

measure. However, in the negative class results, only 56% of the negative class are predicted correctly; this is the specificity measure. The observations of the outcomes for one hidden layer show that the model fails because it is difficult to classify negative values that have few samples. The sensitivity analysis to assess robustness has fair outcomes. Considering sensitivity tests the effective on positive individuals, the test is good. On the other hand, specificity measures how effectiveness the test is when used on negative individuals; for one hidden layer the outcome is bad, with less than 57% which means it is a random draw. Therefore, because sensitivity is used to show how effectively a prediction identifies cases who are "Alive", the result shows a good outcome, as higher sensitivity is better; but for the negative class, the test cannot identify the individuals correctly, since the outcome for specificity is low. The other measures related to the concepts of sensitivity and specificity are PPV and NPV respectively. PPV and NPV measure the group of people whose test results reflect their mortality status and are affected by mortality prevalence. PPV is the proportion of individuals who test positively and truly are "Alive"; it increases with a high prevalence of "Alive", NPV is the proportion of individuals who test negatively "Dead" and truly are not positive; the NPV decreases with a high prevalence of positive "Alive".

Table 6-4: Performance Results of the Autoencoder in one Hidden Layer

No. of Neurons in the Hidden Layer	**TP**	**FN**	**FP**	**TN**	**ACC**	**SEN**	**SPEC**	**PPV**
30	1130	188	183	249	78.8%	85.7%	57.6%	86.1%
20	1131	195	182	242	78.4%	85.2%	57.1%	86.2%
40	1133	195	180	242	78.1%	85.3%	57.3%	86.3%
10	1119	192	194	245	77.5%	85.3%	55.8%	85.2%
50	1131	183	182	254	78.7%	86.0%	58.2%	86.1%

For more explanation we will test the likelihood ratios (LR), seeing that LR as an expression of the accuracy test is used to evaluate how good an analytic test (CEBM,

2017). Whereas sensitivity and specificity are more automatically used in selecting the rule test to apply, LR shows more directly how we can understand positive and negative test results (Kohlberg & Hammer, 2014). In fact, it measures the power of a test to change the probability of pre-test into post-test. Therefore, likelihood ratios can be used to estimate how much a trial result will modify our prospect (McGee, 2002).

$$\text{Positive LR (LR+)} = \frac{TP/(TP+FN)}{FP/(TN+FP)} = \frac{sensitivity}{1-specificity} \quad (6.16)$$

$$\text{Negative LR (LR}-\text{)} = \frac{FN/(TP+FN)}{TN/(TN+FP)} = \frac{1-sensitivity}{specificity} \quad (6.17)$$

The interpretation of the likelihood ratios is defined by Table 6-5.

Table 6-5: Likelihood Rates and Their Interpretations

LR+	**Increase in likelihood**	**LR-**
>10	Cause large change	<0.1
5-10	Cause moderate changes	0.1-0.2
2-5	Cause small changes	0.2-0.5
1-2	Cause minimal changes	0.5-1
1	Uninformative, no change	1
Larger is better		Smaller is better

Thus, in our last example, positive LR and negative LR is shown in the following Table 6-6. We can see that positive LR and negative LR are around 2 and 0.25 respectively. Therefore, there is a tiny improvement in diagnosis based on our pre-test probability. In other words, an instance that is actually "Alive" is two times more likely to test positive, where the instance "Dead" has minimal increase in likelihood. Besides, an instance that is actually "Alive" is only 0.25 as likely to test negative for the instance "Dead", with a small decrease in likelihood ratios.

Table 6-6: Likelihood Results for the Autoencoder with One Hidden Layer

Number of neurons in the hidden Layer	LR+	LR-
30	2.02	0.25
20	1.99	0.26
40	2.00	0.26
10	1.93	0.26
50	2.06	0.24

To sum up, one hidden layer in an autoencoder network cannot get good results. Receiver Operation Characteristic (ROC) is a plot that simplifies the display of results. The ROC curve is a commonly used way to visualise the performance of a binary classifier. The curve shows the sensitivity against (1-specificity) (Pencina et al, 2008). When the curve climbs toward the top-left meaning, the model correctly predicted the cases. Figure 6-7 shows the ROC curve for the classification using autoencoder with 30 neurons in the hidden layer; as shown, the performance is acceptable but not very good. Although the plotting for true positive rate and false negative rate is far from the 45-degree diagonal, it is not close to the top-left border.

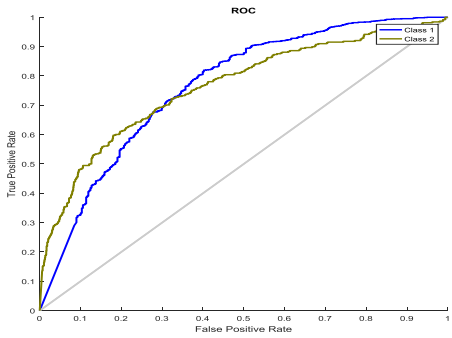

Figure 6-6 ROC curve for Autoencoder with 30 neurons in One Hidden Layer

6.5.2 Two Hidden Layers

With two hidden layers for the autoencoder network, it becomes more computationally costly, but can solve complex problems. Using two hidden layers to create an

autoencoder network obtained the outcomes illustrated in Table 6-7. The table shows some examples; the full results are shown in the appendix -III. As we can see, there are various options with two hidden layers. The first hidden layer may have more neurons than the second hidden layer or vice versa, or the size of both hidden layers are equal. When the first hidden layer has 50 neurons; the best results were obtained with 50 or 45 neurons in the second hidden layer; the accuracies obtained were 84.11% and 83.57% respectively. Sensitivity was almost equal for all cases with 50 neurons in the first hidden layer, around 86%. The difference between the two layer and one layer models is evident in the increase in true positive (TP) results. Another option is to have the size of the first hidden layer as 40 neurons, with different sizes of the second hidden layer. The outcomes show an enhancement with accuracy reaching 84.63% with 40 and 35 neurons for the first and second hidden layers respectively. Thus, the number of true negative (TN) is slightly increased, which refers to a boost in the minority class results. As we see, there is more differentiation between the results when we change the number of neurons in the second hidden layer. The third case has 35 neurons in the first hidden layer and different sizes for the second hidden. The best results were obtained with 35 neurons in the first hidden layer, while the second hidden layer had either 25 or 45 neurons. Accuracy reached 84.68%, and there was also an improvement in specificity, which was more than 72.5%, but no noticeable changes in sensitivity, since the increase was for the majority class more than the minority class.

For a number of neurons in a hidden layer with half the number of inputs, 30 nodes, the results show enhancement, especially when the second hidden layer has 40 neurons, where the accuracy reached 84.94%. Therefore, the majority class showed a large increase in TP. Moreover, there was an effect on the other performance measures such as specificity, and PPV, while sensitivity stayed around 85%. Next, we make the first hidden layer size equal to 25 neurons combined with several sizes of the second hidden layer. In

this case, the performance declines as the range of accuracy are from 82.3% to 83.98% and there is noticeable variation in specificity and PPV measures, while there is a little variance for sensitivity. The best outcomes for this case are when the number of neuron in the second hidden layer is either 15 or 10. Then, using 20 neurons as the first hidden layer and changing the size of the second hidden layer will decrease performance more than the other options. The network architectures that achieve the best results are in bold. The ROC curves when using [30, 40] and [35, 30] neurons in the hidden layers are shown in figure 6-8. As we see, the curve is enhanced and becomes closer to the left border compared to using one hidden layer.

Table 6-7: Performance Outcomes using Autoencoder with Two Hidden Layers

No. of Neurons in the Hidden Layers	**TP**	**FN**	**FP**	**TN**	**ACC**	**SEN**	**SPEC**	**PPV**
30 40	1241	217	72	220	84.94%	85.12%	75.34%	94.52%
50 50	1217	207	96	230	84.11%	85.46%	70.55%	92.69%
40 30	1219	208	94	229	84.19%	85.42%	70.90%	92.84%
40 35	1233	213	80	224	84.63%	85.27%	73.68%	93.91%
35 10	1194	198	119	239	83.32%	85.78%	66.76%	90.94%
35 35	1211	200	102	237	83.63%	85.83%	69.91%	92.23%
35 45	1228	212	85	225	84.51%	85.28%	72.58%	93.53%
30 25	1216	204	97	233	83.92%	85.63%	70.61%	92.61%
30 30	1183	187	130	250	82.55%	86.35%	65.79%	90.10%
30 40	1241	217	72	220	84.94%	85.12%	75.34%	94.52%
30 45	1240	214	79	223	84.76%	85.28%	73.84%	94.01%
25 20	1171	182	142	255	82.12%	86.55%	64.23%	89.19%
25 40	1206	198	107	239	83.46%	85.90%	69.08%	91.85%
20 20	1187	191	126	246	82.83%	86.14%	66.13%	90.40%
20 40	1182	190	131	247	82.72%	86.15%	65.34%	90.02%
15 20	1132	184	181	253	81.73%	86.02%	58.29%	86.21%
15 25	1129	183	184	254	81.63%	86.05%	57.99%	85.99%
10 15	1070	163	243	274	79.61%	86.78%	53.00%	81.49%
10 60	1106	176	207	261	80.91%	86.27%	55.77%	84.23%

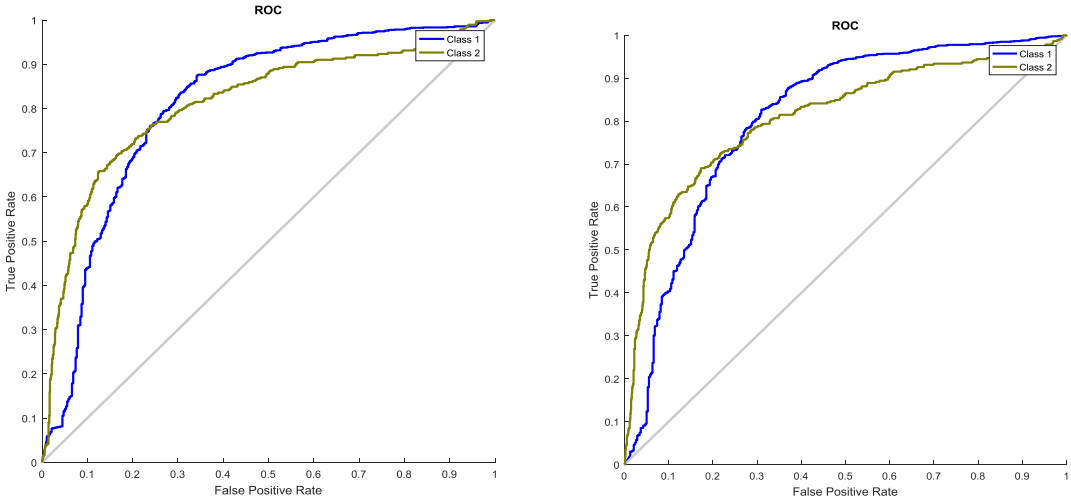

Figure 6-7 ROC curve for the two Hidden Layers , (a) Hidden Layers [30,40] ,(b) Hidden Layers [35,30]

The diagnosis test in Table 6-8 illustrates the positive LR and negative LR for the best four models. The results are better than those for one hidden layer, shown previously. The positive LR becomes greater than 3, which is good and indicates small changes rather than the minimal changes in the previous test. Moreover, the negative LR goes to 0.20, which indicates small changes, as in the previous test, but the value is increased.

Table 6-8: Likelihood Results for Autoencoder with two Hidden Layers

Hidden Layers	**LR+**	**LR-**
30 40	3.45	0.20
30 45	3.26	0.20
40 35	3.24	0.20
35 45	3.11	0.20

6.5.3 Three Hidden Layers

As we can see in Table 6-9 with three hidden layers, autoencoder will classify all the major class correctly and all the minor class incorrectly, which is unsatisfactory. Thus, more than two layers caused overfitting of the training data.

Table 6-9: Performance Results for Autoencoder with Three Hidden Layer

No. of Neurons in the Hidden Layers	**ACC**	**TP**	**FN**	**FP**	**TN**	**SEN**	**SPEC**	**PPV**
50, 40, 30	75%	1313	437	0	0	75%		100%
40, 30, 20	75%	1313	437	0	0	75%		100%

6.5.4 Autoencoder with class balance

As shown in the previous sections, autoencoder is affected by class imbalance. The results indicate that the model has the best results with the majority class label, while the overall outcomes are decreased, as the minority class label has poor results. Thus, overall performance is strong but for the minority class it is very poor. This could happen because the classifiers assume that unseen data points are drawn from the same distribution as the training set (Wasikowski & Chen, 2010). Also, as autoencoder tries to optimize the MSE on the training data, it generalizes very well to overall predictive accuracy on the training set. The class imbalance can be balanced by different techniques including random resampling, as seen in the previous chapter. By random resampling, the samples could be decreased or increased to make a better balance between two classes. The outcomes of an autoencoder model with two hidden layers applied on a resampled dataset is shown in Table 6-10. As illustrated, the results for the balanced class show reduced sensitivity, which is almost the same as the sensitivity of classification will unbalanced data since the TN samples have increased. However TP of the balanced class has decreased, which affect the performance outcomes including accuracy and specificity. The resampled dataset did not improve the performance of autoencoder classification because increasing the samples of the minority class will change the majority class, whilst the resampling will not be an accurate representation of the population, thus resulting in inaccurate results

with the actual test data set. Although oversampling will not lead to loss of information, this will increase the likelihood of overfitting since it replicates the minority class samples. Resampling class imbalance will to boost some features that may be recognised as irrelevant or redundant by the network. This will lead to such features being extracted, instead of others that are relevant, and this is why the interpretation of the outcome changes from unbalanced to balanced data.

Table 6-10: Performance Results for Autoencoder with Balanced data

Neurons in hidden layers	TP	FN	FP	TN	Accuracy	Sensitivity	Specificity	PPV
35 30	1060	189	221	279	76.6%	84.87%	55.80%	82.75%
40 20	1060	195	221	273	76.2%	84.46%	55.26%	82.75%
30 10	1061	176	220	292	77.4%	85.77%	57.03%	82.83%
30 30	1072	187	209	281	77.4%	85.15%	57.35%	83.68%
30 15	1066	182	215	286	77.3%	85.42%	57.09%	83.22%
25 15	1067	184	214	284	77.2%	85.29%	57.03%	83.29%
50 25	1059	185	222	283	76.7%	85.13%	56.04%	82.67%

- **Evaluate the effect of class imbalance in training data on performance:**

Training an autoencoder network on balanced data was used to assess the impact of class unbalanced training data on the performance of a classifier. However, there are other factors that affect the performance of neural network based classifiers, such as the number of features, the correlation between features, and training sample size. The network used is a feedforward neural network, as training this network obtains the best classification performance for finding the best set of network weights (Mazurowski et al, 2008). Next, backpropagation finds the MSE in weight space to train the classifier. As we can see in Table 6-11, the class correctly classification includes increased TP and TN compared with Table 6-7. The table shows performance improvement compared to the outcomes of the autoencoder trained on unbalanced data. Although FP has increased in the results

obtained from the network trained on balanced data the outcomes are better because the increase in TP is more than FP; also as seen the FN has dropped from 217 to 120, which contributes to the good results. The trained network enhances the outcome of the minority class, since having more samples increases their outcomes. In general, a class imbalance in the training data has an effect on the autoencoder classifier's performance. Hence, the minor class also affected the variability of the classifier's performance due to the random sampling from the population and random factors present in the training neural network.

Table 6-11: Performance results for Autoencoder training on balance data network

No. of neurons in the hidden layers	TP	FN	FP	TN	accuracy	sensitivity	specificity	PPV
30 25	1301	120	113	216	86.34%	91.44%	62.94%	91.89%
30 35	1308	126	119	197	86.00%	91.21%	62.34%	91.66%
30 40	1301	129	123	197	85.60%	90.98%	61.56%	91.36%
40 20	1305	125	120	200	86.00%	91.26%	62.50%	91.58%

6.6 Conclusion

This chapter has presented experiments on the autoencoder model using the heart failure dataset. The model consists of one input layer, several hidden layers, and one output layer. The difference between autoencoder and other neural network models is that in autoencoder the output is equal to the input data. Moreover, autoencoder has a bottleneck in the middle of the network, which is the extracted features reduced from the input data.

We have demonstrated how to implement the autoencoder network to learn meaningful features, called feature extraction. The network has various parameters that identify the form of the network and its output. The main parameters are the transfer function, learning rate, number of hidden layers, and number of neurons in each hidden layer. The results show that two hidden layers are the best choice to have good results. One hidden layer

did not give good performance outcomes, while three hidden layers caused overfitting and the model did not work properly. The best performance was obtained from the network with two hidden layers [30, 40].

The analysis of the results shows that because the dataset has unbalanced classes, the model is suitable for the majority class and unfair for the minority class. The majority class, which is the positive state in our dataset, has 75% of the data, which is enough to extract the features that can correctly classify the data. On the other hand, the minority class, which is the negative state, has 25% of the data and the model cannot predict cases correctly because there are not enough samples. In addition, autoencoder shows improve performance when learning on balanced data, since a small number of training examples also has a detrimental effect on the average performance.

Chapter 7. Conclusion and Future Research

7.1 Introduction

The purpose of this research was to investigate clinical data mining issues and to build a machine learning model that can compress the data mining methodology; by classifying data and extracting features in one step. Problems that affect clinical datasets are missing values, high dimensionality, and class imbalance. Imputing missing values is important as a pre-processing step for all frameworks used for mining data. High dimensionality can be reduced either by extracting or selecting features. We show that selecting features is highly complex, since many iterations and evaluation steps are needed to assess each group of features. Addressing all data mining issues can improve prediction performance and reduce the complexity of data. The research was motivated by these issues to investigate machine learning models to impute missing values, reduce dimensionality, classify data and enhance prediction performance.

The framework started by cleaning data by discarding data that has too many missing values. Then, missing values were imputed using several machine learning methods, and the imputation evaluated using different techniques to show the performance and analyse the weaknesses and strengths of these tools. Then the relevant features were selected from the large dataset to enhance the accuracy of prediction. Although this technique shows high complexity due to the many iterations used for each feature selection method, it determines the significant features. Next, class imbalance was investigated and the dataset was balanced to select features and classify the balanced data. Eventually, a model that can compress data mining methodology by extracting the features and classifying the data were proposed using an autoencoder, in order to investigate this model and how to implement it for a clinical dataset. The autoencoder model we used has reduced the

complexity by using feature extraction and classification; although the model is affected by class imbalance, it still reduces the complexity and obtains good results.

The thesis successfully addressed the thesis objectives and proposed techniques to improve data imputation, feature selection and class prediction for the clinical dataset. This chapter presents the conclusions for the research objectives initially presented in chapter 1, and gives a summary and some suggestions for future research.

7.2 Contributions of the research

The thesis contributes to the area of data mining and machine learning. Specifically, it introduces novel thinking and techniques to the fields of missing values, feature selection, data imbalance, and clinical data. Regarding missing values, it was shown that techniques based on class separation will outperform other techniques in predictive ability. Next, we found the major variables in the heart failure dataset that can improve predictive modelling and enhance classification performance. Moreover, we analysed the best number of features to be selected by each method. For the balanced data, we found that training data on a balanced model results in improved. Finally, the autoencoder model was used to classify clinical data by extracting features.

The thesis explores the challenges of real clinical data through machine learning methods to investigate a suitable approach for imputing, feature selection, and classification. Furthermore, it used an autoencoder model to classify data by extracting significant features. In evaluating the models used, several decision trees were applied, including RF, J48, and REPTree. The problems posed by real-life clinical datasets were introduced in chapter 1. We investigated a list of key issues that are of concern in the data mining and machine learning fields. The specific goals of the research were set out in section 1.5, following from the motivation and research problem identified in section 1.3 to answer

the research questions in section 1.4. The objectives are revisited in this section to summarise how they have been achieved and to discuss the research findings.

Objective 1: To study existing missing value imputation techniques.

In Chapter 3 we investigated six imputation algorithms to impute missing values in a heart failure dataset. The chapter discussed the missing value problem in the research dataset and how imputation methods can manipulate these problem data. Then, these methods were evaluated using several classification techniques such as random forest, REPTree, and J48. The imputation methods used were K-nearest Neighbour Imputation (KNNI), Expectation Maximisation Imputation (EMI), K-mean imputation, Most Common Imputation (MCI), Concept Most Common Imputation (CMC), and Support Vector Machine (SVM). Concept Most Common (CMC) has been suggested as the most appropriate imputation algorithm to impute the missing values in Hull LifeLab data, and random forest as an algorithm to classify data. The best outcome was obtained by using CMC and SVM because the technique of these methods calculates the missing values in consideration of the class label. Moreover, analysing the computational complexity shows that the incomplete data has high learning complexity compared to the imputed dataset.

Objective 2: To analyse eight feature selection methods to find the best methods that can be used to select features from clinical data.

In Chapter 4, we investigated the relevant features that can predict the target class. A unique contribution of our work is that we compare the underlying attribute rankings of each technique, as opposed to building classifiers using the selected features and comparing performance metrics, such as overall accuracy, of those classifiers. By measuring the rank correlation between the attribute rankings, it is easier to discern which techniques produce similar results, irrespective of the ultimate use of the data.

The selected subset improved the prediction performance of the research dataset. Thus, the selected subset can support clinicians to analyse and interpret specific features. The methods used to analyse the features were chi-square, ReliefF, information gain, correlation attributes, Cfs, Consistency, the embedded method, and wrapper. The results show that wrapper and chi-square are the two best methods that can be used to select features from a clinical dataset; however, wrapper has high computational complexity.

Objective 2.a: To find the significant features that improve predictive modelling and enhance the classification performance for the Hull *LifeLab* dataset.

In Chapter 4, we investigated the relevant features that can predict the target class. The selected subset improved the prediction and performance of the research dataset. Thus, the selected subset can support clinicians to analyse and interpret specific features. The similarity of features selected from all feature selection methods can identify the basic features that improve the prediction.

Objective 2.b: To identify the number of features that appropriate for each method.

We compared the attribute ranking of each technique, as opposed to building classifiers using the selected features and comparing performance metrics. Therefore, we found for each method how many features is the best choice. The methods used to select features entailed different techniques, so that some methods improve performance with small subset sizes, whereas others improved performance for large subsets. Chi-square, wrapper, ReliefF, embedded, consistency, and Cfs improve performance with a small number of features, whereas information gain and correlation need a large number of features to improve performance.

Objective 2.c: To investigate class imbalance and the effect on the feature selection subsets, as well as apply feature selection methods with the balanced classes.

In Chapter 5, we analysed class imbalance on the results obtained from the feature selection method. It was seen that some methods gave positive effects, whereas others gave negative effects. We found that only information gain improved the results with 11 and 23 features, whereas other methods yielded poor results for the balanced classes compared to the original dataset. The comparison was made between selected variables from the unbalanced classes and the variables gained from balanced data after resampling. The subsets selected by several feature selection methods were then resampled. On measuring the selected subsets after resampling, we found a significant improvement in terms of the performance matrix, especially specificity.

Objective 3: To investigate the autoencoder technique that can reduce the data mining methodology steps, by extracting features and classifying data.

In Chapter 6, an empirical study was made to examine an autoencoder neural network technique to classify data by extracting features. The findings show that the model obtained good results in terms of classification. Most of the machine learning methods used to classify data have the same parameters, which leads to the same results. In the autoencoder model, the parameters can be changed to find a suitable model for a particular dataset. In each experiment, we can change parameters such as learning rates, hidden layers, and regularisation. The best choice for the chronic heart failure dataset is to use two hidden layers, containing 30 and 40 neurons.

7.3 Scope for future research

Although this thesis has made some significant contributions to the area of mining medical data, there remain some limitations that open new avenues for further research. As we use of a chronic heart failure dataset, it would be better to have a different clinical dataset to be implemented through this framework, since another example of clinical data can give more information about the algorithms and tools used.

The wrapper method is a computationally complex method, although it shows the best results for the selected features. Thus, it would be interesting to combine another filtering method with wrapper to get a hybrid approach that will reduce the complexity of wrapper. This can be done by starting with a filter technique such as Cfs or information gain, to select a set of features. Then wrapper would be applied to the selected features instead of the whole feature set.

In the thesis, we demonstrate and implement most of the feature selection techniques, but did not include the feature extraction methods. Dimensionality reduction using feature extraction methods such as ICA and PCA should be considered in future research, particularly in the heart failure dataset. PCA can create new features using the existing features that may have a better association with the class. Accordingly, principal component analysis can be employed in different ways. We have block PCA with non-greedy l_1 −norm and l_2 −norm, as well as, greedy l_1 −norm and l_2 −norm. As a result, recommendation for future research are as follows:

1) Investigate feature extraction methods with a view to developing decision support systems.
    a. Employ feature extraction methods such as PCA and ICA.
    b. Apply different techniques of PCA such l_1-norm and l_2-norm.
2) Investigate the outcomes of feature extraction and its relationship with feature selection.

BIBLIOGRAPHY

Abraham, R., Simha, J. B. & Iyengar, S. S. (2007) Medical data mining with a new algorithm for feature selection and naive Bayesian classifier, *10th International Conference on Information Technology (ICIT 2007)*. Orissa, India, 17-20 Dec. 2007.

Acuna, E. & Rodriguez, C. (2004) The treatment of missing values and its effect on classifier accuracy, *Classification, Clustering, and Data Mining Applications*Springer, 639-647.

Aggarwal, C. C. & Zhai, C. (2012) *Mining text data*. London: Springer Science & Business Media.

Ahmad, T., Jameel, A. & Ahmad, B. (2011) Pattern recognition using statistical and neural techniques, *Computer Networks and Information Technology (ICCNIT), 2011 International Conference on*. IEEE.

Al-Shahib, A., Breitling, R. & Gilbert, D. (2005) Feature selection and the class imbalance problem in predicting protein function from sequence. *Applied Bioinformatics*, 4(3), 195-203.

Alkhasawneh, R. & Hargraves, R. H. (2014) Developing a hybrid model to predict student first year retention in STEM disciplines using machine learning techniques. *Journal of STEM Education: Innovations and Research*, 15(3), pp. 35-42.

Allison, P. D. (2000) Multiple imputation for missing data: A cautionary tale. *Sociological Methods & Research*, 28(3), 301-309.

Alpaydin, E. (2014) *Introduction to machine learning*. Cambridge, Massachusetts: The MIT press.

Andreou, P. C., Charalambous, C. & Martzoukos, S. H. (2002) Critical assessment of option pricing methods using artificial neural networks, *International Conference on Artificial Neural Networks*. Madrid, Spain: Springer.

Asheibi, A., Stirling, D. & Sutanto, D. (2009) Analyzing harmonic monitoring data using supervised and unsupervised learning. *IEEE Transactions on Power Delivery*, 24(1), 293-301.

Assawamakin, A., Prueksaaroon, S., Kulawonganunchai, S., Shaw, P. J., Varavithya, V., Ruangrajitpakorn, T. & Tongsima, S. (2013) Biomarker selection and classification of "-omics" data using a two-step bayes classification framework. *BioMed Research International*, 2013, 1-9.

Auria, L. & Moro, R. A. (2008) Support vector machines (SVM) as a technique for solvency analysis.

Baker, Y. S. (2014) *Applying machine learning techniques in diagnosing bacterial vaginosis*. Doctoral Dessirtation, . North Carolina Agricultural and Technical State University.

Balakrishnan, S., Narayanaswamy, R., Savarimuthu, N. & Samikannu, R. (2008) SVM ranking with backward search for feature selection in type II diabetes databases, *IEEE International Conference onSystems, Man and Cybernetics, 2008. SMC 2008* Singapore, Singapore: IEEE.

Barzi, F. & Woodward, M. (2004) Imputations of missing values in practice: results from imputations of serum cholesterol in 28 cohort studies. *American Journal of Epidemiology*, 160(1), 34-45.

Bashiri, M. & Geranmayeh, A. F. (2011) Tuning the parameters of an artificial neural network using central composite design and genetic algorithm. *Scientia Iranica*, 18(6), 1600-1608.

Bataineh, M. H. (2012) *Artificial neural network for studying human performance*. PhD (Dessertation). University of Lowa.

Batista, G. E. & Monard, M. C. (2002) A study of K-nearest neighbour as an imputation method. *HIS*, 87(251-260), 48.

Batista, G. E. & Monard, M. C. (2003) An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6), 519-533.

Batra, S., Parashar, H. J., Sachdeva, S. & Mehndiratta, P. (2013) Applying data mining techniques to standardized electronic health records for decision support, *Sixth International Conference onContemporary Computing (IC3), 2013* Noida, India: IEEE.

Batuwita, R. & Palade, V. (2010) Efficient resampling methods for training support vector machines with imbalanced datasets, *The 2010 International Joint Conference onNeural Networks (IJCNN),* . Barcelona, Spain: IEEE.

Bellazzi, R. & Zupan, B. (2008) Predictive data mining in clinical medicine: Current issues and guidelines. *International Journal of Medical Informatics*, 77, 81-97.

Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. (2007) Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19, 153.

Billings, J., Mijanovich, T., Dixon, J., Curry, N., Wennberg, D., Darin, B. & Steinort, K. (2006) *Case finding algorithms for patients at risk of re-hospitalisation*.London.

Blake, R. & Mangiameli, P. (2011) The effects and interactions of data quality and problem complexity on classification. *Journal of Data and Information Quality (JDIQ)*, 2(2), 1-28.

Branco, P., Torgo, L. & Ribeiro, R. (2015) A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv:1505.01658*.

Caesarendra, W., T Putri, F., Ariyanto, M. & D Setiawan, J. (2015) Pattern recognition methods for multi stage classification of parkinson's disease utilizing voice features, *International Conference onAdvanced Intelligent Mechatronics (AIM)*. Busan, South Korea: IEEE.

Cao, P., Liu, X., Zhang, J., Zhao, D., Huang, M. & Zaiane, O. (2016) ℓ 2, 1 norm regularized multi-kernel based joint nonlinear feature selection and over-sampling for imbalanced data classification. *Neurocomputing*, 234(19 April 2017), 38-57.

Carmona, C. J., Luengo, J., Gonzalez, P. & del Jesus, M. J. (2012) A preliminary study on missing data imputation in evolutionary fuzzy systems of subgroup discovery, *International Conference onFuzzy Systems (FUZZ-IEEE)*. Brisbane, QLD, Australia, 10-15 June 2012. IEEE.

Catley, C., Smith, K., McGregor, C. & Tracy, M. (2009) Extending CRISP-DM to incorporate temporal data mining of multidimensional medical data streams: A neonatal intensive care unit case study, *Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE International Symposium on*. Albuquerque, NM, USA, 2-5 Aug. 2009. IEEE.

Cerrito, P. B. (2006) *Introduction to data mining using SAS Enterprise Miner*SAS Publishing.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. & Wirth., R. (2000) *CRISP-DM 1.0 Step-by-step data mining guide*, 2000. Available online: [Accessed.

Chauhan, H., Kumar, V., Pundir, S. & Pilli, E. S. (2013) A comparative study of classification techniques for intrusion detection, *Computational and Business Intelligence (ISCBI), 2013 International Symposium on*. IEEE.

Chellappa, R. (2016) The changing fortunes of pattern recognition and computer vision. *Image and Vision Computing*, 55, 3-5.

Chen, S.-H. (2007) Computationally intelligent agents in economics and finance. *Information Sciences*, 177(5), 1153-1168.

Cho, K. (2014) *Foundations and advances in deep learning*PhD. Aalto University.

Choi, Y., Ozawa, S. & Lee, M. (2014) Incremental two-dimensional kernel principal component analysis. *Neurocomputing*, 134, 280-288.

Cios, K. J. & Moore, G. W. (2002) Uniqueness of medical data mining. *Artificial Intelligence in Medicine*, 26(1), 1-24.

Cleland, J. G., Zhang, J., Pellicori, P., Dicken, B., Dierckx, R., Shoaib, A., Wong, K., Rigby, A., Goode, K. & Clark, A. L. (2016) Prevalence and outcomes of anemia and hematinic deficiencies in patients with chronic heart failure. *JAMA Cardiology*, 1(5), 539-547.

Cohen, W. W. & Hirsh, H. (1994) The learnability of description logics with equality constraints. *Machine Learning*, 17(2), 169-199.

Dash, M. & Liu, H. (1997) Feature selection for classification. *Intelligent data analysis*, 1(1-4), 131-156.

Dash, M. & Liu, H. (2003) Consistency-based search in feature selection. *Artificial Intelligence*, 151(1), 155-176.

Daskalaki, S., Kopanas, I. & Avouris, N. (2006) Evaluation of classifiers for an uneven class distribution problem. *Applied Artificial Intelligence*, 20(5), 381-417.

Demšar, J. (2010) Algorithms for subsetting attribute values with Relief. *Machine Learning*, 78(3), 421-428.

Deng, J., Zhang, Z., Marchi, E. & Schuller, B. (2013) Sparse autoencoder-based feature transfer learning for speech emotion recognition, *Humaine Association Conference onAffective Computing and Intelligent Interaction (ACII)*. Geneva, Switzerland: IEEE.

Deng, L., Seltzer, M. L., Yu, D., Acero, A., Mohamed, A.-r. & Hinton, G. E. (2010) Binary coding of speech spectrograms using a deep auto-encoder, *Interspeech*. Makuhari, Chiba, Japan: Citeseer.

Devasena, C. (2015) Proficiency comparison of LADTree and REPTree classifiers for credit risk forecast. *International Journal on Computational Sciences & Applications (IJCSA)*, 5(1), 39-50.

Dodge, Y. & Zoppe, A. (2004) Adjusting the EM algorithm for design of experiments with missing data, *International Conference onInformation Technology Interfaces*. Cavtat, Croatia, 7-10 June 2004. IEEE.

Domingos, P. & Pazzani, M. (1997) On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3), 103-130.

Durgabai, R. (2014) Feature selection using ReliefF Algorithm. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(10), 8215-8218.

Endo, A., Shibata, T. & Tanaka, H. (2008) Comparison of seven algorithms to predict breast cancer survival(<Special Issue>Contribution to 21 Century Intelligent Technologies and Bioinformatics). *International Journal of Biomedical Soft Computing and Human Sciences*, 13(2), 11-16.

Engels, J. M. & Diehr, P. (2003) Imputation of missing longitudinal data: a comparison of methods. *Journal of Clinical Epidemiology*, 56(10), 968-976.

Esfandiari, N., Babavalian, M. R., Moghadam, A.-M. E. & Tabar, V. K. (2014) Knowledge discovery in medicine: Current issue and future trend. *Expert Systems with Applications*, 41(9), 4434-4463.

Europe, S. V. (2017) *CRISP-DM* 2017. Available online: http://www.sv-europe.com/crisp-dm-methodology/ [Accessed.

Farhangfar, A., Kurgan, L. & Dy, J. (2008) Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41(12), 3692-3705.

Farhangfar, A., Kurgan, L. & Pedrycz, W. (2007) A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 37(5), 692-709.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R. (1996) Advances in knowledge discovery and data mining.

Fewzee, P. & Karray, F. (2012) Dimensionality reduction for emotional speech recognition and 2012 International Confernece on Social Computing (SocialCom), *International Conference on Privacy, Security, Risk and Trust (PASSAT)* Amsterdam, Netherlands: IEEE.

Friedlin, J., Mahoui, M., Jones, J. & Jamieson, P. (2011) Knowledge discovery and data mining of free text radiology reports, *First IEEE International Conference on Healthcare Informatics, Imaging and Systems Biology (HISB)*. San Jose, CA, USA: IEEE.

Friedman, J., Hastie, T. & Tibshirani, R. (2001) *The elements of statistical learning*, *1*Berlin:Springer series in statistics, .

Fusco, G., Colombaroni, C., Comelli, L. & Isaenko, N. (2015) Short-term traffic predictions on large urban traffic networks: applications of network-based machine learning models and dynamic traffic assignment models, *International Conference onModels and Technologies for Intelligent Transportation Systems (MT-ITS)*. Budapest, Hungary: IEEE.

Ghoneim, V. F., Solouma, N. H. & Kadah, Y. M. (2011) The impact of missing values imputation methods in cDNA microarrays on downstream data analysis, *28th National Radio Science Conference (NRSC)*, . Cairo, Egypt, 26-28 April 2011. IEEE.

Gonen, M. (2013) Bayesian supervised dimensionality reduction. *Cybernetics, IEEE Transactions on*, 43(6), 2179-2189.

González, R. L. (2009) *Neural networks for variational problems in engineering*. Phd Universitat Politècnica de Catalunya.

Grzymala-Busse, J. W., Goodwin, L. K., Grzymala-Busse, W. J. & Zheng, X. (2005) Handling missing attribute values in preterm birth data sets, *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing.* Berlin, Heidelberg: Springer, 342-351.

Guo, X., Yin, Y., Dong, C., Yang, G. & Zhou, G. (2008) On the class imbalance problem, *Fourth International Conference on Natural Computation*. Jinan, China: IEEE.

Gupta, D., Malviya, A. & Singh, S. (2012) Performance analysis of classification tree learning algorithms. *IJCA) International Journal of Computer Applications*, 55(6).

Gürbüz, E. & Kılıç, E. (2011) Diagnosis of diabetes by using Adaptive SVM and feature selection, *IEEE 19th Conference on Signal Processing and Communications Applications (SIU)*. Antalya, Turkey: IEEE.

Guyon, I. & Elisseeff, A. (2003) An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar), 1157-1182.

Hall, M. A. & Holmes, G. (2003) Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, 15(6), 1437-1447.

Ham, J., Chen, Y., Crawford, M. M. & Ghosh, J. (2005) Investigation of the random forest framework for classification of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3), 492-501.

Hamed, T., Dara, R. & Kremer, S. C. (2014) An accurate, fast embedded feature selection for SVMs, *13th International Conference on Machine Learning and Applications (ICMLA)*. Detroit, MI, USA: IEEE.

Han, J. & Kamber, M. (2011) *Data mining: concepts and techniques*, Third edition. Waltham, MA, USA: Elsevier.

Han, Y., Park, K. & Lee, Y.-K. (2011) Confident wrapper-type semi-supervised feature selection using an ensemble classifier, *2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*. Dengleng, China: IEEE.

Hand, D. J., Mannila, H. & Smyth, P. (2001) *Principles of data mining*MIT press.

He, H. & Garcia, E. A. (2009) Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9), 1263-1284.

Hinton, G. E. (2002) Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8), 1771-1800.

Hinton, G. E., Dayan, P. & Revow, M. (1997) Modeling the manifolds of images of handwritten digits. *Neural Networks, IEEE Transactions on*, 8(1), 65-74.

Hinton, G. E., Osindero, S. & Teh, Y.-W. (2006) A fast learning algorithm for deep belief nets. *Neural computation*, 18(7), 1527-1554.

Hinton, G. E. & Salakhutdinov, R. R. (2006) Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504-507.

Hoyer, P. O. & Hyvärinen, A. (2000) Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, 11(3), 191-210.

Huang, J., Sun, H., Li, Y.-F. & Xie, M. (2015) An empirical study of dynamic incomplete-case nearest neighbor imputation in software quality data, *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*. IEEE.

Huang, S. H. (2015) Supervised feature selection: A tutorial. *Artificial Intelligence Research*, 4(2), 22-37.

IBM (2012) *Data mining techniques*, 2012. Available online: https://www.ibm.com/developerworks/library/ba-data-mining-techniques/ [Accessed.

Inc, T. M. (2016) *MATLAB and Statistics Toolbox Release 2016b*. Natick, Massachusetts, United States.

Jia, J., Yang, N., Zhang, C., Yue, A., Yang, J. & Zhu, D. (2013) Object-oriented feature selection of high spatial resolution images using an improved Relief algorithm. *Mathematical and Computer Modelling*, 58(3), 619-626.

Jilani, M. Z. M. B., Tucker, A. & Swift, S. (2016) Simultaneous modelling and clustering of visual field data, *29th International Symposium on Computer-Based Medical Systems*. IEEE.

Jing, L. (2012) *Missing data imputation*. PhD, University of California, Los Angeles.

Jonsson, P. & Wohlin, C. (2004) An evaluation of k-nearest neighbour imputation using likert data, *10th International Symposium on Software Metrics, 2004. Proceedings.* . Chicago, Illinois, USA, USA: IEEE.

Joshi, S. K. & Machchhar, S. (2014) An evolution and evaluation of dimensionality reduction techniques—A comparative study, *IEEE International Conference on Computational Intelligence and Computing Research* Coimbatore, India: IEEE.

Kaastra, I. & Boyd, M. (1996) Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10(3), 215-236.

Kaiser, J. (2014) Dealing with missing values in data. *Journal of Systems Integration*, 5(1), 42-51.

Kalmegh, S. (2015) Analysis of WEKA data mining algorithm REPTree, Simple CART and RandomTree for classification of Indian news. *International Journal of Innovative Science, Engineering and Technology*, 2(2), 438-46.

Karamizadeh, S., Abdullah, S. M. & Halimi, M. (2014) Advantage and drawback of support vector machine functionality, *International Conference on Computer, Communication, and Control Technology (I4CT 2014).* Langkawi, Kedah, Malaysia: IEEE.

Karczmarek, P., Kiersztyn, A., Pedrycz, W. & Dolecki, M. (2017) An application of chain code-based local descriptor and its extension to face recognition. *Pattern Recognition*, 65, 26-34.

Karmaker, A. & Kwek, S. (2005) Incorporating an EM-approach for handling missing attribute-values in decision tree induction, *Fifth International Conference on Hybrid Intelligent Systems, 2005. HIS '05.* . Rio de Janeiro, Brazil, Brazil, 6-9 Nov. 2005.

Kaur, G. & Chhabra, A. (2014) Improved J48 classification algorithm for the prediction of diabetes. *International Journal of Computer Applications*, 98(22), 13-22.

Kausar, N., Palaniappan, S., Samir, B. B., Abdullah, A. & Dey, N. (2016) Systematic analysis of applied data mining based optimization algorithms in clinical attribute extraction and classification for diagnosis of cardiac patients, *Applications of intelligent optimization in biology and medicine*Springer, 217-231.

Khodaskar, A. & Ladhake, S. (2014) Pattern recognition: Advanced development, techniques and application for image retrieval, *International Conference on Communication and Network Technologies (ICCNT)*. Sivakasi, India: IEEE.

Kile, H. & Uhlen, K. (2012) Supervised and unsupervised learning in composite reliability evaluation, *2012 IEEE Power and Energy Society General Meeting*. IEEE.

Kirshners, A., Parshutin, S. & Gorskis, H. (2017) Entropy-based classifier enhancement to handle imbalanced class problem. *Procedia Computer Science*, 104, 586-591.

Kohavi, R. & John, G. H. (1997) Wrappers for feature subset selection. *Artificial Intelligence*, 97(1), 273-324.

Kohlberg, G. & Hammer, M. (2014) *GetTheDiagnosis.org - A Database of Diagnostic Accuracy*, 2014. Available online: http://getthediagnosis.org [Accessed.

Kononenko, I. & Kukar, M. (2007) *Machine learning and data mining: introduction to principles and algorithms*. West Sussex, UK: Horwood Publishing.

Kumar, V. & Minz, S. (2014) Feature selection. *SmartCR*, 4(3), 211-229.

Kumdee, O., Ritthipravat, P., Bhongmakapat, T. & Cheewaruangroj, W. (2008) Dealing with missing values for effective prediction of NPC recurrence, *SICE Annual Conference*. Tokyo, Japan, 20-22 Aug. 2008.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J. & Bengio, Y. (2007) An empirical evaluation of deep architectures on problems with many factors of variation, *Proceedings of the 24th international conference on Machine learning*. Corvalis, Oregon, USA: ACM.

Lavrač, N. (1999) Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16(1), 3-23.

Lawrence, S., Giles, C. L. & Tsoi, A. C. (1997) Lessons in neural network training: Overfitting may be harder than expected, *National Conference on Artificial Intelligence*. Providence, Rhode Island.

Ledolter, J. (2013) *Data mining and business analysis with R*. New Jersy: John Wiley & Sons.

Levy, W. C., Mozaffarian, D., Linker, D. T., Sutradhar, S. C., Anker, S. D., Cropp, A. B., Anand, I., Maggioni, A., Burton, P. & Sullivan, M. D. (2006) The Seattle heart failure model prediction of survival in heart failure. *Circulation*, 113(11), 1424-1433.

Li, D., Deogun, J., Spaulding, W. & Shuart, B. (2004) Towards missing data imputation: A study of fuzzy k-means clustering method, *International Conference on Rough Sets and Current Trends in Computing*. Uppsala, Sweden: Springer-Verlag, Berlin, 573-579.

Li, H., Jiang, T. & Zhang, K. (2006) Efficient and robust feature extraction by maximum margin criterion. *IEEE Transactions on Neural Networks*, 17(1), 157-165.

Li, J., Fu, A. W.-c., He, H., Chen, J., Jin, H., McAullay, D., Williams, G., Sparks, R. & Kelman, C. (2005) Mining risk patterns in medical data, *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. Chicago, Illinois, USA ACM.

Li, X., Yu, J., Jia, Z. & Song, J. (2014) Harmful algal blooms prediction with machine learning models in Tolo Harbour, *International Conference on Smart Computing (SMARTCOMP)*. Hong Kong, China: IEEE.

Li, Y., Neilson, M. P., Whellan, D. J., Schulman, K. A., Levy, W. C. & Reed, S. D. (2013) Associations between seattle heart failure model scores and health utilities: findings from HF-ACTION. *Journal of Cardiac Failure*, 19(5), 311-316.

Lin, J.-H. & Haug, P. J. (2008) Exploiting missing clinical data in Bayesian network modeling for predicting medical problems. *Journal of Biomedical informatics*, 41(1), 1-14.

Lin, T. H. (2010) A comparison of multiple imputation with EM algorithm and MCMC method for quality of life missing data. *Quality & Quantity*, 44(2), 277-287.

Little, R. J. & Rubin, D. B. (1989) The analysis of social science data with missing values. *Sociological Methods & Research*, 18(2-3), 292-326.

Liu, Y. & Liu, Y. (2010) Incremental learning method of least squares support vector machine, *International Conference on Intelligent Computation Technology and Automation (ICICTA)*. Changsha, China: IEEE.

Lobato, F., Sales, C., Araujo, I., Tadaiesky, V., Dias, L., Ramos, L. & Santana, A. (2015) Multi-objective genetic algorithm for missing data imputation. *Pattern Recognition Letters*, 68, 126-131.

Lobur, M., Stekh, Y. & Artsibasov, V. (2011) Challenges in knowledge discovery and data mining in datasets, *Perspective Technologies and Methods in MEMS Design.* Polyana, Ukraine: IEEE, 232-233.

Lobur, M., Stekh, Y., Kernytskyy, A. & Sardieh, F. M. (2008) Some trends in knowledge discovery and data mining, *International Conference on Perspective Technologies and Methods in MEMS Design*. Polyana, Ukraine: IEEE.

Lokeswari, Y. & Jacob, S. G. (2015) A cloud-based data mining framework for improved clinical diagnosis through parallel classification, *International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. Davangere, India: IEEE.

Loyola-González, O., Medina-Pérez, M. A., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., Monroy, R. & García-Borroto, M. (2017) PBC4cip: A new contrast pattern-based classifier for class imbalance problems. *Knowledge-Based Systems*, 115, 100-109.

Lu, J., Hales, A., Rew, D. & Keech, M. (2016) Timeline and episode-structured clinical data: Pre-processing for data mining and analytics, *32nd International Conference on Data Engineering Workshops (ICDEW)*. Helsinki, Finland: IEEE.

Lu, X., Tsao, Y., Matsuda, S. & Hori, C. (2013) Speech enhancement based on deep denoising autoencoder. *Interspeech*, 436-440.

Lu, Z. & Su, J. (2010) Clinical data management: Current status, challenges, and future directions from industry perspectives. *Open Access J Clin Trials*, 2, 93-105.

Lukoševičius, M. & Jaeger, H. (2009) Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3), 127-149.

Ma, Y., Tan, Y., Zhang, C. & Mao, Y. (2015) A data mining model of knowledge discovery based on the deep learning, *10th Conference on Industrial Electronics and Applications (ICIEA)*. Auckland, New Zealand: IEEE.

Mahdiyah, U., Irawan, M. I. & Imah, E. M. (2015) Integrating data selection and extreme learning machine for imbalanced data. *Procedia Computer Science*, 59, 221-229.

Mallinson, H. & Gammerman, A. (2003) Imputation using support vector machines. London: Department of Computer Science. Royal Holloway, University of London.

Margineantu, D. D. & Dietterich, T. G. (2000) *Bootstrap methods for the cost-sensitive evaluation of classifiers*.Corvallis.

Masci, J., Meier, U., Cireşan, D. & Schmidhuber, J. (2011) Stacked convolutional auto-encoders for hierarchical feature extraction, *International Conference on Artificial Neural Networks.* Espoo, Finland: Springer-Verlag 52–59.

Matignon, R. (2007) *Data mining using SAS enterprise miner*. New Jersy: John Wiley & Sons.

Mazurowski, M. A., Habas, P. A., Zurada, J. M., Lo, J. Y., Baker, J. A. & Tourassi, G. D. (2008) Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks*, 21(2), 427-436.

McGee, S. (2002) Simplifying likelihood ratios. *Journal of General Internal Medicine*, 17(8), 647-650.

Melgani, F. & Bruzzone, L. (2004) Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on geoscience and remote sensing*, 42(8), 1778-1790.

Menardi, G. & Torelli, N. (2014) Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery,* 28(1), 92-122.

Moore, L. (2015) *Data mining for heart failure: an investigation into the challenges in real life clinical datasets*. (PhD) University of Hull.

Mullins, I. M., Siadaty, M. S., Lyman, J., Scully, K., Garrett, C. T., Greg Miller, W., Muller, R., Robson, B., Apte, C., Weiss, S., Rigoutsos, I., Platt, D., Cohen, S. &

Knaus, W. A. (2006) Data mining and clinical data repositories: Insights from a 667,000 patient data set. *Computers in Biology and Medicine*, 36(12), 1351-1377.

Murty, M. N. & Devi, V. S. (2011) *Pattern recognition: an algorithmic approach*. London: Springer.

Namratha, M., Prajwala, T. & Malvika, M. (2013) Collative study of classifiers in pattern recognition. *International Journal of Computer Network and Security(IJCNS)*, 5(1), 12-15.

Ng, A., Ngiam, J., Yu Foo, C., Mai, Y., Suen , C., Coates, A., Maas, A., Hannun, A., Huval, B., Wang, T. & Tandon, S. (2013) *Deep learning tutorial!*, 2013. Available online: http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/ [Accessed.

Novakovic, J. (2009) Using information gain attribute evaluation to classify sonar targets, *17th Telecommunications forum TELFOR*. Serbia, Belgrade,.

Novaković, J., Štrbac, P. & Bulatović, D. (2011) Toward optimal feature selection using ranking methods and classification algorithms. *Yugoslav Journal of Operations Research*, 21(1), 119-135

Nuffieldtrust (2012) *Nuffield Trust works on PARR30*, 2012. Available online: [Accessed.

Ojala, D. P. (2012) *Computation of option prices with neural networks and option price lags*. (PhD) State University of New York at Binghamton.

Olshausen, B. A. & Field, D. J. (1997) Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23), 3311-3325.

Olson, D. L. & Delen, D. (2008) *Advanced data mining techniques*. Verlag Berlin Heidelberg: Springer Science & Business Media.

Ozcift, A. (2012) SVM feature selection based rotation forest ensemble classifiers to improve computer-aided diagnosis of Parkinson disease. *Journal of Medical Systems*, 36(4), 2141-2147.

Pal, M. & Mather, P. M. (2003) An assessment of the effectiveness of decision tree methods for land cover classification. *Remote Sensing of Environment*, 86(4), 554-565.

Patel, K. A. & Prajapati, R. C. (2016) A survey-vulnerability classification of bug reports using multiple machine learning approach. *Compusoft*, 5(3), 2071 - 2073.

Pencina, M. J., D'Agostino, R. B. & Vasan, R. S. (2008) Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in Medicine*, 27(2), 157-172.

Pfeffer, M. A. & Skali, H. (2013) PRAISE (prospective randomized amlodipine survival evaluation) and criticism. *The Journal of the American College of Cardiology (JACC)*, 1(4), 315-317.

Pigott, T. D. (2001) A review of methods for missing data. *Educational Research and Evaluation*, 7(4), 353-383.

Pindoriya, N. M., Jirutitijaroen, P., Srinivasan, D. & Singh, C. (2011) Composite reliability evaluation using Monte Carlo simulation and least squares support vector classifier. *IEEE Transactions on Power Systems*, 26(4), 2483-2490.

Poolsawad, N., Kambhampati, C. & Cleland, J. (2014a) Balancing class for performance of classification with a clinical dataset. *In: Proceedings of the World Congress on Engineering*, 1.

Poolsawad, N., Moore, L., Kambhampati, C. & Cleland, J. G. (2014b) Issues in the mining of heart failure datasets. *International Journal of Automation and Computing*, 11(2), 162-179.

Potamias, G. & Moustakis, V. (2001) Knowledge discovery from distributed clinical data sources: the era for internet-based epidemiology, *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*. IEEE.

Poultney, C., Chopra, S. & Cun, Y. L. (2006) Efficient learning of sparse representations with an energy-based model, *Proceedings of the 19th International Conference on Neural Information Processing Systems.* Canada: Advances in neural information processing systems, 1137-1144.

Prather, J. C., Lobach, D. F., Goodwin, L. K., Hales, J. W., Hage, M. L. & Hammond, W. E. (1997) Medical data mining: knowledge discovery in a clinical data warehouse. *Proceedings of the AMIA Annual Fall Symposium*, 101-105.

Quinlan, J. R. (1996) Improved use of continuous attributes in C4. 5. *Journal of Artificial Intelligence Research*, 4, 77-90.

Rahman, M. M. & Davis, D. N. (2013) Machine learning-based missing value imputation method for clinical datasets, *IAENG Transactions on Engineering Technologies*Springer, 245-257.

Rahman, S. A., Yuxiao, H., Claassen, J. & Kleinberg, S. (2014) Imputation of missing values in time series with lagged correlations, *International Conference on Data Mining Workshop (ICDMW).* Shenzhen, China, 14-14 Dec. 2014. IEEE, 753-762.

Rajiv Wadhwa, M., Saul, M. I. & Penrod, L. E. (2008) Analysis of a failed clinical decision support system for management of congestive heart failure. *Proceedings AMIA 2008 Symposium* 773-777.

Ravichandran, S., Srinivasan, V. B. & Ramasamy, C. (2012) Comparative study on decision tree techniques for mobile call detail record. *Journal of Communication and Computer*, 9(12), 1331-1335.

Rifai, S., Vincent , P., Muller, X., Glorot, X. & Bengio, Y. (2011) Contractive auto-encoders: explicit invariance during feature extraction, *In Proceedings of the 28th international conference on machine learning.* Bellevue, WA, USA,: ICML-1, 833-840.

Rinaldi, F. (2009) *Mathematical programming methods for minimizing the zero-norm over polyhedral sets*. (PhD) University of Rome.

Rubin, D. B. (1987) *Multiple imputation for nonresponse in survey*. New Jersey: Wiley-Interscience.

Rutkowski, L., Jaworski, M., Pietruczuk, L. & Duda, P. (2014) Decision trees for mining data streams based on the gaussian approximation. *IEEE Transactions on Knowledge and Data Engineering*, 26(1), 108-119.

Salakhutdinov, R., Mnih, A. & Hinton, G. (2007) Restricted boltzmann machines for collaborative filtering, *24th International Conference on Machine Learning* Corvallis, OR, Canada: ACM.

Sang Jeen, H., May, G. S. & Dong-Cheol, P. (2003) Neural network modeling of reactive ion etching using optical emission spectroscopy data. *Semiconductor Manufacturing, IEEE Transactions on*, 16(4), 598-608.

Sapkal, S. D., Kakarwal, S. N. & Revankar, P. (2007) Analysis of classification by supervised and unsupervised learning, *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*.

Saqlain, M., Athar, A., Saqib, N. A. & Khan, M. A. (2016) Developing a classification model for an effective treatment of heart failure. *International Journal of Computer Science and Information Security*, 14(8), 413.

Sartipy, U., Goda, A., Yuzefpolskaya, M., Mancini, D. M. & Lund, L. H. (20014) Utility of the Seattle Heart Failure Model in patients with cardiac resynchronization therapy and implantable cardioverter defibrillator referred for heart transplantation. *American Heart Journal*, 168(3), 325-331.

SAS-Institute (2017) *Data Mining and SEMMA*, 2017. Available online: http://documentation.sas.com/?docsetId=emcs&docsetTarget=n0pejm83csbja4n1xueveo2uoujy.htm&docsetVersion=12.3&locale=en [Accessed.

Schmitt, P., Mandel, J. & Guedj, M. (2015) A Comparison of Six Methods for Missing Data Imputation. *Journal of Biometrics & Biostatistics*, 6(1), 1-6.

SCI2S (2007) *Classification with imbalanced datasets*, 2007. Available online: http://sci2s.ugr.es/imbalanced [Accessed.

Selvakuberan, K., Kayathiri, D., Harini, B. & Devi, M. I. (2011) An efficient feature selection method for classification in health care systems using machine learning techniques, *3rd International Conference on Electronics Computer Technology* Kanyakumari, India: IEEE.

Shankar, S., Sarkar, B. D., Sabitha, S. & Mehrotra, D. (2016) Performance analysis of student learning metric using K-mean clustering approach K-mean cluster, *6th International Conference Cloud System and Big Data Engineering (Confluence)*. Noida, India: IEEE.

Sharma, R., Singh, S. N. & Khatri, S. (2016) Medical data mining using different classification and clustering techniques: a critical survey, *Computational Intelligence & Communication Technology (CICT), Second International Conference on.* 12-13 Feb. 2016. IEEE, 687-691.

Shelton, R. J., Clark, A. L., Kaye, G. C. & Cleland, J. G. (2010) The atrial fibrillation paradox of heart failure. *Congestive Heart Failure*, 16(1), 3-9.

Shin, K., Fernandes, D. & Miyazaki, S. (2011) Consistency measures for feature selection: a formal definition, relative sensitivity comparison, and a fast algorithm, *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Barcelona, Spain: Citeseer.

Shouman, M., Turner, T. & Stocker, R. (2012) Using data mining techniques in heart disease diagnosis and treatment, *Japan-Egypt Conference on Electronics, Communications and Computers (JEC-ECC).* Alexandria, Egypt: IEEE.

Sivapriya, T., Kamal, A. N. B. & Thavavel, V. (2012) Imputation and classification of missing data using least square support vector machines–a new approach in dementia diagnosis. *International Journal of Advanced Research in Artificial Intelligence*, 1(4), 29-33.

Sokolova, M. & Lapalme, G. (2009) A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427-437.

Song, Q., Shepperd, M., Chen, X. & Liu, J. (2008) Can k-NN imputation improve the performance of C4. 5 with small software project data sets? A comparative evaluation. *Journal of Systems and Software*, 81(12), 2361-2370.

Soni, J., Ansari, U., Sharma, D. & Soni, S. (2011) Predictive data mining for medical diagnosis: An overview of heart disease prediction. *International Journal of Computer Applications*, 17(8), 43-48.

Sow, D., Turaga, D. S. & Schmidt, M. (2013) Mining of sensor data in healthcare: a survey, *Managing and Mining Sensor Data*. Boca Raton, FL, USA: Springer, 459-504.

Su, X., Greiner, R., Khoshgoftaar, T. M. & Napolitano, A. (2011a) Using classifier-based nominal imputation to improve machine learning, *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Berlin, Heidelberg: Springer, 124-135.

Su, Y.-S., Gelman, A., Hill, J. & Yajima, M. (2011b) Multiple imputation with diagnostics (mi) in R: Opening windows into the black box. *Journal of Statistical Software*, 45(2), 1-31.

Sun, Y. & Du, M. (2006) DT-CWT feature based classification using orthogonal neighborhood preserving projections for face recognition, *International Conference on Computational Intelligence and Security,*. Guangzhou, China: IEEE.

Suthaharan, S. (2016) *Machine learning models and algorithms for big data classification*. Greensboro, NC, USA: : Springer.

Tan, C. C. & Eswaran, C. (2008) Performance comparison of three types of autoencoder neural networks, *Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference on*. Kuala Lumpur, Malaysia, 13-15 May 2008. IEEE.

Tan, C. C. & Eswaran, C. (2009) *Autoencoder neural networks: a performance study based on image reconstruction, recognition and compression* Koln, Germany: Lambert Academic Publishing.

Teoh, E. J., Tan, K. C. & Xiang, C. (2006) Estimating the number of hidden neurons in a feedforward network using the singular value decomposition. *IEEE Transactions on Neural Networks*, 17(6), 1623-1629.

Thompson, B. B., Marks, R. J., Choi, J. J., El-Sharkawi, M. A., Ming-Yuh, H. & Bunje, C. (2002) Implicit learning in autoencoder novelty assessment, *International Joint Conference on Neural Networks*. Honolulu, HI, USA, USA, 2002.

Thrun, S., Martin, C., Liu, Y., Hähnel, D., Emery-Montemerlo, R., Chakrabarti, D. & Burgard, W. (2004) A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics and Automation*, 20(3), 433 - 442.

Tong, S. & Koller, D. (2001) Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(Nov), 45-66.

Tripoliti, E. E., Papadopoulos, T. G., Karanasiou, G. S., Naka, K. K. & Fotiadis, D. I. (2017) Heart failure: diagnosis, severity estimation and prediction of adverse events through machine learning techniques. *Computational and Structural Biotechnology Journal*, 15, 26-47.

Tsumoto, S. (2000) Problems with mining medical data, *The 24th Annual International Computer Software and Applications Conference*. Taipei, Taiwan: IEEE.

Uzer, M. S., Inan, O. & Yılmaz, N. (2013) A hybrid breast cancer detection system via neural network and feature selection based on SBS, SFS and PCA. *Neural Computing and Applications*, 23(3), 719-728.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. (2010) Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec), 3371-3408.

Wahed, M. A. & Wahba, K. (2003) Data mining based-assistant tools for physicians to diagnose diseases, *IEEE 46th Midwest Symposium on Circuits and Systems*. Cairo, Egypt: IEEE.

Wang, L., Zhang, Y. & Feng, J. (2005) On the Euclidean distance of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8), 1334-1339.

Washington, U. o. (2012) *Seatle Heart Failure Model*, 2012. Available online: https://depts.washington.edu/shfm [Accessed.

Wasikowski, M. & Chen, X.-w. (2010) Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on knowledge and data engineering*, 22(10), 1388-1400.

Wei, H. (2005) Theory and methods for neural networks architecture design. *National defence industry press, Beijing*.

Weitschek, E., Felici, G. & Bertolazzi, P. (2013) Clinical Data Mining: Problems, Pitfalls and Solutions, *Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on*. 26-30 Aug. 2013.

Wirth, R. & Hipp, J. (2000) CRISP-DM: Towards a standard process model for data mining, *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*. New York, NY: Citeseer.

Witten, I. H. & Frank, E. (2011) *Data Mining: Practical machine learning tools and techniques*. Cambridge, MA, United States: Morgan Kaufmann.

Yang, X.-S., Chien, S. F. & Ting, T. O. (2015) *Bio-inspired computation in telecommunications*. Waltham, USA: Elsevier.

Yu, L. & Liu, H. (2004) Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5(Oct), 1205-1224.

Yuan, Y. C. (2010) *Multiple imputation for missing data: Concepts and new development (Version 9.0). Rockville, MD, SAS Institute Inc*, 49(1), 1-11.

Zhang, H., Berg, A. C., Maire, M. & Mali, J. (2006) SVM-KNN: Discriminative nearest neighbor classification for visual category recognition, *CVPR'06*. New York: IEEE.

Zhang, X., Wu, G., Dong, Z. & Crawford, C. (2015) Embedded feature-selection support vector machine for driving pattern recognition. *Journal of the Franklin Institute*, 352(2), 669-685.

Zhang, Y., Kambhampati, C., Davis, D. N., Goode, K. & Cleland, J. G. (2012) A comparative study of missing value imputation with multiclass classification for clinical heart failure data, *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. Sichuan, China: IEEE.

Zheng, Z., Wu, X. & Srihari, R. (2004) Feature selection for text categorization on imbalanced data. *ACM Sigkdd Explorations Newsletter*, 6(1), 80-89.

Zhu, L., Han, B., Li, L., Xu, S., Mou, H. & Zheng, Z. (2009) Null Space LDA based feature extraction of mass spectrometry data for cancer classification, *Biomedical Engineering and Informatics, 2009. BMEI'09. 2nd International Conference on*. IEEE.

Appendix I: Variables of the Hull LifeLab Dataset.

	Variable	Min	Max	Mean (μ)	STD (σ)	Missing
1	Age (years)	27	96	70.578	10.568	0
2	Sodium (mmol/L)	123	148	138.808	3.156	68 (3%)
3	Potassium (mmol/L)	2.2	6.3	4.342	0.482	79 (4%)
4	Chloride (mmol/L)	78	112	101.522	4.153	67 (3%)
5	Bicarbonate (mmol/L)	19	45	28.659	2.979	67 (3%)
6	Urea (mmol/L)	1.2	38.5	7.392	4.198	67 (3%)
7	Creatinine (umol/L)	37	561	108.467	46.039	67 (3%)
8	Calcium (mmol/L)	1.32	2.78	2.344	0.104	97 (5%)
9	Adj Calcium (mmol/L)	1.04	2.85	2.431	0.096	98 (5%)
10	Phosphate (mmol/L)	0.52	10.04	1.128	0.291	112 (6%)
11	Bilirubin (umol/L)	1.1	69	15.563	6.553	84 (4%)
12	Alkaline Phosphatase (iu/L)	0.55	344	78.463	32.487	81 (4%)
13	ALT (iu/L)	4	283	24.967	16.994	81 (4%)
14	Total protein (g/L)	36	97	67.043	5.085	77 (4%)
15	Albumin (g/L)	20	47	37.592	3.619	82 (4%)
16	Uric acid (mmol/L)	0.13	12.3	0.425	0.387	342 (18%)
17	Glucose (mmol/L)	1.9	25.1	6.697	2.815	241 (12%)
18	Cholesterol (mmol/L)	1.7	13.8	4.774	1.256	200 (10%)
19	Triglycerides (mmol/L)	0.3	9.5	1.689	1.066	206 (11%)
20	Haemoglobin (g/dL)	0.22	18.7	13.395	1.705	70 (4%)
21	White Cell Count (10^9/L)	2.3	27.4	7.258	2.136	73 (4%)
22	Platelets (10^9/L)	13.2	763	237.219	72.444	74 (4%)
23	MCV (fL)	7.5	115.4	90.516	6.322	306 (16 %)
24	Hct (fraction)	0.232	0.95	0.413	0.046	297 (15%)
25	Iron (umol/L)	1	52	15.068	6.071	284 (15%)
26	Vitamin B12 (ng/L)	1.28	1667	362.082	190.863	182 (9%)
27	Ferritin (ug/L)	0.34	929	115.311	112.357	393 (20%)
28	CRP (mg/L)	0.2	184	8.985	16.357	96 (5%)
29	TSH (mU/L)	0.04	50	2.075	2.393	163 (8%)
30	MR-proANP	4	1602	225.844	193.422	0
31	MR-proADM	0.2	4.46	0.835	0.487	319 (16%)
32	CT-proET1	20	396	84.529	46.053	293 (15%)

33	CT-proAVP	0.6	199	12.22	15.221	4 (0%)
34	PCT	0.007	0.537	0.029	0.03	17 (1%)
35	ECG (bpm)	36	160	73.444	17.671	20 (1%)
36	QRS width msec)	40	412	107.191	30.851	65 (3%)
37	QT (msec)	132	577	408.194	47.259	83 (4%)
38	LVEDD (cm)	2.18	9.8	5.568	1.073	380 (20%)
39	LVEDD (Hgt indexed)	1.35	5.57	3.316	0.604	396 (20%)
40	BSA (m^2)	1.063	3.164	1.957	0.268	38 (2%)
41	Aortic Root (cm)	0.61	5.4	3.25	0.492	247 (13%)
42	Left Atrium (cm)	1.24	8.1	4.169	0.8	220 (11%)
43	Left Atrium (BSA Indexed)	0.69	4.79	2.158	0.473	253 (13%)
44	Left Atrium (Hgt indexed)	0.82	4.79	2.487	0.473	235 (12%)
45	Aortic Velocity (m/s)	0.1	5.4	1.375	0.594	352 (18%)
46	E	0.22	3	0.813	0.315	354 (18%)
47	Height (m)	1.2	1.96	1.677	0.099	18 (1%)
48	Weight (kg)	30	193.8	81.267	18.834	39 (2%)
49	BMI (kg/m^2)	12.623	66.277	28.807	5.84	38 (2%)
50	Pulse (bpm)	38	150	74.057	16.371	18 (1%)
51	Systolic BP (mmHg)	75	233	139.972	25.346	13 (1%)
52	Diastolic BP (mmHg)	36	195	79.778	14.562	13(1%)
53	Pulse BP (mmHg)	9	148	60.194	20.742	13 (1%)
54	FEV1 (L)	0.19	4.72	1.726	0.791	70 (4%)
55	FEV1 Predicted (L)	0.645	4.59	2.4	0.63	68 (4%)
56	FEV1	8.653	175.004	71.486	24.631	72 (4%)
57	FVC (L)	0.19	6.31	2.569	0.97	68 (3%)
58	FVC Predicted (L)	1.225	6.002	3.517	0.864	68 (3%)
59	FVC	8.025	165.187	72.831	19.969	75 (4%)
60	PEFR (L)	0.46	834	217.157	134.624	133 (7%)
61	Mortality	Alive=1459		Dead=485		

Appendix II: Statistics on the Research dataset for heart failure from HYMS.

	Feature name	Data type	Missing values	Range		Unique values	Mean	Median
1	Age	integer	0	27	96	66	70	72
2	Sodium Mmol/L	integer	68	123	148	26	138	139
3	Potassum	Real	79	2.2	6.3	93	4.36	4.3
4	Chloride	integer	67	78	112	28	101	102
5	Bicarbonate	integer	67	19	45	24	28	29
6	Urea	real	67	1.2	38.5	257	7.45	6.3
7	Creatinine	integer	67	46	561	207	109	99
8	Calcium	Real	97	1.32	2.78	136	2.36	2.35
9	Adj Calcium	Real	98	1.4	2.85	147	2.44	2.44
10	Phosphate	Real	112	0.52	10.04	217	1.14	1.13
11	Bilirubin	integer	84	1.1	69	137	15.8	14
12	Alkaline	integer	81	0.55	344	255	80.01	73
13	ALT (iu/L)	integer	81	4	283	89	25	22
14	Total Protien g/L	Integer	77	36	97	43	67	67
15	Albumin g/L	Integer	82	20	47	27	37	38
16	Uric Acid mmol/L	Real	342	0.13	12.3	413	0.46	0.42
17	Glucose mmol/L	Real	241	2.6	25.1	417	8.2	6.4
18	Cholesterol mmol/L	Real	200	1.7	13.8	300	6.05	5.2
19	Triglycerides mmol/L	Real	206	0.3	9.5	289	2.16	1.8
20	Haemoglobin g/dL	Real	70	1	18.7	118	14.74	14.2
21	White cell count 10^9/L	Real	73	2.3	27.4	266	9.62	7.8
22	Platelets 10^9/L	Integer	74	13.2	763	553	313.14	256
23	MCV fL	Real	306	7.5	115.4	399	98.82	94.5
24	Hct (fraction)	Real	297	0.23	0.95	505	0.55	0.44
25	Iron (umol/L)	Integer	284	2	52	324	24.29	22
26	Vitamin B12 (ng/L)	Integer	182	4.5	1669	1013	637.86	635.18
27	Ferritin (ug/L)	Real	393	5	929	868	203.33	211.73
28	CRP (mg/L)	Real	96	0.2	184	384	18.89	19.87
29	TSH (mU/L)	Real	163	0.05	50	421	4.46	5.05
30	MR-proANP	Real	0	4	1602	918	505	575.42
31	MR-proADM	Real	319	0.2	4.46	751	1.95	2.31
32	CT-proET1	Real	293	20	396	700	204.12	236.51
33	CT-proAVP	Real	4	0.7	199	656	34.82	38.53
34	PCT	Real	17	0.01	0.54	188	0.08	0.1

35	Rate (ECG) (bpm)	Integer	20	36	160	84	129	160
36	QRS Width (msec)	Integer	65	66	412	110	296	387
37	QT	Integer	83	244	577	154	519	577
38	LVEDD (cm)	Real	380	2.85	9.81	224	8.59	9.8
39	LVEDD (Hgt indexed)	Real	396	1.64	5.57	209	4.92	5.57
40	BSA (m^2)	Real	38	1.19	3.16	497	2.82	3.16
41	Aortic Root (cm)	Real	247	0.61	5.4	140	4.82	5.4
42	Left Atrium (cm)	Real	220	1.5	9.09	179	7.74	9.09
43	Left Atrium (BSA indexed)	Real	253	0.76	5.7	187	4.72	5.7
44	Left Atrium (Hgt indexed)	Real	253	0.87	5.98	189	5.02	5.98
45	Aortic Velocity (m/s)	Real	352	0.1	5.4	207	4.38	5.4
46	E	Real	354	0.28	3	120	2.51	3
47	Height (Exam) (m)	Real	18	1.4	1.96	50	1.9	1.96
48	Weight (EXAM) (Kg)	Real	39	35.5	193.8	289	168.1	193.8
49	BMI	Real	38	12.62	66.28	395	57.67	66.28
50	Pulse (Exam) (bpm)	Integer	18	39	150	69	132	150
51	Systolic BP (mmHg)	Integer	13	75	237	108	214	237
52	Diastolic BP(mmHg)	Integer	13	39	195	66	169	195
53	Pulse BP (mmHg)	Integer	13	19	142	89	123	142
54	FEV1 (L)	Real	72	0.29	4.72	230	4.07	4.72
55	FEV1 Predicted (L)	Real	69	0.95	4.59	392	4.11	4.59
56	FEV1	Real	79	13.77	175	395	152.52	175
57	FVC (L)	Real	68	0.75	6.31	237	5.5	6.31
58	FVC Predicted (L)	Real	68	1.57	6	391	5.46	6
59	FVC	Real	75	27.2	165.19	394	145.09	165.19
60	PEFR (L)	Real	133	2.02	834	271	698	834
61	Martality	Nominal	0	0	1	2	0	1

Appendix III: Performance Results of Applying Several Hidden Layers for Autoencoder.

Hidden Layers	TP	FN	FP	TN	ACC	SEN	SPEC	PPV
10 10	1021	137	292	300	77.29%	88.17%	50.68%	77.76%
10 15	1070	163	243	274	79.61%	86.78%	53.00%	81.49%
10 20	1107	174	206	265	80.69%	86.42%	56.26%	84.31%
10 25	1060	163	253	274	79.46%	86.67%	51.99%	80.73%
10 30	1079	167	234	272	79.87%	86.60%	53.75%	82.18%
10 40	1056	158	257	279	79.10%	86.99%	52.05%	80.43%
10 50	1089	157	224	280	79.55%	87.40%	55.56%	82.94%
10 60	1106	176	207	261	80.91%	86.27%	55.77%	84.23%
100 50	1176	187	137	250	82.47%	86.28%	64.60%	89.57%
15 10	1188	191	125	246	82.85%	86.15%	66.31%	90.48%
15 15	1211	205	102	232	83.92%	85.52%	69.46%	92.23%
15 20	1132	184	181	253	81.73%	86.02%	58.29%	86.21%
15 25	1129	183	184	254	81.63%	86.05%	57.99%	85.99%
15 30	1133	187	180	250	81.92%	85.83%	58.14%	86.29%
15 35	1131	186	182	251	81.84%	85.88%	57.97%	86.14%
15 40	1205	200	108	237	83.56%	85.77%	68.70%	91.77%
15 60	1190	189	123	248	82.75%	86.29%	66.85%	90.63%
19 38	1189	192	124	245	82.91%	86.10%	66.40%	90.56%
20 10	1188	190	125	247	82.79%	86.21%	66.40%	90.48%
20 15	1168	176	145	261	81.74%	86.90%	64.29%	88.96%
20 20	1187	191	126	246	82.83%	86.14%	66.13%	90.40%
20 25	1199	194	114	243	83.15%	86.07%	68.07%	91.32%
20 30	1132	187	181	250	81.91%	85.82%	58.00%	86.21%
20 35	1184	191	129	246	82.80%	86.11%	65.60%	90.18%
20 40	1182	190	131	247	82.72%	86.15%	65.34%	90.02%
20 50	1205	201	108	236	83.62%	85.70%	68.60%	91.77%
20 60	1177	186	136	251	82.42%	86.35%	64.86%	89.64%
23 33	1188	189	125	248	82.73%	86.27%	66.49%	90.48%
25 10	1216	205	97	232	83.98%	85.57%	70.52%	92.61%
25 15	1208	200	105	237	83.60%	85.80%	69.30%	92.00%
25 20	1171	182	142	255	82.12%	86.55%	64.23%	89.19%
25 25	1199	195	114	242	83.21%	86.01%	67.98%	91.32%
25 30	1206	198	107	239	83.46%	85.90%	69.08%	91.85%
25 35	1188	196	125	241	83.14%	85.84%	65.85%	90.48%
25 40	1206	198	107	239	83.46%	85.90%	69.08%	91.85%
25 45	1176	186	137	251	82.41%	86.34%	64.69%	89.57%
27 43	1223	210	90	227	84.34%	85.35%	71.61%	93.15%
30 10	1177	187	136	250	82.48%	86.29%	64.77%	89.64%
30 15	1172	185	141	252	82.30%	86.37%	64.12%	89.26%
30 20	1225	206	88	231	84.13%	85.60%	72.41%	93.30%

30 25	1216	204	97	233	83.92%	85.63%	70.61%	92.61%
30 30	1183	187	130	250	82.55%	86.35%	65.79%	90.10%
30 35	1211	202	102	235	83.75%	85.70%	69.73%	92.23%
30 40	1241	217	72	220	84.94%	85.12%	75.34%	94.52%
30 45	1240	214	79	223	84.76%	85.28%	73.84%	94.01%
30 45	1189	191	124	246	82.86%	86.16%	66.49%	90.56%
30 50	1215	197	98	240	83.51%	86.05%	71.01%	92.54%
30 60	1219	209	94	228	84.24%	85.36%	70.81%	92.84%
35 10	1194	198	119	239	83.32%	85.78%	66.76%	90.94%
35 15	1224	203	89	234	83.95%	85.77%	72.45%	93.22%
35 20	1227	212	86	225	84.50%	85.27%	72.35%	93.45%
35 25	1227	215	86	222	84.68%	85.09%	72.08%	93.45%
35 30	1193	192	120	245	82.96%	86.14%	67.12%	90.86%
35 35	1211	200	102	237	83.63%	85.83%	69.91%	92.23%
35 40	1191	189	122	248	82.77%	86.30%	67.03%	90.71%
35 45	1228	212	85	225	84.51%	85.28%	72.58%	93.53%
35 50	1223	200	90	237	83.77%	85.95%	72.48%	93.15%
40 10	1135	183	178	254	81.71%	86.12%	58.80%	86.44%
40 20	1193	191	120	246	82.90%	86.20%	67.21%	90.86%
40 25	1168	179	145	258	81.91%	86.71%	64.02%	88.96%
40 30	1219	208	94	229	84.19%	85.42%	70.90%	92.84%
40 35	1233	213	80	224	84.63%	85.27%	73.68%	93.91%
40 40	1199	197	114	240	83.32%	85.89%	67.80%	91.32%
40 45	1193	194	120	243	83.08%	86.01%	66.94%	90.86%
40 50	1198	197	115	240	83.31%	85.88%	67.61%	91.24%
40 60	1212	200	101	237	83.64%	85.84%	70.12%	92.31%
43 27	1205	199	108	238	83.51%	85.83%	68.79%	91.77%
45 25	1180	186	133	251	82.46%	86.38%	65.36%	89.87%
50 15	1210	198	103	239	83.51%	85.94%	69.88%	92.16%
50 20	1200	197	113	240	83.33%	85.90%	67.99%	91.39%
50 25	1195	193	118	244	83.04%	86.10%	67.40%	91.01%
50 30	1117	176	196	261	81.06%	86.39%	57.11%	85.07%
50 35	1200	201	113	236	83.57%	85.65%	67.62%	91.39%
50 40	1189	191	124	246	82.86%	86.16%	66.49%	90.56%
50 45	1211	199	102	238	83.57%	85.89%	70.00%	92.23%
50 50	1217	207	96	230	84.11%	85.46%	70.55%	92.69%
50 60	1180	188	133	249	82.58%	86.26%	65.18%	89.87%
60 10	1227	212	86	225	84.50%	85.27%	72.35%	93.45%
60 15	1208	198	105	239	83.48%	85.92%	69.48%	92.00%
60 20	1174	186	139	251	82.39%	86.32%	64.36%	89.41%
60 30	1191	190	122	247	82.82%	86.24%	66.94%	90.71%
60 40	1209	196	104	241	83.38%	86.05%	69.86%	92.08%
80 50	1117	170	196	267	80.71%	86.79%	57.67%	85.07%