A Comparative Study of Machine Learning Techniques for Classifying Type II Diabetes Mellitus

Moeketsi Ambrose Ndaba

209503432

A thesis submitted to
the University of KwaZulu-Natal,
College of Agriculture, Engineering and Science,
in fulfillment of the requirements for the degree of

Master of Computer Science

Supervisor: Anban Pillay
Co-Supervisor: Absalom E. Ezugwu

School of Mathematics, Statistics and Computer Science

University of KwaZulu-Natal

June 2018

I, Moeketsi Ambrose Ndaba , declare that:

(i) The research reported in this thesis, except where otherwise indicated, is my original research.

(ii) This thesis has not been submitted for any degree or examination at any other university.

(iii) This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

(iv) This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

a) their words have been re-written but the general information attributed to them has been referenced:

b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced.

(v) This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation/thesis and in the References sections.

Candidate: Moeketsi Ambrose Ndaba

Signature: M.A. Ndaba

As the candidate's supervisor I agree to the submission of this thesis for examination.

Supervisor: Anban Pillay

Signature: _____

ABSTRACT

Moeketsi Ambrose Ndaba
School of Mathematics, Statistics and Computer Science
Master of Computer Science

Diabetes is a metabolic disorder that develops when the body does not make enough insulin or is not able to use insulin effectively. Accurate and early detection of diabetes can aid in effective management of the disease. Several machine learning techniques have shown promise as cost effective ways for early diagnosis of the disease to reduce the occurrence of health complications arising due to delayed diagnosis. This study compares the efficacy of three broad machine learning approaches; viz. Artificial Neural Networks (ANNs), Instance-based classification technique, and Statistical Regression to diagnose type II diabetes. For each approach, this study proposes novel techniques that extend the state of the art. The new techniques include Artificial Neural Networks hybridized with an improved K-Means clustering and a boosting technique; improved variants of Logistic Regression (LR), K-Nearest Neighbours algorithm (KNN), and K-Means clustering. The techniques were evaluated on the Pima Indian diabetes dataset and the results were compared to recent results reported in the literature. The highest classification accuracy of 100% with 100% sensitivity and 100% specificity were achieved using an ensemble of the Boosting technique, the enhanced K-Means clustering algorithm (CVE-K-Means) and the Generalized Regression Neural Network (GRNN): B-KGRNN. A hybrid of CVE-K-Means algorithm and GRNN (KGRNN) achieved the best accuracy of 86% with 83% sensitivity. The improved LR model (LR-*n*) achieved the highest classification accuracy of 84% with 72% sensitivity. The new multi-layer perceptron (MLP-BPX) achieved the best accuracy of 82% and 72% sensitivity. A hybrid of KNN and CVE-K-Means (CKNN) technique achieved the best accuracy of 81% and 89% sensitivity. CVE-K-Means technique achieved the best accuracy of 80% and 61% sensitivity. The B-KGRNN, KGRNN, LR-*n*, and CVE-K-Means technique outperformed similar techniques in literature in terms of classification accuracy by 15%, 1%, 2%, and 3% respectively. CKNN and KGRNN technique proved to have less computational complexity compared to the standard KNN and GRNN algorithm. Employing data pre-processing techniques such as feature extraction and missing value removal improved the classification accuracy of machine learning techniques by more than 11% in most instances.

LIST OF PUBLICATIONS

Moeketsi Ambrose Ndaba
School of Mathematics, Statistics and Computer Science
Master of Computer Science

1. Moeketsi Ndaba, Anban Pillay, and Absalom E. Ezugwu, *An Improved Generalized Regression Neural Network for Type II Diabetes Classification*, paper was submitted to LNCS conference proceedings (SEPA) 2018 and accepted on 27 April 2018 as LNCS paper.

ACKNOWLEDGMENTS

I would like to thank God for giving me strength to work and finish this thesis through challenging times. I also would like to thank my supervisors Anban Pillay and Absalom E. Ezugwu for their immense contribution, their guidance and engagement during the production of this thesis. Lastly but not least, I would like to extend my gratitude to my wife, my friends and family for their encouragement and their belief in me.

I dedicate this thesis to my late initial supervisor Prof. A.O. Adewumi who did not get a chance to see me complete this study. I will always be grateful for his genuine interest in my work and for always being available to give me guidance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The World Health Organization (WHO) lists diabetes as one of the most widespread and challenging chronic diseases in the world [60]. The lack of early diagnosis often makes it dangerous because in most instances it is diagnosed too late for effective management. Diabetes is a metabolic disorder which affects the way the body uses digested food for energy. The digestive tract breaks down carbohydrates into glucose. Glucose is a form of sugar that enters the bloodstream after carbohydrates are broken down by an enzyme. With the help of the hormone insulin, cells throughout the body absorb glucose and use it for energy. Diabetes develops when the body does not make enough insulin or is not able to use insulin effectively, or both [2]. There are three types of diabetes: type I (insulin-dependent diabetes), type II diabetes (non-insulin-dependent diabetes), and gestational diabetes which affects pregnant women. Type I occurs when the body is not able to produce enough insulin, while type II diabetes develops when the body is not able to use insulin the right way. Diabetes ultimately leads to many life threatening complications such as heart disease, visual impairment and kidney failure to mention few.

It is estimated that 90-95% of people living with diabetes have type II diabetes. The International Diabetes Federation (IDF) estimates that more than 425 million people have been diagnosed with diabetes and 79% of them live in low and middle income countries [136]. It is estimated that South

East Asia and Western pacific regions have the highest number of diabetic people in the world (241 million), followed by Europe with 58 million people, and Africa with 55 million people. It is estimated that by 2045 the number of people living with the disease will rise to 629 million [136]. In 2014, approximately 4.9 million people died due to diabetes related illnesses [4].

In Sub-Saharan Africa, diabetes used to be considered an uncommon disease, "a disease only contracted by those in wealthy countries". This has significantly changed over the last few years and the disease is becoming more common in all countries in Sub-Saharan Africa. For example in 2014, 2.7 million people in South Africa were estimated to be diagnosed with type II diabetes and approximately 70,000 of them died due to health complications as a consequence [5]. The IDF states that in low and middle income countries, two thirds of people living with diabetes have inadequate control and management methods of the disease due to limited access to anti-diabetic treatments such as insulin injections and limited access to quality professional health assistance. It is estimated that 70% of people with diabetes in Africa are unaware of their diabetic state. This is due to limited resources and low prioritization of diabetes screening [136].

In Sub-Saharan Africa, especially in rural communities, effective prevention and diabetes management is slow because of difficulty in reaching testing and treatment centres due to transportation constraints and affordability of medicines for which individuals usually have to pay for. The global health care expenditure on diabetes has increased from 232 billion USD in the year 2006 to 700 billion USD in 2017 [136]. The rapid global increase in number of diabetic people and expenditure on diabetes related illnesses shows that traditional methods for diagnosing and containing the spread of diabetes are not effective. Health professionals and researchers are in agreement that the most effective way of controlling a disease is early detection, immediate commencement of treatment and active awareness about the disease. Therefore, there is a need for accurate, quick and easy to access tools that can aid in the diagnosis and early detection of diabetes in order for health professionals to efficiently manage it. Research has shown that 80% of complications related to

type II diabetes can be delayed from starting or be completely prevented by early identification of people who are at risk [7].

Diabetes diagnosis is a data classification problem in ML. Data classification is a technique that is employed in ML to predict or allocate a class label to an instance in a dataset based on a given input. In order to classify an instance, a classifier processes a dataset known as a training set to uncover patterns or relationships in a set of instance attributes. Once the classifier "knows" patterns or relationships in the data well enough, a test dataset with unseen instances is introduced to the classifier so that it can classify the instances based on the knowledge it gained from "learning". The performance of the classifier is then evaluated by how many instances in the test set it classified correctly. This allows other real-world problems to be formulated as a classification problem in different domains such as fraud detection, manufacturing, and customer segmentation to mention few.

1.1 Problem Statement

The diagnosis of type II diabetes using ML techniques and patient data collected from clinical trials and routine health check-ups is an important problem which still requires extensive research since many studies in literature have not yet found optimal results - i.e., 100% diagnosis accuracy. This is due to different factors such as the size and quality of datasets used by the researchers, their choice of ML techniques to apply, and ML technique parameters to mention few. These factors make it very challenging to obtain optimal results since they have to be carefully considered jointly. This study attempts to improve the effectiveness of three types of ANNs, two Instance-based classification techniques, and a Logistic Regression Model with the hope to find optimal results. The influence of missing data and the lack of data preprocessing in diabetes diagnosis accuracy is an area that needs to be researched further since it has not been thoroughly reviewed in literature. This study also intends to perform this thorough evaluation.

1.2 Purpose of The Study and Motivation

The purpose of the research is to implement and evaluate novel ML techniques to the type II diabetes classification problem. The motivation of the research is to find more efficient and accurate ML techniques for type II diabetes classification - i.e, techniques that outperform those in literature. Due to the rapid increase in undiagnosed diabetes cases and the number of people living with the disease, there is an urgent need for alternative tools that are more accurate and easily accessible to health professionals and patients, especially those in the poorest communities. The need for ML techniques that are more accurate is also evident in studies that were conducted in literature which discovered that the average type II diabetes diagnosis accuracy of ML classification techniques is between 75 and 80% [18,20,54].

1.3 Research contribution

The key contribution of this study is the use of 6 novel ML techniques to improve the classification accuracy obtained in previous studies in the diabetes classification problem. The novel ML techniques that are proposed in this study can also be used to solve other binary classification problems in different domains. The enhancements on ML techniques proposed in this thesis are an original work of this study and to the best of the researcher's knowledge they have never been applied in other studies. This study also proposes different ways to deal with missing values in the dataset and how to improve the quality of the dataset in order to harness the full potential of ML techniques.
The novel ML techniques proposed in this study can be used to implement tools that will allow people to perform self-assessment of their type II diabetes state. This will in turn reduce the high number of undiagnosed cases, save health care related costs (transportation costs for patients, consultation fees, etc.) and it will also save time for both patients and health care providers especially

in developing countries where there is a shortage of health professionals and resources.

1.4 Research Methodology

In this study, the application of ANNs, Statistical Regression, Instance-based techniques, and other ML techniques in previous studies on the diabetes classification problem will be reviewed. After that, the diabetes dataset from the University of California Irvine (UCI) ML repository will be thoroughly explored and presented. Several data preprocessing and feature extraction methods for improving the quality of the dataset and the diabetes classification accuracy will then be discussed and applied on the dataset to produce new datasets for training and testing the ML techniques proposed in this study. Thereafter, the enhancements on the explored ML techniques will be proposed and outlined. These enhanced techniques will then be tested and the experimental results will be used to evaluate their performance. The experimental results will also be used to determine the influence of data preprocessing techniques on the performance of the techniques. Furthermore, these results will be used to compare the classification accuracy of the novel ML techniques proposed in this study with other techniques in literature. After that, the results and their implications will be discussed. Finally, the conclusions and future works on the conducted research will be provided.

1.5 Research Scope and Limitations

This study only focuses on the classification of Type II diabetes. Type I and gestational diabetes are not part of this study. The results obtained in this study are based on the Pima Indian diabetes dataset and datasets that were derived from it. These datasets do not represent the diabetic state and properties of the entire population in the world. The datasets used in this study have between 350 and 768 records. The size of the datasets could also impact the performance of the proposed ML techniques since some techniques need larger datasets to learn effectively. Therefore, it would

be beneficial to have larger datasets that are obtained from different countries and ethnic groups to ensure that the results obtained with the proposed ML techniques are true for the entire population in the world. Most studies in literature on the diabetes classification problem did not provide the results of other algorithm performance evaluation methods except the classification accuracy. This made it difficult to thoroughly compare the performance results and the strength of the techniques that were proposed in literature in classifying diabetes with those proposed in this study.

1.6 Overview of ML Techniques for Medical Diagnosis

Lazy learning (Instance-based) and eager learning ML techniques can be used to solve the data classification problem. Lazy learning techniques simply store the training data or partially process the training data and then waits for the test data to be provided in order to perform classification. On the other hand, eager learning techniques use a set of training data to construct a classification model that can be used to classify a new test data that was not used in model construction. Lazy learning takes less computational time to train and more computational time to perform classification of unseen data while eager learning takes more computational time building a model and less computational time performing classification of unseen data. Lazy learning techniques are found to be very resource intensive in many real-world applications which have enormous amounts of data to be classified since all training instances need to be stored and retrieved for classification. KNN is one of the most popular lazy learning techniques. Artificial Neural Networks and Logistic Regression are examples of eager based learning techniques.

Artificial Neural Networks are biologically inspired computer programs designed to simulate the way in which the human brain processes information [76]. The key element of ANNs is their information processing structure that is composed of a large number of immensely interconnected processing units called neurons that are collectively working together towards solving a given

problem. The neurons are interconnected through weights (coefficients) that are organised in layers. Every neuron has inputs, an activation function and one output. One example of ANNs is the GRNN which is a special case of the Radial Basis Function Network. GRNN draws the estimation of the function directly from the training data [79]. GRNN does not require an iterative learning process like the multi-layer perceptron that has to adjust its neuron connection weights in order to reduce the classification error.

K-Nearest Neighbours Algorithm classifies instances based on their similarity or their closeness to training examples that are in a feature space. The closest training examples to a test instance are called neighbours. KNN picks the "popular" class among the neighbours and classifies the test instance as belonging to that class.

In K-Means Clustering when a new instance is introduced to be allocated to a cluster, it is allocated into a cluster in which its centroid (cluster centre) is the nearest to the instance. The selection of initial cluster centres and the number of clusters in this algorithm is one of the critical things that should be carefully done since one can get different results depending on the number of clusters formed and the initial cluster centres that were selected.

Logistic Regression is a regression technique that is used to model a relationship between one dependent variable and one or more independent variables. In Logistic Regression, the dependent variable takes a categorical value either "0" or "1" to represent outcomes such as diabetic/not diabetic, alive/dead, etc. The independent variables can either be discrete or continuous variables.

1.7 Thesis Layout

This section provides a summary of chapters in this thesis.

- **Chapter 2: Literature Review**

  This chapter provides a thorough comparative study of several diverse approaches employed

in previous studies to solve the diabetes classification problem. The study outlines the approaches proposed in literature, their strengths, their weaknesses and their results.

- **Chapter 3: Methods**

  This chapter provides a detailed description and technical details of datasets, data preprocessing methods and novel ML techniques proposed in this study for diabetes classification.

- **Chapter 4: Experimental Results**

  The results obtained using the proposed methods in chapter 3 are presented and compared with results obtained in literature by similar techniques.

- **Chapter 5: Discussion**

  In this chapter, the interpretation of experimental results and their implications is presented. The the impacts of the dataset preprocessing techniques used and the limitations of methods proposed in chapter 3 are also discussed in this chapter. A comparative analysis on the performance of the proposed Machine Learning techniques and those in literature is also provided.

- **Chapter 6: Conclusion and Future Works**

  A summary of findings of this study, research conclusions and future work are presented in this chapter.

Chapter 2

Literature Review

Diabetes is diagnosed by considering several factors or features. This allows ML techniques to be employed as alternative methods for analyzing these features from the data and be able to make classifications that can aid in the diagnosis by physicians. Many studies have been done on this subject and they all propose different approaches for performing type II diabetes classification. These approaches are also continually improved by other scholars in order to be able to detect the presence of diabetes more accurately and as efficiently as possible. Most diabetes classification models that have been researched in previous studies are based on different implementations of supervised and unsupervised learning techniques such as Artificial Neural Networks, K-nearest Neighbour Algorithm, Varying implementations of Decision Tree Algorithm, and Support Vector Machines to mention few. This chapter provides an overview of the work done in previous studies of type II diabetes classification using ML techniques.

2.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks have been extensively studied and applied to the diabetes diagnosis problem. This is because of its many advantages such as its ability to detect non-linear relationships

between dependent and independent variables, fault tolerance, missing data tolerance, the ability to detect all possible relationships between predictor variables and having a wide range of training algorithms [76, 78, 93, 136]. ANNs have many different architectures and learning methods. One popular type of ANN is the Multi-Layer Feed-Forward ANN which uses a back propagation learning algorithm. This ANN is called a Multi-Layer Back-Propagation Network (ML-BPN). ML-BPNs have proved to be very effective in the implementation of disease diagnosis systems and have been successfully used in replacing conventional pattern recognition methods.

In 2012, Mehmet et al. [55] conducted a diabetes classification study using 6 different types of ANNs namely: Probabilistic Neural Network (PNN), Learning Vector Quantization (LVQ), Feed-Forward Network (FFN), Cascade-Forward Networks (CFN), Distributed Time Delay Networks (DTDN), and Time Delay Networks (TDN). Mehmet et al. [55] used the Pima Indian diabetes dataset to train and test the ANNs. The best classification accuracy of 76% was achieved by the DTDN, followed by LVQ with a classification accuracy of 73%, and PNN with 72% accuracy. The DTDN performed better than other ANNs because it incorporated knowledge from previous iterations to learn better. FFN and CFN both achieved 68% accuracy. TDN achieved the lowest accuracy of 66%. The best sensitivity of 63% was achieved by the PNN, while the best specificity of 89% was achieved by DTDN. Mehmet et al. [55] used the dataset for training and testing the ANNs without preprocessing it, this contributed in obtaining average results. The networks failed to deal with a slight class imbalance in the dataset. One can observe that at there is a huge gap between the maximum sensitivity and specificity values which were obtained by the networks. All networks classified a high number of negative instances than positive instances.

Kayaer and Yildirim [79] used a Lavenberg-Marquardt (LM) training algorithm for the ML-BPN on Pima Indian dataset. They achieved an accuracy of 77%. Kayaer and Yildirim [79] also proposed a GRNN and a RBF network. Their GRNN and RBF network achieved the highest accuracy of 80% and 68% respectively. For GRNN and RBF network, the optimal spread values were found

by trial-and-error. The performance of RBF was found to be worse than the ML-BPN for all spread values tried. Although the LM algorithm is known to help ANNs converge faster and find better results, the ML-BPN performed worse than the GRNN technique. This could be due to the selection of training parameters such as momentum and learning rate for the ML-BPN. Temurtas et al. [137] proposed the ML-BPN and PNN, both trained using the LM algorithm in 10-fold cross validation. Their ML-BPN achieved the classification accuracy of 82%, while the PNN achieved the accuracy 78%. The performance of the ML-BPN proposed by Temurtas et al. [137] is better than that of the similar approach proposed by Kayaer and Yildirim [79]. This can be due to [79] not addressing the memorization effect of the LM algorithm when the overtraining occurs. The LM algorithm converges very fast, when overfitting occurs it starts to memorize the learning data which decreases the ability of the network to generalize. Adeyemo and Akinwonmi [102], employed GRNN and PNN to classify diabetes using patient data obtained from Family Medicine Clinic in University Teaching Hospitals of Nigeria. Both networks used tanh as the activation function. Their GRNN achieved a classification accuracy of 84% while PNN achieved an accuracy of 76%.

One of the drawbacks of ML-BPNs is their slow convergence rate that makes them achieve sub-optimal results because of the use of gradient optimization algorithm to update their weights [13]. Many algorithms have been introduced to address the slow convergence rate of the ML-BPN. A recent study conducted by [13] used a LM training algorithm to address the slow convergence rate. LM provided faster convergence and a classification accuracy of 91% was achieved [13]. Karegowda et al. [14] used a hybrid of a Genetic Algorithm (GA) and ML-BPN to classify diabetes. One of the drawbacks of ML-BPN they identified was that the gradient descent method used to train the network took too long to produce a solution [14]. Karegowda et al. [14] suggested that instead of employing gradient-based learning techniques, one can apply other techniques such as the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and the Ant Colony Optimization (ACO) algorithm to find the best network weights. The GA was used to initialize and optimize

the connection weights of the ML-BPN, such as the number of hidden layers, feature selection of relevant subsets, the learning rate, and optimal initial connection weights. Features were also identified by [14] through applying Decision Tree and GA-CFS (Correlation Feature Selection) technique. These methods were used as input to the hybrid model of the ML-BPN and GA to classify diabetes. A classification accuracy of 84% was achieved by this hybdrid model. Priya et al. [15] proposed the use of a hybrid classification model using K-Means clustering and C4.5 classifier which resulted in 92% classification accuracy. This method was further enhanced by using ANNs for classification. This enhancement improved the classification accuracy to 97%.

The studies in literature show that the choice of the learning algorithm, training parameters, and feature selection techniques play an important role in obtaining good results. The studies also show that hybridizing ANNs with other techniques helps to obtain improved results. The lack of dataset preprocessing can make good algorithms perform poorly. It is always recommended to preprocess the dataset before using it.

2.2 K-Nearest Neighbour Algorithm (KNN)

A notable strength of KNN is its ability to be combined with other algorithms for better accuracy. Researches have harnessed this advantage to provide efficient solutions to classification problems. KNN is a simple, yet powerful Instance-based learning algorithm that is well known for its highly effective method of inductive inference for noisy training data and complex target functions [8]. Furthermore, it fully exploits the available local information to form highly linear and adaptive decision boundaries for each object. The disadvantage of the KNN algorithm is that it takes a long time to classify test instances even though the learning is quick and simple [8]. Another drawback is that it requires the whole training set to be stored in memory and this tends to be costly when a training set is large. Various studies were conducted to address issues associated with large training datasets on the KNN [24, 25, 26, 27].

Vijayan et al. [22] conducted a study on the use of the traditional KNN, K-Means clustering algorithm and amalgam KNN for diabetes classification using the Pima Indian dataset. The amalgam algorithm was a hybrid of KNN and K-Means clustering algorithm with some additional processing. This algorithm was designed to harness the strengths of KNN and K-Means clustering to improve the classification accuracy even for large number of data sets. Their KNN algorithm achieved 73% classification accuracy. Their K-Means clustering algorithm achieved 77% accuracy, and the amalgam KNN algorithm achieved the accuracy of 80%. The amalgam KNN used K-Means clustering to identify and remove instances that were erroneously classified by KNN. The dataset was also preprocessed before applying the amalgam KNN to reduce noise in the data. This made the algorithm obtain improved results. KNN and K-Means clustering algorithm did not perform well because of the distance measure used, the choice of k values, and the use of un-preprocessed noisy data.

To address the memory and high computation issue of KNN algorithm, P. Hart [27] proposed the use of a local search method called Condensed Nearest Neighbour (CNN). CNN only stores a subset of the training set for classification. P. Hart [27] assumed that all patterns in the training set could be similar and some were highly unlikely to add any additional information for improving the classification accuracy. G. Gates [24] developed a Reduced Nearest Neighbour (RNN) method to further reduce the subset. RNN removed all those elements in a subset that would not lead to an error in classification after their removal. They concluded that the RNN provided a notable benefit by reducing the execution time of a CNN and this contributed to improved classification. The disadvantage of using these approaches is that they are not effective when the dataset is very small.

A classification accuracy rate of 89.1% with k = 5 and 87.3% with k = 3 was achieved by Saxena et al. [8] on a dataset from the LARS project at Stanford University. Saxena et al. [8] stated that during the experiment, the results showed that as the value of k increased beyond 5, the accuracy

rate decreased. Hence, the selection of the best k was crucial when using KNN. Karegowda et al. [20] used Cascading K-Means clustering, GA and KNN to predict diabetes. The classification accuracy achieved was 97% with k = 5. Karegowda et al. [20] approached the problem by creating a model that used K-Means clustering to identify and remove all training instances that were not classified correctly. After that they employed GA and CFS in a cascaded fashion for relevant feature extraction. KNN was then used for classification by taking each test instance that was correctly clustered in K-Means clustering process with a feature subset identified during feature extraction process as inputs for the KNN.

The studies show that the selection of the parameter k for KNN has a significant influence on the performance of the algorithm. The KNN algorithm on its own achieved average results on the diabetes classification problem. However, when it was used in hybrid models, its strengths were harnessed and improved results were obtained. Similarly for K-Means clustering algorithm. Feature extraction and dataset preprocessing also proved to be useful in obtaining improved results.

2.3 Decision Tree Algorithm (DT)

The DT technique is a commonly used ML technique for developing classification systems based on multiple covariates or for developing classification algorithms for a target variable [30]. In a DT, a set of objects are classified into branch-like segments mimicking an inverted natural tree such that there is a root represented by a root node, branches represented by internal nodes and leaves represented by leaf nodes. In a DT, each internal node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf node denotes a class label [30]. To classify an instance, the attribute-vector of an instance is computed and applied to the tree.

There are various DT algorithms such as ID3, C4.5, C5, J48, CART and CHAID to mention a few. One of the benefits of the DT is that it does not change in structure no matter how large a dataset might be and it can efficiently deal with large complicated data sets. DT allows an exploration of

many possible outcomes of a decision and weigh these outcomes against each other to decide the best to minimize chances of selecting the worst decision as a solution. Recently there is a surge in application of DT algorithms in medical research. An example of medical use of the Decision Tree algorithms is in the diagnosis of a medical condition by analyzing patterns and relationships in the attributes of a dataset.

Habibie et al. [31] investigated the use of the J48 algorithm to create a classification model for diabetes. This technique was applied on the dataset that was obtained from a database in a diabetes control system in Tabriz, Iran. The dataset was collected from people who were referred for diabetes screening between 2009 and 2011. The J48 algorithm achieved a classification accuracy rate of 72%. J48 algorithm used WEKA's implementation of C4.5 for building the decision tree. One of the interesting things about the classification research done by Habibie et al. [31] was the exclusion of laboratory tests for classification. The results of collected attributes such plasma glucose and 2-Hour Serum Insulin were excluded. J48 was still successful without considering these attributes. Habibie et al. [31] admitted that the exclusion of laboratory tests features decreased the sensitivity and precision of the model compared to other models explored in literature. An improved version of the J48 algorithm was used by Kaur et al. [32] in the classification of diabetes using the Pima Indian dataset. Table 2.1 provides a list of comparative results that were obtained.

Algorithm	Accuracy
Improved J48	**99%**
MultiClassClassifier	77%
NaiveBayes	76%
LADTree	74%
Multilayer Perceptron	73%
RandomForest	73%
Standard J48	73%
BFTree	73%
ADTree	72%
RandomTree	68%
REPTree	68%

Table 2.1 Findings by Kaur et al. [32]

J48 is an extension of ID3 algorithm and it has additional features that deal with missing values, decision trees pruning, and derivation of rules among others [32]. These additional features and the use of C4.5 algorithm helped the improved algorithm to perform better than the original J48. Bashir et al. [36] investigated the combination of three different decision tree algorithms: ID3, C45 and CART. The decision trees were used as base classifiers in their experiment. Ensemble techniques were used to combine ID3, C45 and CART. Ensemble techniques which were applied are: Majority Voting, Adaboost, Bayesian Boosting, Stacking and Bagging. The ensemble that produced the highest accuracy in classification was selected as a solution. The combination of these three techniques was tested on Pima Indian dataset and the diabetes dataset from the BioStat repositories. The highest accuracy of 91.56% using Bagging ensemble method was achieved. This result was better than the accuracy achieved by each algorithm in different studies conducted by K. Sangeetha

[40] using C4.5 which achieved 91% accuracy, Ashwinkumar [37] using CART achieved 44.44% and his C4.5 achieved which 68% accuracy. The study by Bashir et al. [36] demonstrated the power of combining multiple powerful algorithms to unleash their unique strengths in order to achieve a classification accuracy that is greater than each algorithm's individual accuracy.

2.4 Support Vector Machines (SVM)

The SVMs are supervised learning models with associated learning algorithms that can efficiently analyse data and recognize patterns that can be used for classification and regression [10]. SVMs are usually referred to as a non-probabilistic binary linear classifiers because they can classify training patterns as belonging to one of the two classes say *"C1"* and *"C2"*. SVMs seek a hyper-plane that can efficiently separate data points of two classes given the labels say $y_i = +1,-1$. The SVM model learns by representing training instances as mapped points in an n-dimensional space with a clear separation that divides instances of separate categories. To classify new instances, SVM maps them into the same space as the training set and then classifies them based on which side of the boundary they belong to [10]. SVMs are known to have less over-fitting because of their robustness to noise and their learning task is not sensitive to the relative number of training instances in the positive and negative classes [41]. Another powerful strength of SVMs is that they can perform non-linear classification tasks by using a kernel to implicitly map their inputs into high dimensional feature spaces efficiently [41, 42]. Due to their efficient classification capability, SVMs have recently became very popular [41].

K. Polat et al. [43] used a combination of Generalized Discernment Analysis (GDA) technique with Least Square Support Vector Machine (LSSVM) to do classification. The model achieved a classification accuracy of 79.16%. LSSVM was used with a Radial Basis Function (RBF) kernel. Tsyurmasto et al. [44] applied a robust version of SVM based on Value-at-Risk (VaR) measure. This technique was called VaR-SVM. VaR-SVM achieved an accuracy rate of 97.83%. The idea

behind this model was to create a classifier that is stable to data outliers unlike other SVM models such as hard-margin SVM, soft-margin SVM, and v-SVM. Purnami et al. [45] approached the problem by enhancing the Smooth SVM (SSVM) using a Multiple Knot function as a smooth function instead of a default-plus function of a SSVM. A multiple Knot function is the modification of the three order function [45]. This enhanced SSVM achieved a classification accuracy of 93.2%. Zolfaghari [47] proposed a three-layer hierarchy multi-classifier using SVM with ensemble ML-BPN. This technique achieved an accuracy of 88%. Although SVMS show good performance, they are very dependent on a selection of a proper kernel function, and they have high memory utilization and CPU time.

2.5 Swarm Intelligence (SI)

The ANNs, DTs, SVMs and KNN algorithms are by far the most popular applied ML techniques for solving the diabetes classification problem. In most instances, SI techniques are usually used to enhance the classification accuracy of these techniques. SI is based on social behavior of organisms like ant colonies, animal herding, and bacterial growth. SI can be used as a problem solving technique that encompasses the implementation of a collective intelligence of groups of simple agents which are based on the behavior of real world insect swarms [142]. *The collective intelligence of agents satisfies cooperative transportation, division of labor, nest-building of social insects, collective sorting and clustering* [142]. Mishra et al. [42] proposed the use of a hybrid of SVM and ACO, and a hybrid of Naive Bayes with ACO. The hybrid of SVM and ACO produced best results compared to the hybrid of Naive Bayes and ACO. SVM and ACO achieved the classification accuracy of 85% while the Naive Bayes and ACO achieved 80% accuracy. Mishra et al. [42] noted that as the number of generations in GP went above 125, the created trees became too big and difficult to handle. Mishra et al. [42] used the ACO technique to remove redundant attributes in the dataset and then used Naive Bayes and SVM for classification.

Aslam and Nandi [50] created a classification model using Genetic Programming (GP) and they achieved an accuracy of 75%. Aslam and Nandi [50] then proposed a hybrid of a GP algorithm and a variation of GP algorithm called GP with Comparative Partner Selection (CPS). They split their algorithm to solve the classification problem in two stages. In the first stage they used the GP algorithm to convert original features for each instance in the training set into a single feature that had different values for diabetic and non-diabetic patients. The second stage involved classifying the test dataset. Their hybrid algorithm achieved the classification accuracy of 79%.

2.6 Logistic Regression (LR)

Nai-Aruna and Moungmaia [87] proposed the use of Logistic Regression and a hybrid of Logistic Regression with ensemble methods. The selected ensemble methods were Boosting and Bagging. Ensemble methods are designed to improve the stability and the accuracy of ML algorithms. Nai-Aruna and Moungmaia [87] utilized a dataset that was obtained from 26 Primary Care Units (PCU) of the Sawanpracharak Regional Hospital in Thailand. The standard LR model proposed by [87] achieved the highest classification accuracy of 82.308%. Their hybrid of LR and Boosting and the hybrid of LR and Bagging achieved 82.312% and 82.318% respectively. These results showed that ensemble methods were not effective in significantly improving the classification accuracy of the LR model. Meng et al. [91] used a combination of LR and cross validation technique to classify diabetes using a dataset collected from 1487 individuals living in Zhuguang, China. Their LR model achieved a classification accuracy of 77% with 80% sensitivity and 72% specificity. Bassam et al. [143] employed LR to classify diabestes using data from Kuwait Health Network which integrated patient data from primary health centres and hospitals in Kuwait. Bassam et al. [143] observed that ethnicity was a major contributor in making a more accurate model. They noticed that Kuwaiti natives and Asian expatriates had significant differences in prevalence and in trends associated with features (such as age at onset and body mass index) of diabetes. As a result,

they decided to also implement separate LR models using the Kuwaiti natives dataset and the Asian expatriates dataset. The results showed that the two LR models did not perform equally well for the two ethnicities. They obtained 84% classification accuracy on Asian expatriates dataset while they obtained 79% on Kuwaiti natives dataset. The classification accuracy of 81% was obtained using the overall dataset which was comprised of both ethnicities.

Devi et al. [144] proposed the use of two LR models on the dataset from clinical laboratory of AR Hospital in Madurai, India. The first LR model was a standard LR model using the sigmoid activation function and regression coefficients. The second LR model used a Bipolar Sigmoid Function (BSF) which was calculated using a Neuro-based Weight Activation Function (WAF) instead of the sigmoid function to improve the results. A Distance-based Outlier Detection (DBOD) technique was employed to preprocess the dataset. Their standard LR model achieved a classification accuracy of 79%, while the improved LR model achieved an accuracy of 90%.

2.7 Rule-based Systems

Rule-based systems have been successfully employed to solve many problems encountered in science, economics and engineering [117]. Rule-based systems are used as a way to store and manipulate knowledge to interpret information in a meaningful way.

Hayashi and Yukita [138] proposed a Recursive-Rule eXtraction (Re-RX) algorithm which generated more rules than other rule extraction algorithms due to its recursive nature. To overcome this they applied the J48graft algorithm combined it with sampling selection techniques. Re-RX algorithm could classify diabetes considerably better than other rule-based algorithms in previous studies. A classification accuracy of 83.83% was achieved by Re-RX algorithm after 10 runs of 10-fold cross validation. Their improved Re-RX with J48graft algorithm provided a few number of rules than the original Re-RX algorithm. Lekkas et al. [120] proposed an evolving fuzzy classification method which allowed data to be processed online by recursively modifying a fuzzy rule

base on a per-sample basis from data streams instead of processing data offline like a majority of existing fuzzy systems. Offline data processing in fuzzy rule-based systems means that the process of rule induction is in one shot (batch learning), while in online data processing the process of rule induction occurs over indefinite periods which permits the refinement of rules as new data becomes available. The evolving fuzzy method achieved a classification accuracy of 83% on Pima Indian dataset. Sharifi et al. [118] used the Takagi-Sugene fuzzy system to classify diabetes. The system was named ANFIS-GMDH because it was based on adaptive neuro-fuzzy inference system (ANFIS) structure. A classification accuracy of 80% was achieved using this approach.

2.8 Other ML Techniques

Pasrapoor and Bilstrup [51] used an ensemble method based on human emotional processing system that was inspired by human decision making process. This approach achieved a classification accuracy of 77%. Vembandasamy and Karthikeyan [131] proposed the use of fuzzy clustering with outlier detection method to classify diabetes. They initially performed the c-means clustering algorithm to create clusters that were used at a later stage for identifying outliers in the data. The Modified PSO with the Least Square SVM (MPSO-LSSVM) technique was then applied on the data with removed outliers to perform classification. They found that their proposed technique achieved a classification accuracy of 93% which was 9% better than the standard MPSO-LS-SVM algorithm without outlier detection. Vembandasamy and Karthikeyan also noted that the detection of outliers in the data significantly reduced computation time by 38%.

2.9 Summary and Conclusion

A brief summary of classification results from literature which were discussed in this chapter is given in table 2.2 below.

Algorithm Type	Algorithm	Year	Reference	Classification Accuracy
Artificial Neural Networks	K-Means+C4.5+ANN	2012	Priya et al. [15]	98%
	ML-BPN	2015	Durairaj and Kalaiselvi [13]	91%
	GA-ML-BPN	2011	Karegowda et al. [14]	84%
	GRNN	2011	Adeyemo and Akinwonmi [102]	84%
	Baysian Technique	2012	Priya et al. [15]	79%
	DTN	2012	Mehmet et al. [55]	76%
	LVQ	2012	Mehmet et al. [55]	73%
	PNN	2012	Mehmet et al. [55]	72%
	CFN	2012	Mehmet et al. [55]	68%
	FFN	2012	Mehmet et al. [55]	67%
	TDN	2012	Mehmet et al. [55]	67%
	Gini Algorithm	2012	Mehmet et al. [55]	66%
KNN	KNN	2014	Saxena et al. [8]	97%
	GA-KNN-Kmeans	2012	Karegowda et al. [20]	89%
	KNN-K-Means	2014	Vijayan et al. [22]	80%
	K-Means	2014	Vijayan et al. [22]	77%
	KNN	2014	Vijayan et al. [22]	73%
Decision Tree	J48	2015	Habibie et al. [31]	99%
	Improved J48	2014	Kaur and Amit [32]	91%
	ID3+C4.5+CART	2014	Bashir et al. [36]	72%
	C4.5	2012	Ashwinkumar and Anandakumar [37]	68%
	CART	2012	Ashwinkumar and Anandakumar [37]	44%
Support Vector Machines	GDA+LSSVM	2008	K. Polat et al. [43]	79%
	VaR-SVM	2013	Tsyurmasto et al. [44]	93%
	SSVM	2009	Purnami et al. [45]	88%
	SVM+MLP-BPN	2012	Zolfaghari [47]	79%
Population-Based Algorithms	GA	2010	Aslam and Nandi[50]	75%
	GP+CPS	2010	Aslam and Nandi [50]	82.312%
Logistic Regression	LR	2015	Nai-aruna et al. [87]	82.308%
	LR+Boosting	2015	Nai-aruna et al. [87]	78%
	LR+Bagging	2015	Nai-aruna et al. [87]	77%
	LR	2013	Meng et al. [91]	75%
Rule-based Systems	IF-THEN+ACO	2010	Ganj et al. [48]	80%
	Fuzzy-Rule	2010	Aibinu [49]	80%
	ANFIS-GMDH	2008	Sharifi et al. [118]	80%
	Evolving Rule-Base	2010	Lekkas et al. [120]	83%

Table 2.2 Comparison of results in literature

The use of SVMs and ANNs has shown that good results can be obtained by applying them, especially if they are hybridized with other powerful ML techniques. The studies also show that hybridizing different ML techniques improves the classification accuracy in a significant way in many instances. The performance of various ML techniques was greatly influenced by the selection of good training parameters such as the value of k for KNN and the learning rate for algorithms such as Lavenberg-Marquart algorithm for ANNs.

It can be observed from literature that many hybrid ML techniques which obtained improved results were comprised of standard algorithms, i.e., there were no attempts made to first improve the performance of individual algorithms before hybridizing them. This area requires further research to see whether the diabetes diagnosis results could be improved further by hybridizing ML techniques which were individually improved. The performance of a majority of ML techniques proposed in literature for diabetes diagnosis was only assessed by evaluating their classification accuracy instead of also reporting the results of other performance measures such as sensitivity, specificity, PPV, NPV, and AUC of ROC to mention a few. Classification accuracy on its own does not provide enough evidence to demonstrate the effectiveness of a classifier [139]. There are instances in a dataset where there is class imbalance or a classifier is only able to learn to effectively classify negative instances more than positive instances, or vice versa. For example, a classifier with a classification accuracy of 85% which can correctly classify 80% of negative cases and 5% positive instances in a dataset with 80 negative cases and 20 positive cases is not useful in implementing a medical diagnosis system where the identification of positive cases is more important. This shows that assessing the performance of a classifier by only considering its classification accuracy may lead to incorrect conclusions.

It was also observed in literature that only a handful of studies rigorously applied data preprocessing techniques to improve the quality and consistency of their datasets before applying ML for diagnosis. An objective of this study is to address these gaps which were observed in literature.

Chapter 3

Methods

This chapter presents the methods and materials employed to solve the diabetes diagnosis problem. The data preprocessing and Machine Learning techniques used in the study are detailed. The algorithm performance evaluation measures used are also described in this chapter.

3.1 Datasets

This study made use of the widely-used Pima Indian diabetes dataset [21]. The Pima Indian, a native American population in Phoenix Arizona, USA, has been continually studied and examined since 1965 due to the high incidence of diabetes [52]. Pima Indian females participated in standardized diabetes examinations. Three additional datasets were derived from the Pima Indian dataset and used in this study.

In this dataset, diabetes was diagnosed according to the World Health Organization criteria [12]. A patient was considered diabetic if the 2 hour post-load plasma glucose was at least 200 mg/dl $(11.1 mmol/l)$ at any survey examination, or if the Pima Indian Health Service Hospital serving the community found a glucose concentration of at least 200 mg/dl during the course of routine medical care [22]. The Pima Indian dataset has 768 observations, each with nine attributes. Five-

hundred of these patients were non diabetic, and the rest were diabetic. Each patient has only one record in the dataset. The attributes in this dataset are given below. Basic statistical properties of the dataset are given in Table 3.1.

1. Number of times pregnant (PREG)
2. Plasma Glucose Concentration at 2 Hours in an Oral Glucose Tolerance Test (GLUC)
3. Diastolic Blood Pressure $(mmHg)$ (PRESS)
4. Triceps Skin Fold Thickness (mm) (SKIN)
5. 2-Hour Serum Insulin (Uh/ml) (INSU)
6. Body Mass Index (Weight in kg / (Height in cm)) (BMI)
7. Diabetes Pedigree Function (PDF)
8. Age in years (AGE)
9. Diabetes Class Variable (0 or 1) ▷ *where 0 = Not diabetic and 1 = Diabetic*

Feature	**Minimum**	**Maximum**	**Mean**	**Variance**	**Standard Deviation**
Number of times pregnant	0.00	17.00	3.85	11.34	3.37
Plasma Glucose Concentration at 2 Hours in an Oral Glucose Tolerance Test	0.00	199.00	120.89	1020.92	31.95
Diastolic Blood Pressure $(mmHg)$	0.00	122	69.11	374.16	19.34
Triceps Skin Fold Thickness (mm)	0.00	99.00	20.54	254.14	15.94
2-Hour Serum Insulin (Uh/ml)	0.00	846	79.8	13263.89	115.17
Body Mass Index (Weight in kg / (Height in cm))	0.00	67.10	31.99	62.08	7.88
Diabetes Pedigree Function	0.08	2.42	0.47	0.11	0.33
Age (years)	21.00	81.00	33.22	138.12	11.75

Table 3.1 Statistical properties of the dataset

3.1.1 Data Preprocessing

All attribute values in the Pima Indian dataset were re-scaled (normalized) to avoid the disadvantages with using data that has varying value scales for classification. For example, age and blood pressure attributes are on different numerical scales. Min-Max, Z-score, and Decimal Scaling are popular normalization techniques. In this study, the Min-Max Normalization technique was used to re-scale all attribute values to be in the range [0,1]. This technique was selected because of its simplicity and ability to preserve the relationships among the original data values. Min-Max Normalization provides a linear transformation on the original range of data by computing a new value for each attribute of an instance.

Feature Selection Techniques

Feature selection is a technique that is used to select a subset of original features from a dataset using a certain criteria with the aim of reducing irrelevant, redundant, or noisy features [147]. Feature selection helps to speed up the learning process of Machine Learning techniques and it improves their learning accuracy [147]. The Analytic Hierarchy Process (AHP) and learner-based feature selection techniques were explored and used for feature selection. AHP is a method, introduced by Thomas Saaty [59], to deal with complex decision making process. AHP helps the decision maker to define priorities for variables and to perform comparisons using a scale of absolute judgements to determine the significance of one variable to another with respect to a given attribute. AHP uses a process of pair-wise comparison to determine the relative importance of criteria or alternatives in decision making [67]. The pairwise comparison is represented as a matrix, m, to compute relative scores. The following scores are used to create the pairwise comparison matrix for different criteria:

Learner-based feature selection techniques employ a generic but powerful learning algorithm on

m_{jk}	**Interpretation**
1	j and k are equally important
3	j is slightly more important than k
5	j is more important than k
7	j is strongly more important than k
9	j is absolutely more important than k
2,4,6,8	Used when when there is no good word to describe the comparison

Table 3.2 Relative Scores [148]

the dataset. The performance of the learning algorithm is evaluated with different subsets of attributes which are selected from the dataset. The subset of attributes which make the algorithm achieve the best results are then selected. The algorithm used for feature selection does not have to be an algorithm intended to be used to model the data. The Random Forest algorithm was selected as a classifier for selecting features in this study.

3.1.2 Derived Datasets

Attributes with missing values are known to have a negative impact on the accuracy of the classification model by causing noise and bias. Attributes such as diastolic blood pressure and insulin level cannot have zero values. Breault [54] conducted a thorough analysis of missing attribute values in the Pima Indian dataset. He discovered that many published studies based on the Pima Indian diabetes dataset had overlooked missing values and had simply used the data values as recorded without preprocessing them.

Breault [54] emphasized that overlooking dataset incompleteness is a serious concern especially since some of the missing data fractions are very high. For example, triceps skin fold thickness has approximately 30% of entries missing, while serum insulin concentration has approximately

49% of entries missing. The preprocessing and feature selection techniques described above were used to decide how to handle missing values, identify relevant features, and to address varying attribute value scales in the dataset. As a result of applying these methods on the Pima Indian dataset, three additional datasets were formed. These datasets were used to evaluate the various ML techniques and to determine the efficacy of different dataset preprocessing techniques. These additional datasets were:

1. Dataset **B-[Excl Missing]:** all instances with missing values were removed.

2. Dataset **C-[Replaced by Mean]:** missing attribute values were replaced by the mean value of that attribute (except attributes: number of times pregnant and class variable).

3. Dataset **D-[Extracted Features]:** insignificant features were excluded using the results of AHP and the learner-based feature selection technique (Random Forest algorithm).

The original un-preprocessed dataset is referred to as **A-[Unprocessed]**. Table 3.3 provides an overview of basic properties of these derived datasets and the original dataset.

Measure	A-[Unprocessed]	B-[Excl Missing]	C-[Replaced by Mean]	D-[Extracted Features]
Number of Instances	768	359	768	359
Number of Positive Instances	268	128	268	128
Number of Negative Instances	500	231	500	231
Number of Instances With Missing Values	409	0	0	0

Table 3.3 Properties of diabetes dataset A, B, C and D.

3.2 Machine Learning Techniques

Artificial Neural Networks, Statistical Regression, and Instance-based ML techniques are widely used in solving the diabetes classification problem. This study aims to enhance these techniques

using novel approaches to extend their capabilities and to perform a comparative analysis of their performance. The performance of the enhanced ML techniques is also compared with previous studies in literature which employed similar techniques to diagnose type II diabetes. Algorithm performance measures such as classification accuracy, sensitivity, and specificity were used to compare the performance of these techniques. The following enhanced ML techniques for Type II diabetes classification are proposed in this study:

1. **Instance-based Techniques**

   (a) *The K-fold Cross-Validated Enhanced K-Means clustering algorithm (CVE-K-Means)*
   This technique is proposed to improve the standard K-means clustering algorithm by finding an optimal value of k within a predefined range of values. To obtain clusters of good quality, this technique selects initial cluster centers by computing the distance between each training instance and the origin. The distance is then used to initially assign training instances to appropriate clusters based on their proximity to the origin. The CVE-K-Means algorithm also excludes outliers in the dataset during training.

   (b) *A Hybrid of CVE-K-Means and the KNN algorithm (CKNN)*
   The cluster centers (centroids) produced by the CVE-K-Means algorithm from training instances are used by the KNN algorithm as the training set to improve its computation time and classification accuracy.

2. **Artificial Neural Networks**

   (a) *A hybrid of CVE-K-Means and the Generalized Regression Neural Network (KGRNN)*
   The KGRNN uses the CVE-K-Means algorithm to produce centroids which represent training instances to reduce the computation time of the network.

   (b) *An ensemble of CVE-K-Means, Boosting, and the Generalized Regression Neural Network (B-KGRNN)*

This technique employs the Boosting ensemble to further enhance the classification accuracy of the KGRNN technique by combining poor performing KGRNN models to produce a single powerful KGRNN classifier.

(c) *An efficient Multi-Layer Artificial Neural Network (MLP-BPX)*

The MLP-BPX network uses a Xavier weight initialization technique to allow it to quickly converge, thus improving the computation cost and classification accuracy. The Back-Propagation algorithm was used to train the network.

3. Statistical Techniques

(a) *An improved Logistic Regression model with n-restarts (LR-n)*

The LR-n model restarts $n-times$ during training to produce k models which are created using different data samples from the same dataset (with replacement). The final classification accuracy is obtained by computing the average accuracy of k models during testing.

All the ML techniques proposed in this study for diabetes classification were implemented in Java programming language on a machine with Intel i5 vPro processor, 2.30GHz speed, 8 GB RAM, and using JDK 1.6 software (Java Development Kit). The following subsections describe these techniques in detail.

3.2.1 Instance-based Techniques

Instance-based techniques in type II diabetes diagnosis have shown promising results in literature and are widely-used [22, 27, 8]. These techniques have also proved to be useful in efficiently sampling and grouping instances in the dataset before applying other advanced ML techniques. This study proposes novel ways to improve the computation time and classification accuracy of K-Means clustering and the KNN algorithm. In this section, a brief overview of the standard

K-Means clustering algorithm and the KNN algorithm will be provided. Extensions to these techniques are then described.

K-Means Clustering Algorithm

K-Means clustering partitions n objects into k clusters in which each object belongs to the cluster with the nearest mean [145]. The objective of K-Means clustering is to minimize the total intra-cluster variance, or, the squared error function [145]:

$$J(v) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} (||x_i - v_j||)^2 \tag{3.1}$$

where: $||x_i - v_j||$ is the Euclidean distance between the datapoint x_i and the cluster centre v_j, c_i is the number of data points in the i-th cluster, and c is the number of cluster centers. The Euclidean distance between two multi-dimensional data points X = $(x_1, x_2, x_3 ... x_m)$ and Y = $(y_1, y_2, y_3 ... y_m)$ is computed as follows:

$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... (x_m - y_m)^2} \tag{3.2}$$

The pseudocode for the K-Means clustering algorithm is given in Algorithm 1.

Algorithm 1: Pseudocode for the K-Means clustering algorithm, adapted from [145]

Require: A set D of n data points where D = $d_1, d_2, d_3, ..., d_i, .., d_n$

Produce: A set of k clusters with k centroids

$centroidList \leftarrow initializeCentroids(D)$

$clusters \leftarrow newClusters(D$,centroidList);

repeat

  foreach *data point in the clusters* **do**

    $closestCentroid \leftarrow findClosestCentroid(computeDistances($datapoint,centroidList$))$;

    $clusters \leftarrow assignDataPoint($datapoint,closestCentroid$)$;

  $centroidList \leftarrow computeNewCentroids($clusters$)$;

until *centroids are not changing or no data point is moving from one cluster to another*;

Return: *clusters*;

Enhanced K-Means Clustering Algorithm (CVE-K-Means)

The selection of initial cluster centers in K-Means clustering greatly influences the inter-cluster and intra-cluster distances and cohesion [70]. Researchers have extensively investigated the impact of intelligently selecting the initial centroids using different approaches to improve the efficiency and accuracy of K-Means clustering algorithm [69]. In this study, the efficiency and accuracy of K-Means clustering was enhanced by extending a method proposed by Madhu Yedla et al. [69] to intelligently determine initial centroids. They computed a distance between each instance with n attributes and the origin in the n-dimensional space using the Euclidean distance method. The data points were then sorted with respect to their distances from the origin. The sorted list was then split into k samples of the same size. To determine initial centroids, an instance in the middle of the

sorted instance list in each sample was selected. They addressed negative attribute values in their datasets by transforming negative attribute values into positive values. The authors concluded that the selection of initial centroids using this approach led to better results with lower time complexity compared to other clustering techniques.

This study proposes an enhanced K-Means clustering algorithm for efficiently grouping similar instances into the same cluster. The proposed algorithm is called CVE-K-Means algorithm because it is an extended K-Means clustering algorithm which uses k-fold cross validation. The proposed CVE-K-Means algorithm uses the initialization technique proposed by Madhu Yedla et al. [69]. However, it differs from the standard K-Means clustering algorithm and the algorithm proposed by Madhu Yedla et al. in that it searches for an optimal value of k (i.e. the number of clusters) within a pre-defined range for a given dataset using k-fold cross validation. K-fold cross validation is implemented by partitioning the dataset into k equal samples and running the classification algorithm k times, ensuring that all samples are used as a test set exactly once and are used at least once as part of the training set. The new CVE-K-Means algorithm also excludes outliers in the dataset using the relative distance of each datapoint from the object that has attribute values similar to the origin in an n-dimensional space. The pseudocode for the CVE-K-Means clustering algorithm is given in Algorithm 2.

The CVE-K-Means algorithm enhances the standard K-Means clustering algorithm in the following ways:

1. It introduces the use of the distance measure between data points and the origin to select initial clusters.
2. It dynamically searches for the best k for a given dataset, thus minimizing the reliance on a single value of k selected by a user.
3. It excludes outliers in the training and test datasets to improve the effectiveness of the algorithm.

4. It uses the *k*-fold cross validation methodology to ensure that every data point is used exactly once as part of the testing set and at least once as part of the training set.

Algorithm 2: Pseudocode for CVE-K-Means Algorithm

```
1  Require: A set D of n data points where D = d1, d2, d3, ..., di, .., dn
2  Input: A range for number of clusters minNum,maxNum
3  Produce: A set of best clusters and the classification accuracy after k-fold Cross Validation
4  begin
5      bestK ← −1;                                                          ▷ initialize best k
6      bestClassificationAccuracy ← 0;
7      currentK ←minNum;
8      D ← removeOutliers(D);
9      partitions ← getEqualRandomSamples(D,m);   ▷ Partition dataset D into m equal samples. m = number of samples
10     repeat
11                                                ▷ Perform k-fold cross validation on m samples in 12-34
12         repeat
13             testSet ← partitions.get(0); partitions.remove(0);
14             trainingSet ← clone(partitions);
15                                  ▷ Intelligently select initial cluster centers and cluster training data in 16-29
16             foreach data point in trainingSet do
17                 Calculate its distance from origin;
18             sortedTrainingSet ← sortByDistance(trainingSet);
19             clusters ← getKClusters(sortedTrainingSet,currentK);
20             foreach cluster in clusters do
21                 centroidList.add(cluster.get(midpoint));
22                 cluster.remove(midpoint);
23             repeat
24                 foreach data point in clusters do
25                     distanceList ← computeDistance(datapoint,centroidList);
26                     closestCentroid ← findClosestCentroid(distanceList,centroidList);
27                     clusters ← assignDataPoint(datapoint,closestCentroid);
28                 computeNewCentroids(centroidList, clusters);
29             until All centroids are not changing or no data point is moving from one cluster to another;
30             testAccuracy ← classifyUnseenData(testSet,centroidList);
31             testAccuracyList.add(testAccuracy);
32             partitions.add(testSet,partitions.getSize()-1);   ▷ Return current test set to initial samples
33             testSet.clear();
34         until Each sample in m samples is used exactly once as the test set and at least once as the training set;
35         accuracyForCurrentK ← average(testAccuracyList);                  ▷ Avg of results after CV
36         if accuracyForCurrentK > bestClassificationAccuracy then
37             bestClassificationAccuracy ← accuracyForCurrentK;
38             bestK ← currentK;
39         testAccuracyList.clear();
40     until Until all k-values in the range minNum:maxNum are considered;
41     bestCentroidList ← getBestCentroidList(bestK, partitions);   ▷ Get list of centroids using best k found during CV
42     saveBestCentroids(bestCentroidList);          ▷ Or pass them as input to another ML technique like GRNN
43     return bestClassificationAccuracy;
```

KNN Algorithm

The KNN algorithm performs classification by considering a dominant class among k nearest training instances to the testing instance rather than constructing a classification model from training data. The pseudocode of the KNN algorithm is given in Algorithm 3. The objective of the KNN algorithm is to minimize the Sum of Squared Errors (SSE) function [22]:

$$argmin_C = \sum_{i=1}^{k} \sum p \in C_i |p - m_i|^2 \tag{3.3}$$

where: p is the object that belongs to cluster C_i, m_i is the centroid for cluster C_i, k is number of clusters, and $|p - m_i|^2$ is the squared distance between the cluster centroid and an instance.

Algorithm 3: Pseudocode of KNN Algorithm, adapted from [22]

Require: A set D of m data points where D = $d_1, d_2, d_3, ..., d_i, .., d_n$ is a set of test instances
Require: A set M of n instances where M = = $m_1, m_2, m_3, ..., m_i, .., m_n$ is a set of training instances
Input k as the number of desired neighbours to be used for classification
Produce: A percentage of correctly classified examples of test set using training set
begin
    $numOfCorrectClassifications \leftarrow 0$;
    foreach *test instance* d_i *in* D **do**
        distanceList $\leftarrow computeDistance(d_i, M)$;
        neighbours $\leftarrow getNeighbours(distanceList, M, k)$;
        classification $\leftarrow classify(neighbours, d_i)$; ▷ *Using a voting scheme with tie breaking rules*
        if $d_i.getClass()$ == *classification* **then**
            $numOfCorrectClassifications$++;
    return 100 x ($numOfCorrectClassifications$/D.getSize()); ▷ *Classification accuracy*

An Improved KNN Algorithm (CKNN)

Although KNN is widely used for diverse classification problems, it has major drawbacks which hinder its ability to classify instances with low computational cost and optimum accuracy. This study proposes a Centroids-based KNN (CKNN) which uses the best cluster centres obtained from the CVE-K-Means algorithm for classification. This new approach tries to minimize the computational cost of KNN while also attempting to improve the classification accuracy. In the CKNN technique, the best cluster centers produced by CVE-K-Means algorithm are used as a training set for KNN to classify instances in the test set. Therefore, instead of computing the distances between the test instance x and all instances in the training set to determine neighbours of x, the proposed algorithm computes the distance between x and the selected cluster centres. The algorithm then selects k cluster centres which are the nearest to the test instance in order to assign the test instance to a known class by majority voting. The CKNN algorithm addresses the following weaknesses of KNN which were identified by [22]:

1. The KNN algorithm has a high computational cost because of the necessity to compute the distance between each test instance and all training examples. The proposed CKNN technique addresses this by using a subset of the training set which is produced by employing an efficient K-Means clustering algorithm on the entire training set.

2. The KNN algorithm needs a large amount of data for training which leads to the need for large memory space to store the training set. Since the CKNN algorithm uses the subset of the training dataset, it ensures that less memory is needed to store the training set for classification. Furthermore, the use of cluster centers as training instances allows the CKNN algorithm to perform classification with less training data compared to the KNN algorithm.

Algorithm 4 below gives a pseudocode for the CKNN Algorithm.

Algorithm 4: Pseudocode for CKNN Algorithm

Require: A set D of m instances, where D = d_1, $d_2, d_3, ..., d_i, .., d_m$ is a set of test instances
Require: A set M of n instances, where M = m_1, $m_2, m_3, ..., m_i, .., m_n$ is a set of training instances
Input k as the number of desired neighbors to be used for classification
Input x as the desired number of centroids
Produce: A percentage of correctly classified instances of test set using centroids produced by CVE-K-Means algorithm
begin
    $numOfCorrectClassifications \leftarrow 0$;
    centroidList ←CVE-K-Means(M,x); ▷ *Invoke CVE-K-Means algorithm*
    $trainingSet \leftarrow centroidList$; ▷ *Size of trainingSet < size of M, thus improving computation cost*
    foreach *test instance* d_i *in D* **do**
        $distanceList \leftarrow computeDistance(d_i, trainingSet)$;
        $neighbours \leftarrow getNeighbours(distanceList, trainingSet, k)$;
        $classification \leftarrow classify(neighbours, d_i)$; ▷ *Using a voting scheme with tie breaking rules*
        if $d_i.getClass() == classification$ **then**
            $numOfCorrectClassifications++$;
    return $100\ x\ (numOfCorrectClassifications/D.getSize())$; ▷ *Classification accuracy*

3.2.2 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) have been extensively applied to solve the diabetes classification problem [13, 14, 79, 137]. ANNs are widely used because of their highly adaptive learning, excellent self-organization property, and their high level of fault tolerance [76]. ANNs in literature have shown to perform better when they are hybridized with other ML techniques. The diabetes

classification results using ANNs in previous studies showed that there is still a need to further enhance the performance of ANNs to obtain more accurate results. This study proposes the use of an intelligent weight initialization technique, the CVE-K-Means algorithm, and a boosting ensemble to enhance the performance of a multi-layer ANN and a Generalized Regression Neural Network (GRNN). This section provides an overview of these ANNs and how they were enhanced to improve their performance.

Generalized Regression Neural Network

A GRNN is used to estimate a joint probability density function (pdf) of an independent variable x and the dependant variable y, given only a training set [79]. The GRNN is derived from the Radial Basis Function (RBF) network that is based on a standard statistical technique called kernel regression. Therefore, a GRNN can be viewed as a normalized RBF network which has each unit (neuron) centred at every training instance. A GRNN uses the Gaussian activation function in its processing elements (neurons). Suppose that $f(x,y)$ represents a known joint continuous pdf of a vector random variable x, and a scalar random variable y [78]. Let X be a particular measured value of the random variable x. The conditional mean of y given x (also referred to as regression of y on x) is given by [78] as:

$$E[y|x] = \frac{\int_{-\infty}^{\infty} y.f(x,y)dy}{\int_{-\infty}^{\infty} f(x,y)dy} \tag{3.4}$$

where: y = output of the estimator, x = the estimator input vector, E$[y|x]$ = the expected value of output given the input vector x, and $f(x,y)$ = the pdf of x and y. GRNN is therefore based on the function in (3.4), and this function can be optimally estimated in the following way:

$$y_i = \frac{\sum_{i=1}^{n} g_i.w_{ij}}{\sum_{i=1}^{n} g_i} \tag{3.5}$$

where: x is the input test vector, v_i is the training vector i at neuron i, w_{ij} is the target output corresponding to input training vector v_i, $g_i = e^{-\frac{D_i^2}{2.\sigma^2}}$ is the output of a hidden layer neuron, $D_i^2 = (x-v_i)^T(x-v_i)$ is the squared distance between the input vector x and the training vector v, and σ = a constant controlling the size of the receptive region. The pdf used in a GRNN follows the normal distribution; therefore each training example x_j is used as the mean of a normal distribution. In a GRNN, the distance D_i between the training sample and the test instance x is used to measure how well each training example can represent the position of x.

The smoothness parameter, sigma, can be determined through trial and error or using methods proposed in [78]. When the smoothness parameter is too large, it allows training samples that are far from the point of classification to have a high influence on the final classification [80]. Leading to poor classification accuracy. Furthermore, if the smoothing parameter is chosen to be very small, then it only allows training examples that are very close to the point of classification to have a high influence towards classification. The pseudocode for training and testing the GRNN network is given in Algorithm 5 below.

Algorithm 5: Pseudocode for the GRNN

Require: A set D of m instances, where $D = d_1, d_2, d_3, ..., d_i, .., d_m$ is a set of test examples
Require: A dataset T of n instances, where $T = T_1, T_2, T_3, ..., T_i, .., T_n$ are training instances, and $n > m$
Produce: A classification accuracy of the network given all instances from D as input
begin
    Make each instance T_i in dataset T a centre point for each neuron N_i in the hidden layer
    numClassified ← 0 ▷ *Initialise counter of correctly classified test instances to zero*
    for *each instance* d_i *in* D **do**
        input layer of the network receives d_i *attributes as input;*
        input layer passes inputs to all neurons in the hidden layer;
        ▷ *Hidden layer receives the input from input layer and computes Gaussian function in 11-14*
        for *each neuron in the hidden layer* **do**
            $D_i \leftarrow$ *distance(input,*T_i*);*
            Gaussian activation function $g_i = e^{-\frac{D_i^2}{2.\sigma^2}}$;
            Transfer the result of activation function as input to neurons in the summation layer;
        ▷ *Summation Layer computations in 16-17*
        $numerator \leftarrow \sum_{i=1}^{n}(weight_i \text{ x } GaussianOutput_i)$;
        $denominator \leftarrow \sum_{i=1}^{n}(GaussianOutput_i)$;
        Output ← *numerator/denominator;* ▷ *Output Layer result*
        ▷ *Classification process in 20-25*
        if *result* < 0.5 **then**
            classification = 0;
        else
            classification = 1;
        if *classification* == d_i*.getLabel()* **then**
            numClassified++;
    overallAccuracy = numClassified / size of D;
    return *100* x *overallAccuracy;*

Hybrid of CVE-K-Means and GRNN (KGRNN)

A hybrid of CVE-K-Means and GRNN is proposed in this study to classify type II diabetes. The CVE-K-Means algorithm is used to produce k centroids that are used as centre points for neurons in the hidden layer of the GRNN. This approach is proposed to try to minimize the computation effort that is needed to compute a Gaussian function for each instance in the training set and the test instance x. The centroids usually provide a good representation of instances that belong to the same cluster as the centroid. This hybrid technique is called KGRNN since it is based on CVE-K-Means and GRNN. The pseudocode of KGRNN is given in Algorithm 6. Figure 3.1 shows the structure of the proposed KGRNN network. This network has 4 layers. The first layer has 9 neurons representing the 9 attributes in the dataset (including the class label T). The hidden layer has 49 neurons representing 49 RBF activation functions with 49 centroids. The summation layer has 2 neurons which compute the weighted and unweighted sum of inputs from the hidden layer. Finally, the output neuron computes the final classification value.

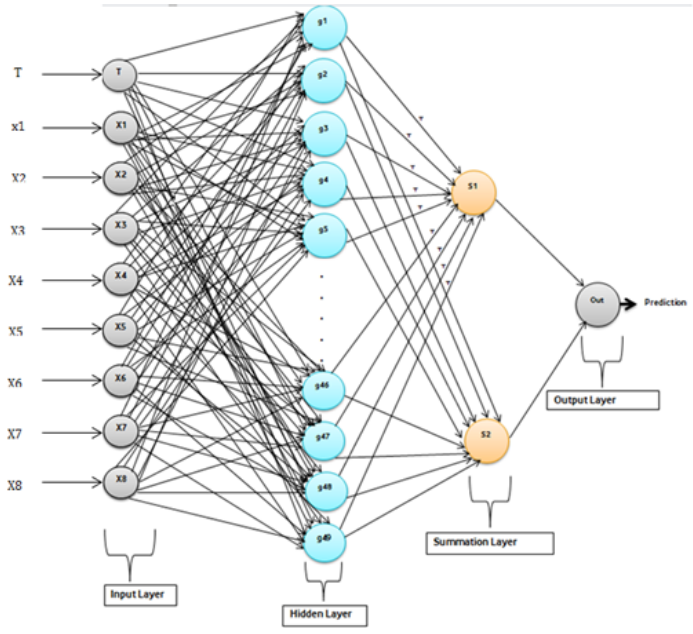

Figure 3.1 KGRNN Architecture

Algorithm 6: Pseudocode for KGRNN

Require: A set D of n instances where $D = d_1, d_2, d_3, ..., d_i, .., d_n$ is a set of test examples
Require: A set C of k-centroids (with $k < n$) produced by CVE-K-Means using training set
Produce: A classification accuracy of the network given all instances from D as input
begin
    Make each centroid C_i in the dataset C be the centre point for each neuron N_i in the hidden layer of GRNN
        ▷ *number of neurons in the hidden layer is = number of centroids*
    numClassified ← *0* ▷ *Initialise counter of correctly classified test instances to zero*
    for *each instance* d_i *in D* **do**
        input layer of the network receives d_i *attributes as input;*
        input layer passes inputs to all neurons in the hidden layer;
            ▷ *Hidden layer receives the input from input layer and computes Gaussian function in 12-14*
        for *each neuron in the hidden layer* **do**
            $D_i \leftarrow$ *distance(input,*C_i*);*
            Gaussian activation function $g_i = e^{-\frac{D_i^2}{2.\sigma^2}}$;
            Transfer the result of activation function as input to neurons in the summation layer;
        ▷ *Summation Layer computations in 17-18*
        $numerator \leftarrow \sum_{i=1}^{n}(weight_i \text{ x } GaussianOutput_i)$;
        $denominator \leftarrow \sum_{i=1}^{n}(GaussianOutput_i)$;
        Output ← *numerator/denominator;* ▷ *Output Layer result*
        ▷ *Classification process in 21-26*
        if *result* < 0.5 **then**
            classification = 0;
        else
            classification = 1;
        if *classification* == d_i*.getLabel()* **then**
            numClassified++;
    overallAccuracy = numClassified / size of D;
    return *100* x *overallAccuracy;*

Ensemble of CVE-K-Means, Boosting, and GRNN (B-KGRNN)

This study proposes an ensemble of GRNN, the CVE-K-Means algorithm, and the Adaboost algorithm to further improve the performance of the GRNN network. Boosting is a machine learning technique that is used to reduce bias and variance in the data during supervised learning, thus improving the accuracy of the model. The primary purpose of boosting is to convert a set of weak learners into strong learners. A weak hypothesis or a weak learner performs slightly better than random chance [71]. There are many boosting algorithms, e.g., AdaBoost (Adaptive Boosting), Gradient Tree Boosting, and XGBoost. These algorithms implement boosting by iteratively learning weak classifiers with respect to a distribution and then combining these weak classifiers into a strong classifier. The contribution (weight) of each weak classifier to the final classification is based on its classification accuracy.

Adaboost was chosen because it was primarily designed to solve binary classification problems such as diabetes diagnosis. Adaboost carries out boosting by assigning weights to training instances. Instances that are hard to classify are assigned more weight and those that have been already classified well by other classifiers are assigned less weight. This allows classifiers to focus their training on more difficult patterns. The output of the final classifier is limited to either 1 or -1. Formally, the final classifier in Adaboost is defined as follows:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{3.6}$$

where: T = the number of weak classifiers, $h_t(x)$ = the output of weak classifier t (output limited to -1 or +1), and α_t = the contribution weight of the classifier t to final classification (weight is based on classification accuracy of the classifier). The final output is a linear combination of all the weak T classifiers. The classification decision is made by observing the sign of the sum. The weak classifiers are trained one at a time. The probability (weight) of each training instance appearing in the training set for the next classifier is computed after each classifier is trained. The first classifier

is always trained with equal weights assigned to all training instances. The value of α_t for each classifier is computed after training. α_t is based on the classification error rate e_t of the classifier. The formula for computing α_t is given as follows [86]:

$$\alpha_t = \frac{1}{2} ln \left(\frac{1-e_t}{e_t} \right) \tag{3.7}$$

The weights of training instances are updated after computing alpha and the error rate using the equation [86]:

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t} \tag{3.8}$$

where: $D_t(i)$ = the vector of training instance weights, $exp(-\alpha_t y_i h_t(x_i))$ = formula used to scale the weight of the training instance up or down based on the instance class which is computed by the classifier $h_t(x_i)$, and Z_t = sum of weights that is used to normalize a set of weights to make them a probability distribution. The Adaboost algorithm is outlined in Algorithm 7. Since the

Algorithm 7: Adaboost Algorithm, adapted from [86]

begin

  Given: $(x_1, y_1), ...(x_m, y_m)$, where $x_i \in X, y_i \in Y = \{-1, +1\}$

  Initialize: $D_1(i) = 1/m$ for $i = 1, 2, ..., m$;

  for `t =1; t<=T; t++` **do**

    Train weak leaner using distribution D_t;

    Get weak hypothesis h_t : X $\rightarrow$ {-1,+1} with error $e_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$;

    Choose $\alpha_t \leftarrow \frac{1}{2} ln \left(\frac{1-e_t}{e_t} \right)$;

    Update, **for** `i =1; i<=m; i++` **do**

      $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$; ▷ Z_t = *normalization factor to make* D_{t+1} *a distribution*

  H(x) = sign $\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$; ▷ *Result of final classifier*

Adaboost algorithm gives an output of 1 or -1, the instances with class label 0 on the dataset used

for training the Adaboost algorithm had their class label replaced with -1. The psuedocode for the boosted KGRNN network is given in Algorithm 8 below.

Algorithm 8: Pseudocode for B-KGRNN Network

Require: A set D of m instances where D = $d_1, d_2, d_3, ..., d_i, .., d_m$ as test set.
Require: A set T of n instances with n greater than m as a training set.
Input: Minimum acceptable error MinError
Input: The value of smoothing factor sigma
Input k as the desired number of centroids to be used as training set
Produce: A classification accuracy of B-KGRNN given all instances from D as input
begin
    repeat
        if *iteration == 1* **then**
            foreach *instance in training set T* **do**
                instance.setWeight(1/n); ▷ *All training instances have equal probability of being selected from the beginning*
        trainingSubSet ← $getInstancesWithHigherWeight(w)$;
        centroidList ←CVE-K-Means(trainingSubSet, k); ▷ *Invoke CVE-K-Means to get k centroids*
        weakClassifier ← $KGRNN(centroidList, trainingSubSet, Sigma)$; ▷ *KGRNN returns classifier with training error less than MinError*
        classifierList.add(weakClassifier);
        MinError ← $weakClassifier.getClassificationError()$;
        weakClassifierWeight ← $getClassifierAlpha(weakClassifier)$;
        classifierWeightList.add(weakClassifierWeight);
        listOfcentroidList.add(centroidList);
        foreach *Instance in training set T* **do**
            Update instance weight using formula (7.8); ▷ *For likelihood of instance to be in next batch*
    until *required number of weak classifiers is produced or no improvement in training error*;
    ▷ *Weak classifier is a KGRNN instance with its set of centroids and* α
    accuracy ← $finalClassifier(classifierList, classifierWeightList, Sigma, TestSetD)$;
    return *accuracy;*

Improved Multi-Layer Back-Propagation Neural Network (MLP-BPX)

A Multi-Layer Neural Network (MLP) is a feed-forward neural network with one or more layers between the input layer and output layer. This type of network is usually trained using the Back-Propagation algorithm and it is one of the most extensively used type of ANNs to solve the diabetes classification problem [92]. The MLP network was proposed in previous studies to overcome the limitations of single-layer neural networks which were only capable of solving linearly separable classification problems [92]. The MLP network approximates the non-linear relationship between the input to the network and the output of the network by adjusting weights that connect neurons between different network layers. MLPs have exactly one input layer and output layer and can have an unlimited number of hidden layers. The MLP proposed in this study is called the MLP-BPX network because it uses a Xavier [149] weight initialization technique and the Back-Propagation algorithm for learning. The learning of the MLP-BPX network using the Back-Propagation algorithm involves four main stages:

1. **Weight Initialization**

   The weights of MLP-BPX network are intelligently initialized using a Xavier weight initialization technique which selects values from a Gaussian distribution with zero mean and variance of *1/((number of input nodes + number of output nodes)/2).*

2. **Feed-Forward:**

   In this stage, each node (including bias nodes) computes a weighted sum of inputs and uses an activation function to produce an output which is transferred to the nodes in the next layer as input. This process continues until the last hidden layer sends its output to the output layer. The formula for the weighted sum x_j is given as follows:

$$x_j = \sum_{k \in K_j} w_{kj} y_k. \tag{3.9}$$

where: K_j is the set of nodes from the k-th layer which feeds node j, w_{kj} is a set of weights connecting nodes in the k-th layer with node j, and y_k is a set of outputs of the activation functions in the k-th layer which serve as input to the j-th layer nodes. The activation function is given as:

$$f(x_j) = y_j. \tag{3.10}$$

This study uses the sigmoid function to train the MLP-BPX network. The sigmoid function is defined as follows:

$$f(x_j) = \frac{1}{1+e^{-x_j}}. \tag{3.11}$$

The sigmoid function is widely used as an activation function for ANNs because its derivative can be easily computed using the quotient rule or the product rule. The derivative of the sigmoid function is computed as follows:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}x_j} f(x_j) &= \frac{\mathrm{d}}{\mathrm{d}x_j}\left[\tfrac{1}{1+e^{-x_j}}\right] \\
&= \frac{\mathrm{d}}{\mathrm{d}x_j}\left(\ 1+e^{x_j}\ \right)^{-1} \\
&= -(1+e^{-x_j})^{-2}(-e^{-x_j}) \\
&= \frac{e^{-x_j}}{(1+e^{-x_j})^2} \\
&= \frac{1}{(1+e^{-x_j})}\cdot\frac{e^{-x_j}}{(1+e^{-x_j})} \\
&= \frac{1}{(1+e^{-x_j})}\cdot\frac{(1+e^{-x_j})-1}{(1+e^{-x_j})} \\
&= \frac{1}{(1+e^{-x_j})}\cdot\left(\ 1-\tfrac{1}{(1+e^{-x_j})}\ \right) \\
&= f(x_j).(1-f(x_j))
\end{aligned} \tag{3.12}$$

3. **Output Error Computation:**

The output nodes multiply the input by the weights connecting them to the final hidden layer to compute the final result. The result of the activation function $f(x_j)$ of each output

neuron is compared with the expected target result t_y to produce the classification error (E) = $(t_y - y_i)$. Since all nodes in the hidden layer contributed to the overall classification error, the error of the output layer needs to be propagated backwards to each node in the hidden layer. The overall error of the network is given as follows:

$$E = \frac{1}{2}\sum_{j=1}^{J}(t_j - y_j)^2. \tag{3.13}$$

Formula 3.13 is only applicable to the output nodes in the output layer. The training error for nodes in the hidden layer is computed differently.

4. **Output Error Back-Propagation and Connection Weights Update:**

The value of each connection weight is updated using the computed errors. This is done after the errors for all nodes are computed. The weight update process and error back-propagation is repeated until the network converges to a state that allows a certain number of training examples to be encoded by the network. The Stochastic Gradient Descent (SGD) [150] technique is usually used to search for the minimum value of the error function. The backpropagation algorithm iteratively changes weights by their proportion of influence to the overall error. The change of weight connecting a node in layer k to a node in layer j is given as follows [150]:

$$\triangle w_{kj} = -\alpha\frac{\partial E}{\partial w_{kj}}. \tag{3.14}$$

where: α is the predefined learning rate that defines the step size. The partial derivative of (3.14) is expanded using a chain rule to get:

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_j}\frac{\partial x_j}{\partial w_{kj}} \tag{3.15}$$

where x_j is the weighted sum inputs into node j. Therefore:

$$\frac{\partial x_j}{\partial w_{kj}} = y_k. \tag{3.16}$$

From (3.15), the error term can be written as:

$$\delta_i = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_j} \tag{3.17}$$

The partial derivative of the sigmoid function is used to find the error of the output layer by utilizing the target value [150]. The partial derivative of the sigmoid function is computed as follows:

$$\frac{\partial y_j}{\partial x_j} = y_j(1-y_j) \tag{3.18}$$

The partial derivative of the error term (3.17) is then computed for an output layer by using the derivative of (3.13) with respect to y_i [150] given by:

$$\frac{\partial E}{\partial y_j} = -(t_j - y_j) \tag{3.19}$$

To compute an error for output layer with respect to weights and the target value; (3.16), (3.18), and (3.19) are combined to produce:

$$\delta_j = \frac{\partial E}{\partial w_{jk}} = -(t_j - y_j)y_j(1-y_j)y_k. \tag{3.20}$$

The error for the nodes in the j-th hidden layer is computed as:

$$\delta_j := (\sum_{i \in I_j} \delta w_{ji})y_j(1-y_j). \tag{3.21}$$

Finally, the weight update rule is then given as:

$$\triangle w_{kj}(n) = \alpha \delta_j y_k + \eta \triangle w_{kj}(n\text{-}1) \tag{3.22}$$

where: α is the learning rate, y_k is the output of the node in layer k (the node upstream of the weight), n and $n-1$ are epochs in a loop, η is the momentum term, and δ_j is the error term for each node.

This study investigated the use of Xavier weight initialization technique to ascertain whether it could enhance the learning and the classification accuracy of the MLP network. The selection of initial weights is critical to the performance of the Back-Propagation algorithm [96]. If the initial weights are too small, the signal shrinks significantly as it passes from layer to layer until it is too small to be useful [96]. If the initial weights are too large, then the output signal grows exponentially until it is too big to produce useful results. If the algorithm is not initialized with proper weights, the loss function might not change even after thousands of iterations during learning. An effective method of initializing weights is to select values from the Gaussian distribution with zero mean [97]. Consider a linear neuron:

$$y = w_1x_1 + w_2x_2 + ... + w_Nx_N + b \tag{3.23}$$

The idea behind Xavier initialization is to try to keep the variance between x and y the same from layer to layer in order to prevent the output signal from exponentially growing to a high value or vanishing to zero causing the learning process to not be effective. The derivation of the Xavier equation for preservation of variance of x and y is given below.

$$var(y) = var(w_1x_1 + w_2x_2 + ... + w_Nx_N + b) \tag{3.24}$$

where: x_i is the i-th independent variable; w_i is the weight (coefficient) associated with the i-th independent variable, and b is the constant bias variable.

$$\therefore var(w_ix_i) = var(w_i)var(x_i) \tag{3.25}$$

$$\therefore var(y) = var(w_1)var(x_1) + ... + var(w_N)var(x_N) \tag{3.26}$$

$$\therefore (y) = N * var(w_i) * var(x_i) \tag{3.27}$$

$$\therefore var(w_i) = 1/N \tag{3.28}$$

Formula (3.28) gives the Xavier weight initialization rule where the initial weights are selected from a Gaussian distribution with zero mean and a variance of $1/N$. N is the number of input

nodes in the network. Formula (3.28) is effective in preserving the variance during the feed-forward phase. In order to preserve variance during back-propagation process, N becomes the average of the number of input neurons and output neurons. This causes equation (3.28) to be slightly modified to:

$$var(w_i) = 1/N_{avg} \tag{3.29}$$

where: $N_{avg} = (N_{in} + N_{out})/2$

The pseudocode of Back-Propagation algorithm using the Xavier technique is given in Algorithm 9.

Algorithm 9: Back-Propagation Algorithm with Xavier Technique, adapted from [94]

Require: A set D of m instances, where D = d_1, $d_2, d_3, ..., d_i, .., d_m$ is a set of training examples

Require: A MLP Network

Produce: A trained MLP Network

begin

    networkWeights
    $\leftarrow getXavierWeights(MLP.getNumOfInputNodes(), MLP.getNumOfOutputNodes())$;
    foreach *for each example e in training Set D* **do**
        inputVector $\leftarrow$ e's input vector;
        target $\leftarrow$e's output vector;
        output $\leftarrow$ ACTIVATE(network,inputVector);
        foreach *network output units* o_k **do**
            $\text{error}_{o_k} \leftarrow target_{o_k} - output_{o_k}$;
            $\triangle_{o_k} \leftarrow g'(in_{o_k}) * error_{o_k}$;
            $\text{sumOfDeltas}_{h_j} \leftarrow 0$;
            foreach *of network's hidden units* h_j **do**
                $w_{h_j, o_k} \leftarrow w_{h_j, o_k} + \alpha * output_{h_j} * \triangle_{o_k}$;
                $\text{sumOfDeltas}_{h_j} \leftarrow sumOfDeltas_{h_j} + w_{h_j, o_k} * \triangle_{o_k}$;
                foreach *of network's hidden units* h_j **do**
                    $\triangle h_j \leftarrow g'(in_{h_j}) * sumOfDeltas_{h_j}$;
                    foreach *of network's input units* s_i **do**
                        $w_{s_i, h_j} \leftarrow w_{s_i, h_j} + \alpha * s_i * h_j$;

3.2.3 Statistical Techniques

A statistical model describes the relationship between data attributes such as a dependent variable with independent variables [71]. The Statistical techniques which are used for classification include regression analysis and its various sub-categories such as linear regression, generalized linear models (Logistic Regression and Probit regression). The LR model is one of the most popular statistical technique for type II diabetes classification [87,143].

Improved Logistic Regression with $n-$restarts (LR-n)

The LR model is a popular statistical technique for classification because of its many strengths. It is quick to train, is resistant to over-fitting, and it does not make assumptions about distributions of classes in feature space [89]. This study proposes a modified LR model which uses a restart mechanism to create n LR models to diagnose type II diabetes. The goal of the LR technique is to find the best fitting model to describe the relationship between the dependent variable and predictor variables. Logistic Regression generates the coefficients of a formula to predict a probability of a binary response based on predictor variables by using a logistic function which is a cumulative logistic distribution. Suppose that Y is a dichotomous random variable which represents an outcome of an experiment, $X = (\mathrm{x}_1, x_2, ..., x_m)$ be a set of independent variables, α and w_k $(k = 1, 2, ..., m)$ be the weight parameters for each independent variable, and E(y) be the expected value of the dependent variable Y [88], then the Logistic Regression equation is given by:

$$E(y) = h_w(x) = \frac{1}{1 + e^{-(\sum_{k=1}^{m} w_i x_i + \alpha)}} \tag{3.30}$$

where:

$$h_w(x) = P(y = 1 | x; w) \tag{3.31}$$

$$1 - h_w(x) = P(y = 0|x; w). \tag{3.32}$$

$$\therefore P(y|x; w) = (h_w(x))^y (1 - (h_w(x))^{1-y} \tag{3.33}$$

Since the logistic function is used to predict probabilities, it can be fitted using the maximum likelihood function [88]. Maximum likelihood is used to find the smallest possible deviance between target values and predicted values using derivatives. Likelihood is computed as follows:

$$\begin{aligned} L(w) &= p(y|X; w) \\ &= \prod_{i=1}^{m} p(y_i|x_i; w) \\ &= \prod_{i=1}^{m} (h_w(x_i))^{y_i} (1 - h_w(x_i))^{1-y_i} \end{aligned} \tag{3.34}$$

where: $\prod_{i=1}^{m}$ is the logistic function.

The goal of logistic regression is to maximize the likelihood function. To efficiently achieve this goal, the log of w is used to get the log likelihood *l(w)* which is given by:

$$\begin{aligned} l(w) &= logL(w) \\ &= \sum_{i=1}^{m} y_i log(h(x_i)) + (1 - y_i) log(1 - h(x_i)) \end{aligned} \tag{3.35}$$

To find the best coefficients that maximize the likelihood function, the derivative of the function with respect to the weights is computed. The derivative is given as follows:

$$\begin{aligned} \frac{\partial}{\partial w^j} l(w) &= \left(\; y \tfrac{1}{g(wx)} - (1 - y) \tfrac{1}{1 - g(wx)} \; \right) \frac{\partial}{\partial w^j} g(wx) \\ &= \left(\; y \tfrac{1}{g(wx)} - (1 - y) \tfrac{1}{1 - g(wx)} \; \right) g(wx)(1 - g(wx)) \frac{\partial}{\partial w^j} wx \\ &= \left(\; y(1 - g(gx)) - (1 - y) g(wx) \; \right) x^j \\ &= (y - h(x)) x^j \end{aligned} \tag{3.36}$$

Weights (coefficients) are iteratively updated using the following Stochastic Gradient Ascent (SGA) rule:

$$w^{j+1} = w^j + \lambda (y_i - h_w(x_i)) x_i^j \tag{3.37}$$

To perform classification with Logistic Regression, the model makes a classification for a training example, then a classification error is calculated, and then the model is updated using SGA algorithm to reduce the error for the next classification. This process is repeated until an error is as small as possible or a set stopping condition is met. The model is then tested for accuracy using a test set. The LR-*n* algorithm enhances the standard LR algorithm by selecting the best model that fits the data well out of k logistic models. The model with the highest diabetes classification accuracy is selected as the best model.

To create, test and select the best model, the initial dataset D is partitioned into k samples of equal size at random without replacement. Each sample is further split into training set and validation set (e.g. using 90% train, 10% validation rule) to create a model. On each iteration of the algorithm, $k-1$ models are created using $k-1$ training samples and the model with the best classification rate on its validation set is selected to classify the remaining *k-th* sample. The coefficients of the model with the best classification rate are used to clasify instances on the *k-th* sample. The algorithm restarts *n*-times with random selection of k samples in order increase chances of getting a model that can accurately generalize on unseen data. Once the number of restarts reaches a set threshold, the classification accuracies of selected models are averaged to get the final classification accuracy of the algorithm. The approach of this algorithm maximizes chances of selecting training sets that best represent the entire dataset in order to generalize better on unseen data. The pseudocode of this algorithm is given in Algorithm 10.

Algorithm 10: Improved Logistic Regression with n restarts.

Require A dataset D with m instances
Input: A Number of restarts n
Input: A Number of samples k
Produce: A classification accuracy LR-n given all instances from D
begin
    samples $\leftarrow partition(D,\text{k})$; ▷ *Partition D into k samples of equal size*
    $i \leftarrow 0$; ▷ *Initialize counter for restarts elapsed*
    repeat
        $randomNumber \leftarrow getRandomNumber(lowerBound, upperBound)$;
        $finalTestSet \leftarrow getRandomSample(samples, randomNumber)$;
        samples.*remove(randomNumber);* ▷ *Remove final test set from samples*
        foreach *sample* k_i *in K-1 samples* **do**
            Split sample k_i into training set T_i and Validation set V_i;
            $coefficientsT_i \leftarrow 0$; ▷ *Initialise coeffs for all independent variables*
            $finalCoeffs \leftarrow SGA(trainingsetT_i)$; ▷ *SGA to find best coeffs*
            $classificationAccuracy \leftarrow getAccuracy(V_i, finalCoeffs)$;
            listOfFinalCoeffs.*add(finalCoeffs);*
            listOfAccuracies.*add(classificationAccuracy);*
        $bestCoeffs \leftarrow getBestModel(listOfFinalCoeffs, listOfAccuracies)$; ▷ *Get coefficients from the list which gave highest accuracy on* V_i
        $finalClassificationAccuracy \leftarrow getAccuracy(finalTestSet, bestCoeffs)$;
        listOfFinalAccuracies.*add(finalClassificationAccuracy);*
        i++;
        samples.*addSample(finalTestSet,randomNumber);* ▷ *Re-add for next iteration*
    until $i == n$;
    return *listOfFinalAccuracies/n;* ▷ *Avg of accuracies on finalTestSet in each restart*

3.3 Algorithm Performance Measures

Algorithm performance measures are used to determine the quality and the effectiveness of a classifier in achieving a desired goal. In this section, the following performance measures are described: Accuracy, Error Rate, Confusion Matrix, Sensitivity, Specificity, Positive Predictive Value (PPV), Negative Predictive Value (NPV), Receiver Operating Characteristic (ROC), Area Under the ROC Curve (AUC), and Chi-Square Goodness-of-fit test. In this study, these evaluation measures were used in order to extensively evaluate the experimental results of the methods proposed in this chapter.

1. **Accuracy** is defined as the total number of instances that were correctly classified by the classifier divided by the total number of instances in the dataset [151]. The equation for computing accuracy in a binary classification problem with classes labelled as negative or positive is given as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.38}$$

Where:

- **True Positive (TP)** is the number of instances from the positive class that were correctly classified as being positive by the classifier.
- **True Negative (TN)** is the number of instances from the negative class that that were correctly classified as being negative by the classifier.
- **False Positive (FP)** is the number of instances from the negative class that were incorrectly classified as being positive by the classifier.
- **False Negative (FN)** is the number of instances from the posit5ive class that were incorrectly classified as being negative by the classifier.

2. **Error Rate** is used to determine the misclassification rate of the classifier [151]. It is simply computed as 1-accuracy rate of the classifier.

3. **Confusion Matrix** is used to describe the performance of a classification model. A confusion matrix tabulates the number of correct and incorrect classifications as count values that are categorized by each class [152]. TP, TN, FP, and FN counts are used to create it.

4. **Sensitivity** represents the percentage of positive instances that were correctly classified as being positive by the classifier [151]. This measure is critical in the medical domain where the goal of the classifier is to diagnose a disease. For example, a classifier with a high accuracy rate of 95% and sensitivity of 15% is not useful in the diagnosis of a disease since it cannot identify a majority of positive cases. The equation for Sensitivity is defined as:

$$Sensitivity = \frac{TP}{TP+FN}s \tag{3.39}$$

5. **Specificity** is the percentage of negative instances that were correctly classified as negative by the classifier [151]. The formula for computing Specificity is given as:

$$Specificity = \frac{TN}{TN+FP} \tag{3.40}$$

6. **Positive Predictive Value (PPV)** is used to determine how likely an instance will be positive if the classifier classified it as positive [153]. This is measure is used in medical applications to determine the probability of a disease. PPV is defined as:

$$PPV = \frac{TP}{TP+FP} \tag{3.41}$$

7. **Negative Predictive Value (NPV)** is used to determine how likely an instance will be negative if the classifier classified it as negative [153]. NPV is defined as:

$$NPV = \frac{TN}{TN+FN} \tag{3.42}$$

8. **Receiver Operating Characteristic curve (ROC)** is used to illustrate the classification ability of a binary classifier as its discrimination threshold is varied [151]. The curve is created by plotting sensitivity against false positive rate which is simply computed as 1-specificity. The ROC curve can be used to decide the cut off value for classification when using continuous valued models like regression for binary classification.

9. **Area Under the ROC Curve (AUC)** is used to summarize the ROC curve results in a single number [90]. A classifier with AUC value between 90% to 100% is considered an excellent classifer [130]. A classifier with AUC between 80% and 89% is considered a good classifier [130]. A classifier with AUC between 70% and 79% is considered a fair classifier, and a classifier with AUC between 50% and 60% is considered a failure [130]. Several mathematical methods are used to estimate an AUC. The Riemann Sums, the Trapezoidal Rule, and Simpson's Rule are few examples of many methods that are used to compute the AUC. In this study the Trapezoidal Rule was used.

10. **Chi-Square Goodness-of-fit test** or simply Chi-Square test is a non-parametric statistical test that is used to determine if there is sufficient evidence in a sample of data to deduce that a certain condition (hypothesis) is true for the whole dataset [154] In this study, Chi-Square tests were conducted to test how well each model or classifier fits the diabetes dataset. Chi-Square test involves the construction of two statements: the null hypothesis *H0* and alternative hypothesis *Ha*. Null hypothesis assumes that there is no significant difference between the observed and the expected value. The alternative hypothesis assumes that there is a significant difference between the observed and the expected value.

Chapter 4

Experimental Results

The performance results of Machine Learning techniques and feature extraction approaches proposed in Chapter 3 are presented in this chapter. This study applied the following Machine Learning techniques: the standard K-Means Clustering Algorithm, the CVE-K-Means Clustering Algorithm, a Hybrid of CVE-K-Means and KNN Algorithm (CKNN), a Hybrid of CVE-K-Means and GRNN (KGRNN), an Ensemble of CVE-K-Means, Boosting, and GRNN(B-KGRNN), an Improved Multi-Layer Back-Propagation Neural Network (MLP-BPX), and an Improved Logistic Regression with *n*-restarts (LR-*n*). These techniques were applied to each of the following datasets: A-[Unprocessed], B-[Excl Missing], C-[Replaced by Mean], and D-[Extracted Features]. The performance of these ML techniques was evaluated using the following measures: classification accuracy, sensitivity, specificity, PPV, NPV, ROC Area, Chi-squared test, and computational time.

4.1 Feature Extraction

The dataset B-[Excl Missing] was used for discovering the most important features using the Random Forest algorithm. Five-fold cross validation was used during training and testing of the Ran-

dom Forest classifier. Weka ML tool was used to implement the Random Forest feature selection technique. Weka is a machine learning tool that allows researchers to use a collection of ML techniques for data mining tasks. The pairwise comparison matrix for AHP was constructed by considering all the attributes in dataset A-[Unprocessed] except the class variable. Feature selection using Random Forest algorithm and the AHP allowed the proposed ML techniques to improve their performance results by 10% on average.

4.1.1 Learner-based Feature Extraction using the Random Forest Algorithm

The results of the proposed learner-based feature selection technique are given in table 4.1. The

Attribute	**Number of Folds**	**Contribution Percentage**
Plasma Glucose Concentration at 2 Hours in an Oral Glucose Tolerance Test	5	100%
Diabetes Pedigree Function(PDF)	4	80%
Age in (years)	3	60%
Number of times pregnant	3	60%
Triceps Skin Fold Thickness (mm)	3	60%
2-Hour Serum Insulin Uh/ml)	2	40%
Diastolic Blood Pressure in mm Hg (PRESS)	1	20%
Body Mass Index (Weight in kg / (Height in m))	1	20%

Table 4.1 Learner-based Feature Extraction Results

results show that PDF and Plasma Glucose are the most significant features in classifying type II diabetes. Plasma Glucose was considered a significant feature in all 5 folds while PDF was considered significant in 4 folds. The results also show that Age, Number of Times Pregnant and Triceps Skin Fold Thickness are equally important features for diabetes classification and they are the third most important features. Insulin, Blood Pressure and Body Mass Index are the least important features. These features were only useful in classifying diabetes in one or 2 folds during cross validation.

4.1.2 AHP

The results of the leaner-based feature selection technique and the assumptions made on the significance of attributes in the dataset for Type II diabetes diagnosis were used to construct the pairwise comparison matrix. It was assumed that insulin and glucose levels are not directly influenced by genetics since their values are influenced by other attributes such as weight and dietary habits. Figure 4.1 gives a diagrammatic view of the AHP structure using attributes from the Pima Indian diabetes dataset.

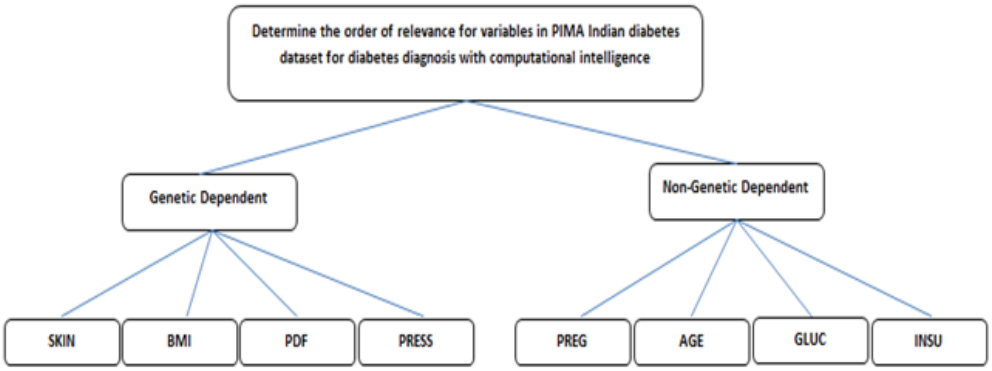

Figure 4.1 AHP hierarchy for diabetes feature importance

A pairwise comparison matrix for the diabetes dataset is given in figure 4.2. The column headings (from left to right) and row headings (from top to bottom) for the compared attributes in the matrix are ordered as follows: GLUC, PDF, AGE, BMI, PREG, PRESS, SKIN, INSU.

A decimal-based matrix (precise to two decimal places) was used for computations to find a priority vector. A priority vector is a maximum eigenvector that is derived from the pairwise comparison matrix. A priority vector is used to quantify the relative importance of attributes. Values of matrix **[M]** in Figure 4.2 were converted to decimals as shown in Figure 4.3. The matrices from Figure 4.4 to Figure 4.7 show the results of the intermediate steps taken to produce the final priority vector in Figure 4.8.

$$\mathbf{[M]} = \begin{bmatrix} 1 & 1 & 1 & 9 & 9 & 9 & 9 & 9 \\ 1 & 1 & 9 & 9 & 9 & 9 & 9 & 9 \\ \frac{1}{9} & \frac{1}{9} & 1 & 1 & 1 & 7 & 7 & 7 \\ \frac{1}{9} & \frac{1}{9} & 1 & 1 & 1 & 7 & 7 & 7 \\ \frac{1}{9} & \frac{1}{9} & 1 & 1 & 1 & 7 & 7 & 7 \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & 1 & 1 & 1 \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & 1 & 1 & 1 \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & 1 & 1 & 1 \end{bmatrix}$$

Figure 4.2 Pairwise Matrix for Diabetes Dataset

It can be observed in the AHP results, presented in Table 4.2, that PDF and Glucose concentration are ranked as the most important factors. Age, BMI and number of times pregnant are relevant by 8% each. High blood pressure, skin fold thickness and insulin level are individually relevant by 2%, which is extremely low. To compute a consistency ratio, the value of the principal eigenvalue and Consistency Index were used. The value of λ_{max} (principal eigenvalue) = 8.86, Consistency Index = 0.12 and the Consistency Ratio = 0.09. This consistency ratio falls within the acceptable value (less or equal to 0.1); hence one can be confident that the judgements made in this study for building a pairwise comparison matrix are not inconsistent and they are close to the truth. Furthermore, the results are in agreement with the results of the learner-based feature extraction

$$\mathbf{[M]} = \begin{pmatrix} 1.00 & 1.00 & 1.00 & 9.00 & 9.00 & 9.00 & 9.00 & 9.00 \\ 1.00 & 1.00 & 9.00 & 9.00 & 9.00 & 9.00 & 9.00 & 9.00 \\ 0.11 & 0.11 & 1.00 & 1.00 & 1.00 & 7.00 & 7.00 & 7.00 \\ 0.11 & 0.11 & 1.00 & 1.00 & 1.00 & 7.00 & 7.00 & 7.00 \\ 0.11 & 0.11 & 1.00 & 1.00 & 1.00 & 7.00 & 7.00 & 7.00 \\ 0.11 & 0.11 & 0.14 & 0.14 & 0.14 & 1.00 & 1.00 & 1.00 \\ 0.11 & 0.11 & 0.14 & 0.14 & 0.14 & 0.10 & 1.00 & 1.00 \\ 0.11 & 0.11 & 0.14 & 0.14 & 0.14 & 0.10 & 1.00 & 1.00 \end{pmatrix}$$

Figure 4.3 Pairwise Comparison Matrix in Decimal Format

$$[\overline{\mathbf{M}}] = \begin{pmatrix} 7.06 & 7.06 & 32.78 & 40.78 & 40.78 & 178.00 & 178.00 & 178.00 \\ 7.94 & 7.94 & 40.78 & 48.78 & 48.78 & 234.00 & 234.00 & 234.00 \\ 2.86 & 2.86 & 7.04 & 7.92 & 7.92 & 43.98 & 43.98 & 43.98 \\ 2.86 & 2.86 & 7.04 & 7.92 & 7.92 & 43.98 & 43.98 & 43.98 \\ 2.86 & 2.86 & 7.04 & 7.92 & 7.92 & 43.98 & 43.98 & 43.98 \\ 0.60 & 0.60 & 1.94 & 2.82 & 2.82 & 7.92 & 7.92 & 7.92 \\ 0.60 & 0.60 & 1.94 & 2.82 & 2.82 & 7.92 & 7.92 & 7.92 \\ 0.60 & 0.60 & 1.94 & 2.82 & 2.82 & 7.92 & 7.92 & 7.92 \end{pmatrix}$$

Figure 4.4 Squared and Normalized Comparison Matrix.

$$w = \begin{pmatrix} 0.37 \\ 0.35 \\ 0.09 \\ 0.07 \\ 0.07 \\ 0.02 \\ 0.02 \end{pmatrix}$$

Figure 4.5 First Eigenvector

$$\overline{[\mathbf{M_1}]} = \begin{pmatrix} 751.28 & 751.28 & 2360.25 & 3043.75 & 3043.75 & 12166.67 & 12166.67 & 12166.67 \\ 933.28 & 933.28 & 2919.86 & 3786.4 & 3786.4 & 14915.31 & 14915.31 & 14915.31 \\ 187.00 & 187.00 & 627.42 & 809.42 & 809.42 & 3229.55 & 3229.55 & 3229.55 \\ 187.00 & 187.00 & 627.42 & 809.42 & 809.42 & 3229.55 & 3229.55 & 3229.55 \\ 187.00 & 187.00 & 627.42 & 809.42 & 809.42 & 3229.55 & 3229.55 & 3229.55 \\ 44.79 & 44.79 & 143.31 & 180.43 & 180.43 & 767.18 & 767.18 & 767.18 \\ 44.79 & 44.79 & 143.31 & 180.43 & 180.43 & 767.18 & 767.18 & 767.18 \\ 44.79 & 44.79 & 143.31 & 180.43 & 180.43 & 767.18 & 767.18 & 767.18 \end{pmatrix}$$

Figure 4.6 Second Squared Matrix

$$\overline{[\mathbf{M_2}]} = \begin{pmatrix} 4480054.58 & 4480054.58 & 14498107.70 & 18554950.04 & 18554950.04 & 75630774.51 & 75630774.51 & 75630774.51 \\ 5538361.30 & 5538361.30 & 17923875.79 & 22941081.36 & 22941081.36 & 93490021.25 & 93490021.25 & 93490021.25 \\ 1168993.34 & 1168993.34 & 3785247.94 & 4843554.78 & 4843554.78 & 19751683.55 & 19751683.55 & 19751683.55 \\ 1168993.34 & 1168993.34 & 3785247.94 & 4843554.78 & 4843554.78 & 19751683.55 & 19751683.55 & 19751683.55 \\ 1168993.34 & 1168993.34 & 3785247.94 & 4843554.78 & 4843554.78 & 19751683.55 & 19751683.55 & 19751683.55 \\ 272809.27 & 272809.27 & 882659.05 & 1129272.50 & 1129272.50 & 4606909.33 & 4606909.33 & 4606909.33 \\ 272809.27 & 272809.27 & 882659.05 & 1129272.50 & 1129272.50 & 4606909.33 & 4606909.33 & 4606909.33 \\ 272809.27 & 272809.27 & 882659.05 & 1129272.50 & 1129272.50 & 4606909.33 & 4606909.33 & 4606909.33 \end{pmatrix}$$

Figure 4.7 Final Squared Matrix

$$w_{max} = \begin{pmatrix} 0.31 \\ 0.39 \\ 0.08 \\ 0.08 \\ 0.08 \\ 0.02 \\ 0.02 \end{pmatrix}$$

Figure 4.8 Final Eigenvector and Priority Vector

technique proposed in this study. Feature extraction using the Random Forest algorithm and the AHP led to the dataset D-[Extracted Features] to be comprised of only 5 features instead of 9 original features. The most important features that were selected are: GLU, PDF, AGE, PREG, and SKIN.

Attribute	Ordered w_3	Relevance (%)	Rank
PDF	0.39	39%	1
GLU	0.31	31%	2
AGE	0.08	8%	3
BMI	0.08	8%	3
PREG	0.08	8%	3
PRESS	0.02	2%	4
SKIN	0.02	2%	4
INS	0.02	2%	4

Table 4.2 Relative Importance of Attributes

4.2 Instance-based Techniques

All the datasets presented in Chapter 3 were used to train and test the performance of the standard K-Means clustering algorithm and the new CVE-K-Means algorithm. Nine-fold cross validation was used to train and test both K-Means and the CVE-K-Means technique on each dataset. The CVE-K-Means technique outperformed the standard K-Means clustering algorithm by 3% in terms of classification accuracy.

4.2.1 K-Means Clustering Algorithm

K-Means clustering algorithm consistently achieved the lowest results for different number of clusters on the un-preprocessed dataset compared to the results obtained using the derived datasets. The algorithm achieved the highest classification accuracy of 77% and sensitivity rate of 55% on dataset B-[Excl Missing] using 10 clusters. The classification accuracy on dataset C-[Replaced by Mean] and D-[Extracted Features] was 75% on each dataset. The classification accuracy of 73%

and 30% sensitivity were obtained using the dataset A-[Unprocessed]. This result was the lowest compared to other datasets. The datasets in which the algorithm produced poor and better results were considered to investigate the influence of different values of k on the performance of the algorithm. It was discovered that the algorithm produced poor results on dataset B-[Excl Missing] as the value of k became larger while on dataset A-[Unprocessed] the opposite occurred. This is illustrated in Figure 4.9.

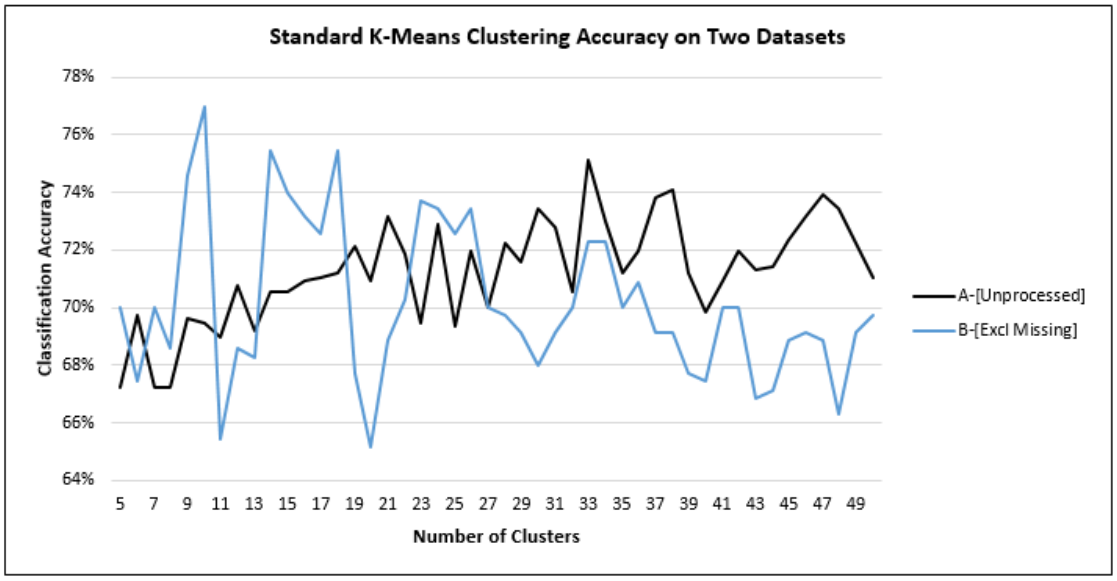

Figure 4.9 Standard K-Means classification accuracy vs k-value

4.2.2 CVE-K-Means Clustering Algorithm

CVE-K-Means algorithm was able to identify more positive cases than the standard algorithm during testing. The highest classification accuracy achieved by CVE-K-Means was 80% with sensitivity of 61% using 12 clusters. The algorithm produced these results on the dataset B-[Excl Missing] using 9-fold cross validation. The classification accuracy on dataset C-[Replaced by Mean] and D-[Extracted Features] was 78% and 79% respectively. On the unprocessed dataset A-[Unprocessed] the algorithm achieved the classification accuracy of 77%. This study consid-

ered the dataset A-[Unprocessed] in which both the standard K-Means clustering algorithm and the CVE-K-Means algorithm produced poor results compared to other datasets and investigated how different values of k influenced their performance. After experimentation, it was observed that the algorithms performed better when k values were between 10 and 21. These results are shown in Figure 4.10. It can also be noticed in Figure 4.10 that CVE-K-Means shows a consistent improvement in the classification accuracy for all values of k compared to the standard algorithm.

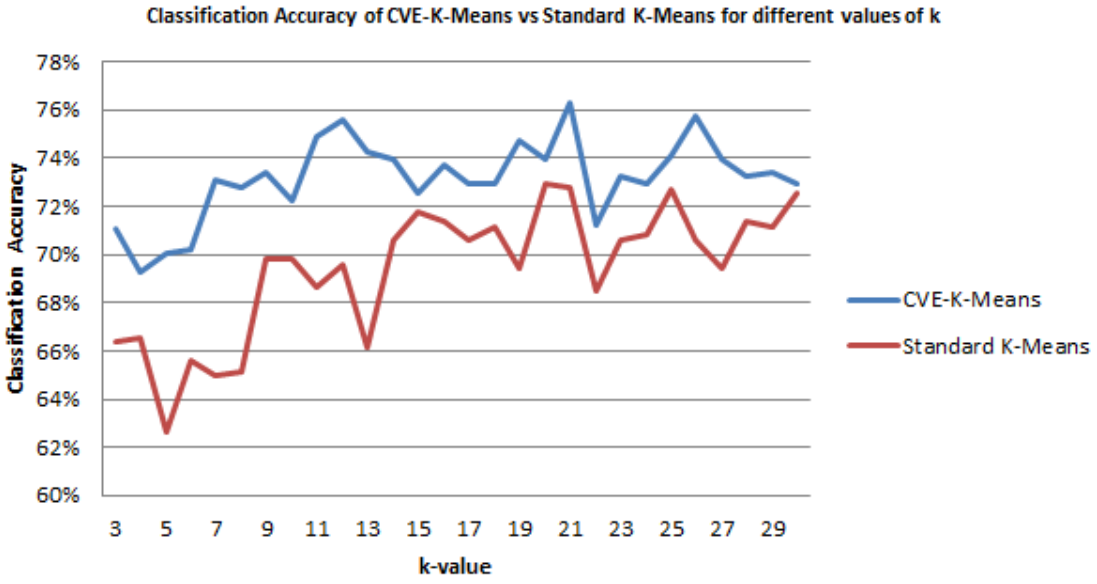

Figure 4.10 CVE-K-Means classification accuracy vs Standard K-Means using different values of k

Figure 4.11 gives an overall performance comparison of the accuracy and execution times of the Standard K-Means algorithm and CVE-K-Means algorithm using the same number of clusters (k) and the same dataset A-[Unprocessed]. The standard K-Means algorithm was slightly tweaked to also search for an optimal value of k, as in CVE-K-Means, to fairly compare their execution times and classification accuracy. The results show that CVE-K-Means consistently found solutions in less time compared to the standard algorithm for all values of k, even after finding an optimal value of k for the standard K-Means algorithm. Furthermore, it can be observed that the number

of clusters used is directly proportional to the execution time of the algorithm since an algorithm has to consider many instances during classification.

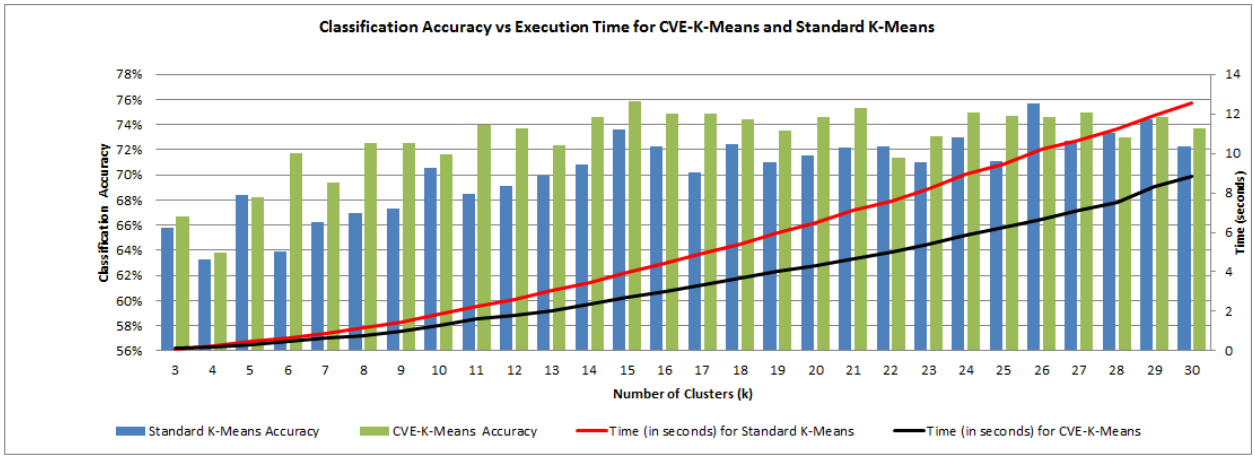

Figure 4.11 CVE-K-Means versus Standard K-Means classification accuracy for each *k*-value over time

Figure 4.12 provides an overall summary of CVE-K-Means' performance measures across all four datasets. Although the CVE-K-Means algorithm shows improvement in the classification accuracy accross all the datasets, it can be observed in Figure 4.12 that the algorithm failed to identify a large number of positive cases since its sensitivity values range between 20% and 61%.

The best classification accuracy that was achieved by the CVE-K-Means algorithm is 3% higher than the best classification accuracy known in literature by other improved variants of K-Means clustering. In literature, the best classification accuracy was obtained by an improved K-Means clustering algorithm proposed by [22] using Pima Indian dataset. Table 4.3 provides a comparison of results between the CVE-K-Means algorithm and other K-Means clustering variants in literature which were outperformed by the CVE-K-Means algorithm in terms of classification accuracy.

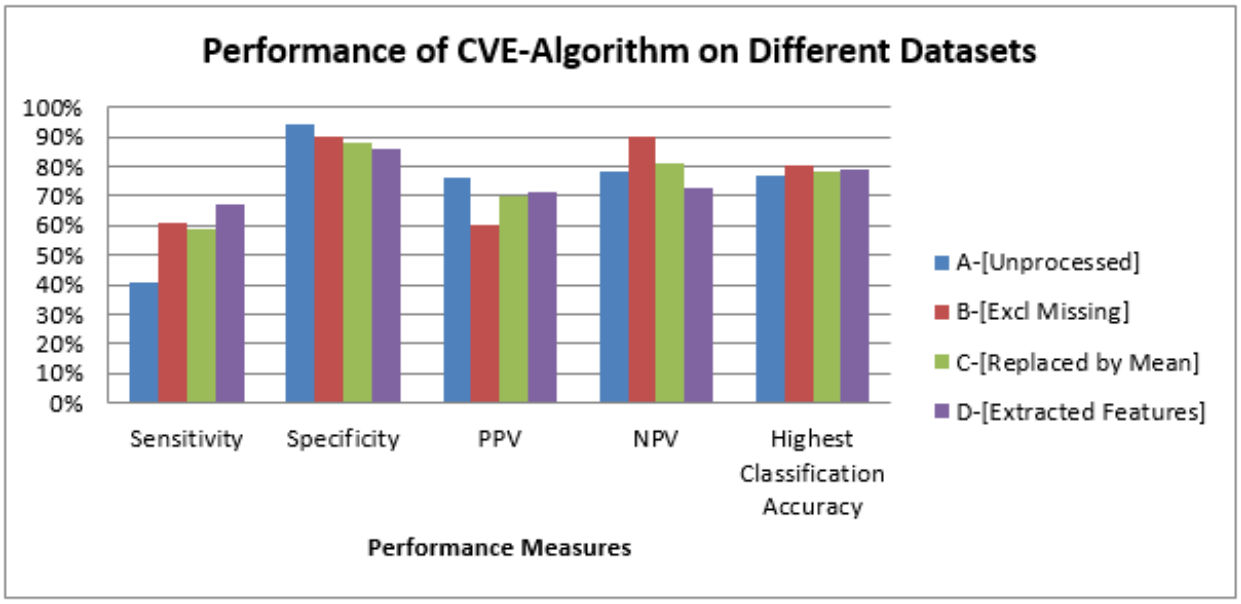

Figure 4.12 Performance measures for CVE-K-Means algorithm

Year	Reference	ML Technique	Accuracy	*k*-value	Dataset
2017	**This study**	**CVE-K-Means**	**80%**	**12**	**B-[Excl Missing]**
2016	[146]	K-Means	77%	Not Reported	Pima Indian
2014	[22]	K-Means	77%	Not reported	Pima Indian
2013	[75]	NovelK-Means	69%	2	Pima Indian
2012	[74]	K-Means	52%	3	Pima Indian

Table 4.3 Accuracy comparison of CVE-K-Means Algorithm with other studies

4.2.3 Hybrid of CVE-K-Means and KNN Algorithm (CKNN)

The performance of the new CKNN was tested using all four datasets. Twenty-five cluster centers produced by CVE-K-Means algorithm were used as the training set with 10-fold cross validation and three nearest neighbours for classification. The optimal number of neighbours and clusters were found through trial and error. CKNN achieved the highest classification accuracy of 81%

with 89% sensitivity and 88% specificity on dataset B-[Excl Missing]. The algorithm achieved the second highest classification accuracy of 80% on dataset D-[Extracted Features], followed by dataset C-[Replaced by Mean] with classification accuracy of 78% and dataset A-[Unprocessed] as the lowest with classification accuracy of 75%. The performance results of the CKNN algorithm are given in figure 4.13.

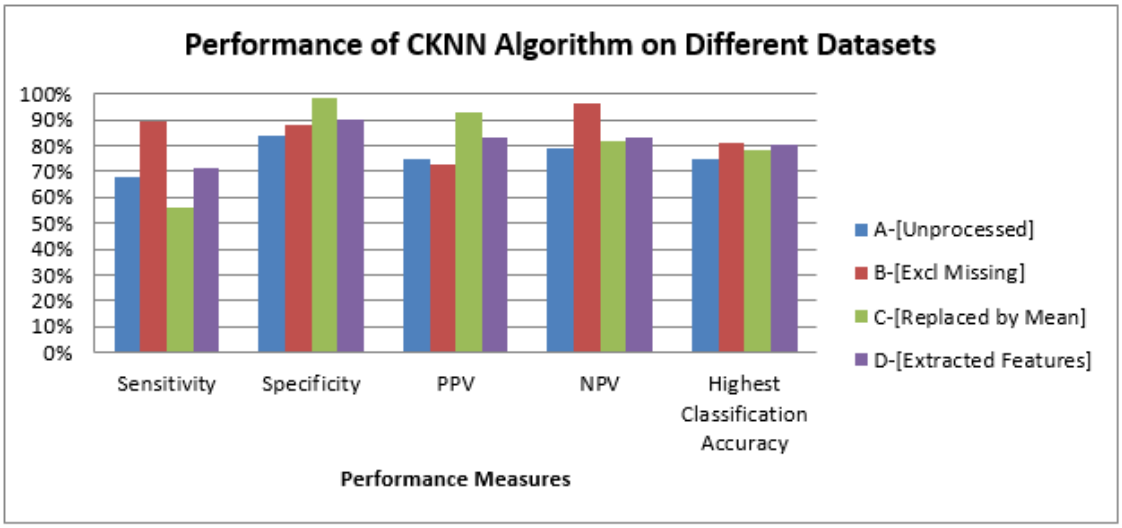

Figure 4.13 Performance measures for CKNN algorithm.

The un-preprocessed dataset, A-[Unprocessed], was used to investigate the impact of using cluster centers as the training set on the computation time during the classification stage. To achieve this, the algorithm was tested using different number of cluster centers (between 10 and 300) and for each iteration the time it took to classify test instances was recorded. The outcomes of this investigation are given in Figure 4.14. It can be observed from the results that as the number of clusters increases, the computation time for the classification stage also increases. This increase in computation time is due to the high number of potential neighbours that have to be assessed for their proximity to the test instance before the instance is classified. The benefits of using cluster centres as the training set instead of the whole set would be even more evident in datasets which contain millions of instances.

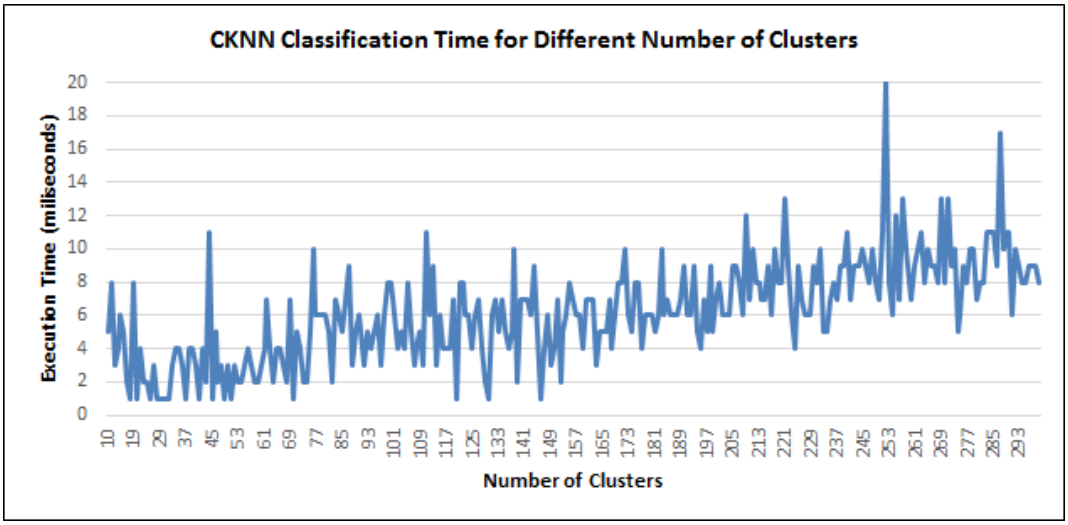

Figure 4.14 CKNN Classification Duration for Different Number of Clusters

Year	Reference	ML Technique	Accuracy	*k*-value	Dataset
2017	**This study**	**CKNN**	**81%**	**3**	**B-[Excl Missing]**
2014	[8]	KNN	75%	5	Pima Indian
2014	[22]	Amalgam kNN	80%	Not reported	Pima Indian
2013	[72]	Class-wise KNN	78%	10	Pima Indian
2009	[73]	KNN-Naive-Bayes	76%	10	Pima Indian
2003	[23]	kNNModel	75%	5	Pima Indian

Table 4.4 Accuracy comparison of CKNN Algorithm with other KNN algorithms.

Table 4.4 compares the performance of the CKNN algorithm to similar techniques reported in literature which were outperformed by the CVE-K-Means algorithm in terms of classification accuracy.

4.3 Artificial Neural Networks

The KGRNN network was configured with nine neurons in the input layer, forty-nine neurons in the hidden layer, two neurons in the summation layer and one neuron in the output layer {9,49,2,1}. Nine-fold cross validation and k = 12 for the CVE-K-Means algorithm were used by the network for all datasets. Dataset A-[Unprocessed] was used to train and test the B-KGRNN technique since all ML techniques proposed in this study achieved the lowest classification accuracy on this dataset. Dataset A-[Unprocessed] was chosen as an ideal candidate for testing the impact of the boosting technique since it has missing values that can negatively impact the performance of classifiers. To make the boosting ensemble more effective, dataset A-[Unprocessed] was slightly modified by normalizing all attribute values to be in the scale of [0,1]. The MLP-BPX network was designed to have a topology of nine input neurons, two hidden layers with nine neurons in the first hidden layer, seven neurons in the second hidden layer and one output neuron in the output layer {9,9,7,1}. The topology and the learning rate with best performance were found through experimentation. The network was trained and tested using 5-fold cross validation on all datasets. The B-KGRNN technique outperformed KGRNN and MLP-BPX in terms of classification accuracy, sensitivity, and specificity. The accuracy of B-KGRNN was 18% and 14% higher than the accuracy of MLP-BPX and KGRNN respectively. Compared to the MLP-BPX and KGRNN, the sensitivity of B-KGRNN was superior by 18% and 28% respectively. The specificity of B-KGRNN was superior to MLP-BPX and KGRNN by 17% and 12% respectively.

4.3.1 Hybrid of CVE-K-Means and GRNN (KGRNN)

With the selected configuration of 49 neurons (using 49 centroids) in the hidden layer and the smoothing factor $\sigma = 0.19$, the network achieved the highest classification accuracy of 86% with 83% sensitivity and 87% specificity using dataset D-[Extracted Features]. KGRNN achieved the second highest classification accuracy of 85% on dataset B-[Excl Missing] using $\sigma = 0.08$ and 49 centroids as training instances. The network performed poorly on two datasets A-[Unprocessed] and C-[Replaced by Mean]. It achieved the highest classification accuracy of 66% on dataset A-[Unprocessed] using 49 centroids and $\sigma = 0.09$. On dataset C-[Replaced by Mean] it achieved the highest classification accuracy of 71% using 49 centroids as training set and $\sigma = 0.05$. Figure 4.15 provides the performance results for the KGRNN technique using forty-nine training centroids and the different values of σ given above.

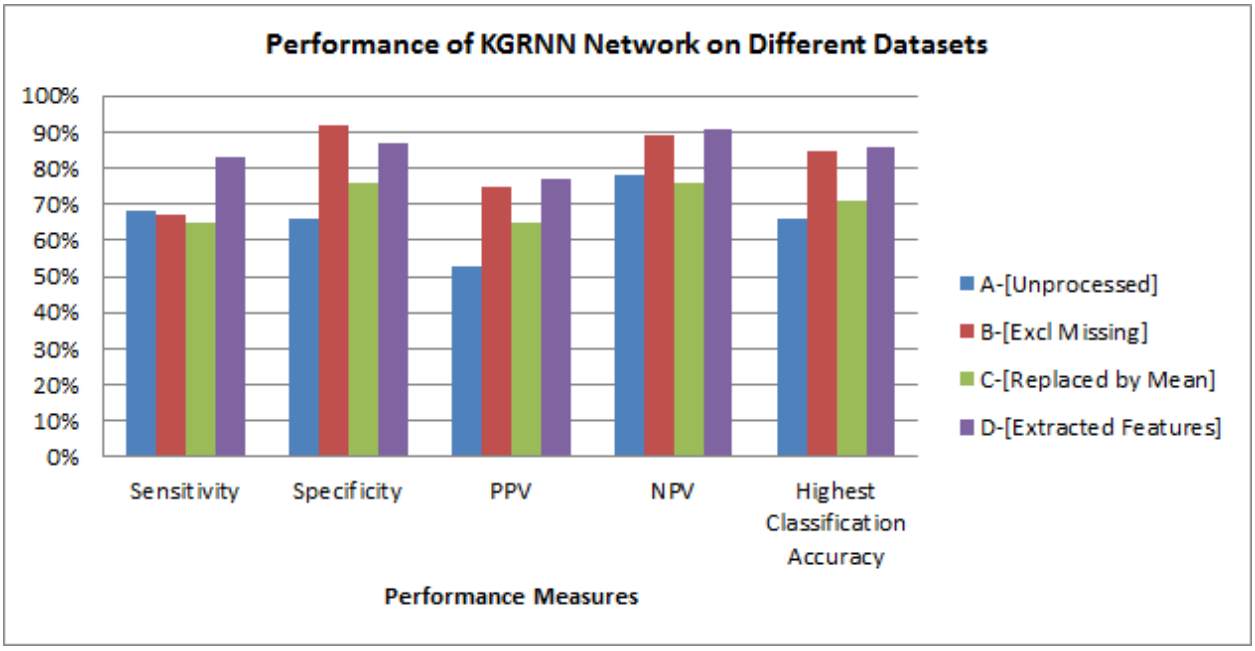

Figure 4.15 Performance measures for KGRNN network.

For KGRNN to be more effective, it requires a balance between the smoothing factor σ which

influences the classification accuracy and the number of cluster centers which influence the computational effort of the network. As shown in Figure 4.16 below, the majority of extreme solutions (very high and low accuracies) are concentrated in the region with smaller values of the smoothing factor. This is due to the strictness of the network in the selection of centroids which influence the classification outcome of a test instance as the smoothing factor approaches zero. This behaviour of the network can also be noticed across all the datasets used in this study as shown in Figure 4.17. The results in Figure 4.16 were obtained by training and testing the network using dataset D-[Extracted Features].

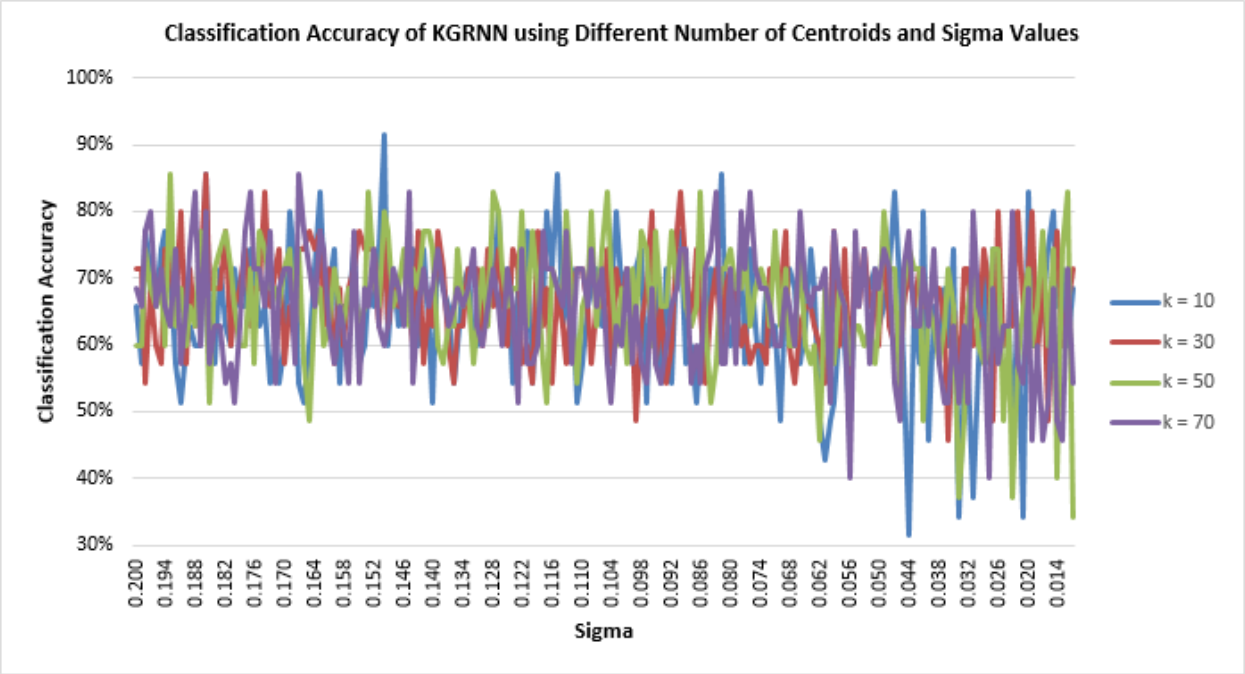

Figure 4.16 KGRNN accuracy using different values of sigma and number of cluster centres

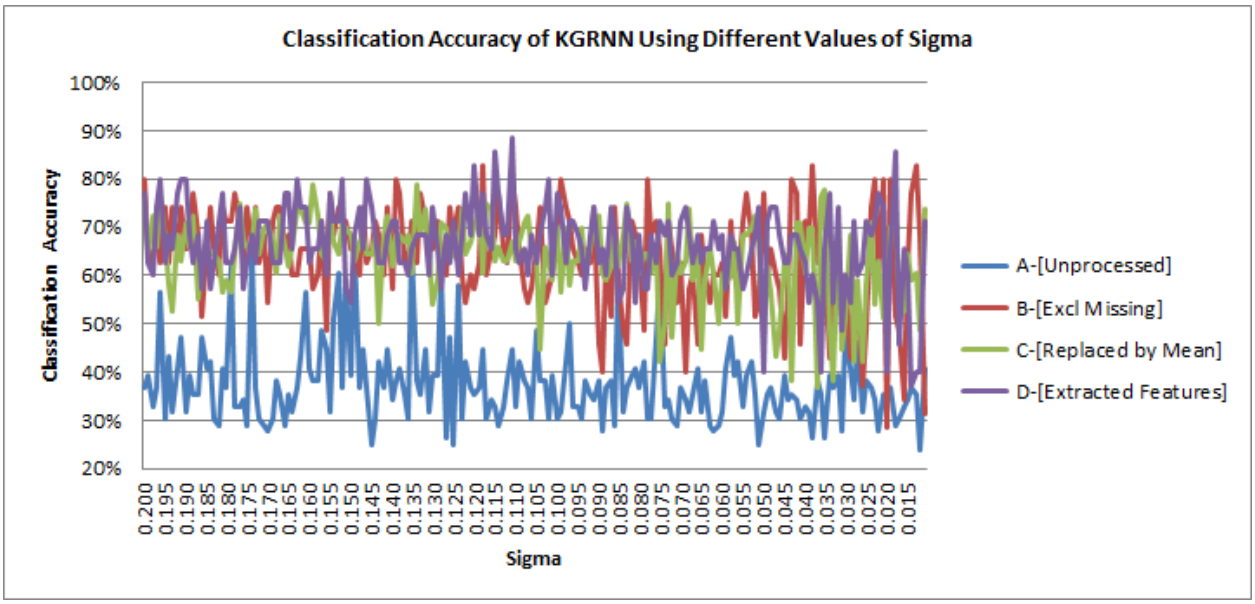

Figure 4.17 KGRNN accuracy using different values of sigma and datasets

The performance of KGRNN was evaluated by considering the training and classification time as the number of centroids for training was incrementally increased. The results in Figure 4.18 show that as the number of centroids increased from 5 to 140, the time it took the network to train and classify test cases also steadily increased. Figure 4.18 shows that it took longer to train and classify when the un-processed dataset was used.

The ROC curve for KGRNN is given in Figure 4.19. The AUC of the KGRNN ROC curve is 87%. This shows that the network is a good classifier. The ROC curve was plotted by training and testing the network using dataset D-[Extracted Features], 49 centroids as training set and $\sigma = 1.9$.

Compared to other GRNN variants proposed in literature, the KGRNN technique outperformed the standard GRNN technique proposed by Alby and Shivakumar [140] by 1%. The GRNN proposed in [140] obtained the highest classification accuracy known in literature for a GRNN. Table 4.5 compares the performance of KGRNN to other ANNs in literature which were outperformed by the KGRNN technique in terms of classification accuracy.

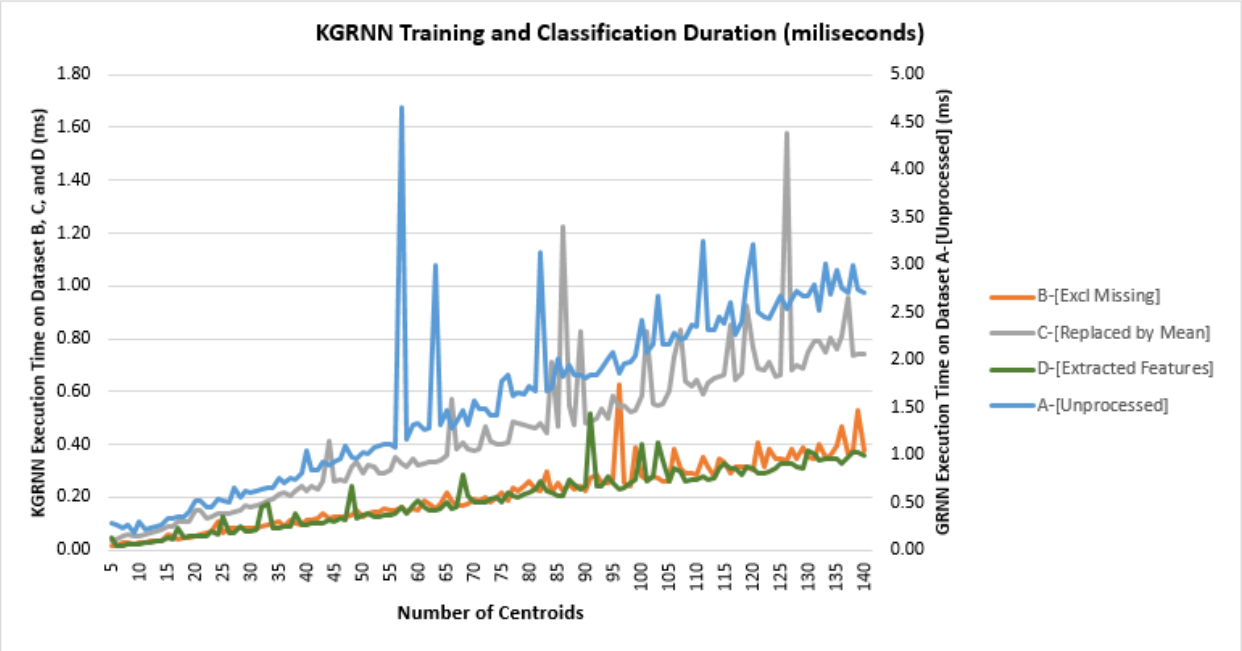

Figure 4.18 KGRNN Execution Time for Different Number of Centroids.

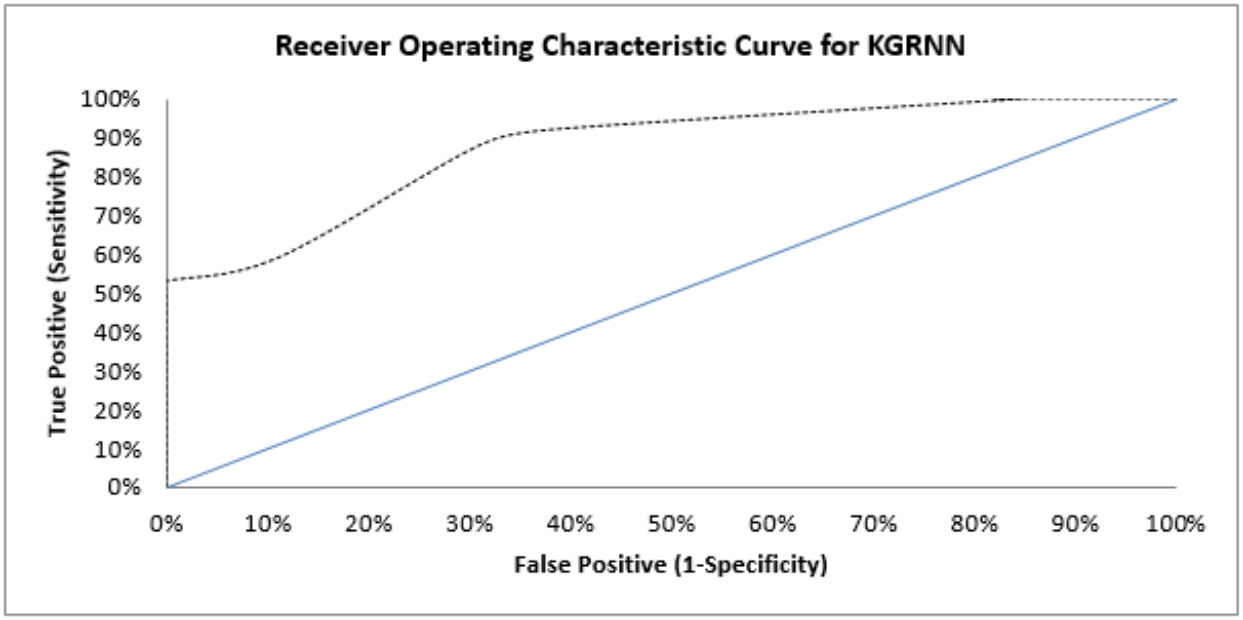

Figure 4.19 ROC Curve for the KGRNN Network

Year	Reference	ML Technique	Accuracy	Sigma	Dataset
2017	**This study**	**KGRNN**	**86%**	**0.19**	**D-[Extracted Features]**
2017	[140]	GRNN	85%	Not Reported	Pima Indian
2017	[141]	RBM ANN	85%	N/A	Pima Indian
2011	[102]	GRNN	84%	Not Reported	W. G. Unit - Nigeria
2016	[83]	CNN	83%	N/A	Pima Indian
2016	[82]	PNN	81%	N/A	Pima Indian
2003	[79]	GRNN	80%	2.5	Pima Indian
2016	[84]	GA-MLPNN	79%	N/A	Pima Indian
2017	[81]	RBFN	70%	N/A	Pima Indian and Synthetic
2014	[135]	VGRNN	57%	Not reported	Pima Indian

Table 4.5 Accuracy comparison of the KGRNN network with other ANNs

4.3.2 Ensemble of CVE-K-Means, Boosting and GRNN(B-KGRNN)

The B-KGRNN technique achieved the highest classification accuracy of 100% with equal sensitivity and specificity rates of 100%. These results were obtained by the proposed technique using 18 centroids and the σ value of 5. For σ values less than 0.02, the network performed slightly below 100% for all performance measures due to the strictness of the smoothing factor σ.

Ten weak classifiers of KGRNN were combined to form one strong classifier using boosting. Table 4.6 provides a list of contribution weight values (alpha) for ten weak classifiers obtained with smoothing factor 5 and eighteen centroids. It can be seen from the results in Table 4.6 that the classifier with the lowest training error has the highest weight in the classification decision. This phenomenon can also be visualized in Figure 4.20 that shows that the weight of the classifier is inversely proportional to its training error.

Weak Classifier $\mathbf{h}_x$	α	α_{value}	Training Error
$h_1(x)$	α_1	0.39	0.32
$h_2(x)$	α_2	0.43	0.30
$h_3(x)$	α_3	0.49	0.27
$h_4(x)$	α_4	0.53	0.26
$h_5(x)$	α_5	0.58	0.24
$h_6(x)$	α_6	0.69	0.20
$h_7(x)$	α_7	0.78	0.17
$h_8(x)$	α_8	0.91	0.14
$h_9(x)$	α_9	1.01	0.12
$h_{10}(x)$	α_{10}	1.47	0.05

Table 4.6 Weight contribution in the classification for each weak classifier

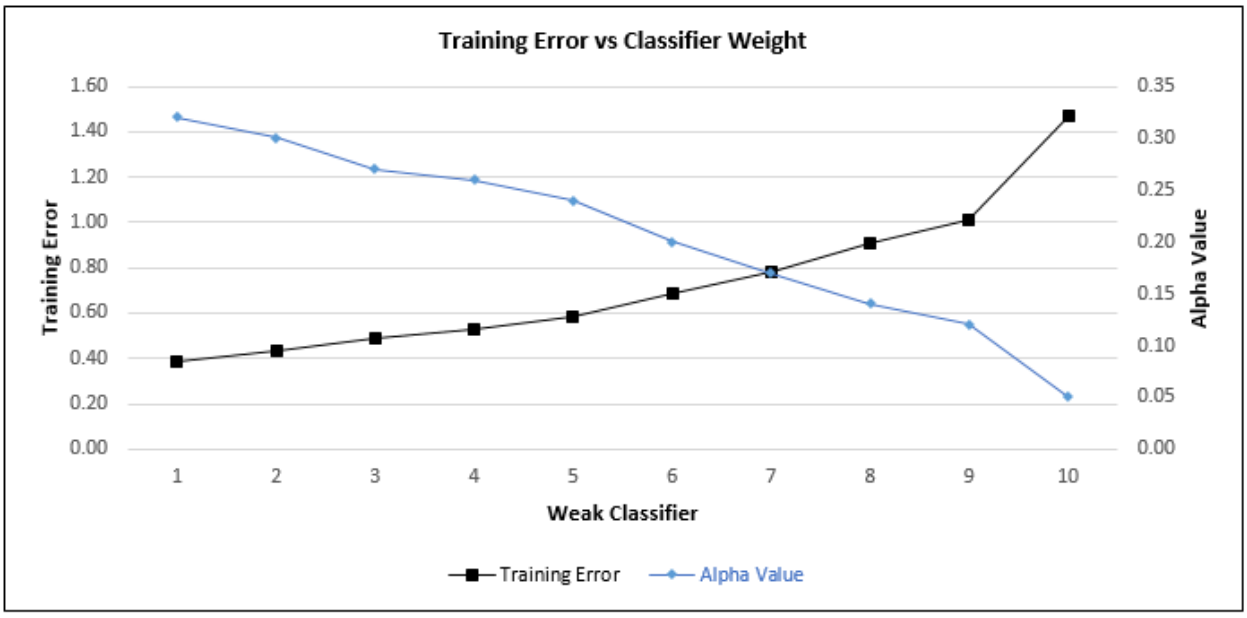

Figure 4.20 Relationship Between Classifier Weight and Training Error

The final strong classifier H can be given as an equation using the values in Table 4.6 as follows:

$$H(x) = \alpha_1 * h_1(x) + \alpha_2 * h_2(x) + \alpha_3 * h_3(x) + \alpha_4 * h_4(x) + \alpha_5 * h_5(x) + \alpha_6 * h_6(x) + \alpha_7 * h_7(x) + \alpha_8 * h_8(x) + \alpha_9 * h_9(x) + \alpha_{10} * h_{10}(x) \tag{4.1}$$

Where x is an instance to be classified; h_1, h_2,..., h_{10} are weak GRNN classifiers with their respective classification weights α_i. B-KGRNN improved the lowest classification accuracy of KGRNN by 44% using the same dataset A-[unprocessed]. B-KGRNN achieved a 15% better classification accuracy than the highest classification accuracy achieved by the GRNN in [87]. Table 4.7 provides the comparison of classification accuracy between B-KGRNN versus other boosted and "un-boosted" variants of ANNs which were outperformed by the B-KGRNN technique in terms of classification accuracy.

Year	Reference	ML Technique	Accuracy	Sigma	Dataset
2017	**This study**	**B-KGRNN**	**100%**	**5**	**A-[Unprocessed]**
2017	**This study**	**KGRNN**	**86%**	**0.19**	**D-[Extracted Features]**
2015	[87]	BT+ANN	85%	N/A	Sawanpracharak RH
2016	[83]	CNN	83%	N/A	Pima Indian
2016	[82]	PNN	81%	N/A	Pima Indian
2003	[79]	GRNN	80%	2.5	Pima Indian
2016	[84]	GA-MLPNN	79%	N/A	Pima Indian
2017	[81]	RBFN	70%	N/A	Pima Indian
2014	[135]	VGRNN	57%	Not reported	Pima Indian

Table 4.7 Comparison of the B-KGRNN with other ANN variants

4.3.3 Improved Multi-Layer Back-Propagation Neural Network (MLP-BPX)

The best classification accuracy that was obtained by the proposed network on test instances using the learning rate of 0.67 and 500 training epochs was 82% with 72% sensitivity and 73% specificity. The MLP-BPX network achieved a classification accuracy of 84% on the training set during 5-fold cross validation. These results were obtained on the dataset B-[Excl Missing]. The proposed network achieved the second highest classification accuracy of 81% on dataset D-[Extracted Features], followed by the classification accuracy of 79% on dataset C-[Replaced by Mean]. The lowest classification accuracy achieved by the proposed network was 71% on dataset A-[Unprocessed]. The same network configuration was used for all four datasets. The learning rate of 0.67 did not only produce a good classification accuracy but it also enabled the network to produce acceptable sensitivity and specificity results. Figure 4.21 shows performance results of the MLP-BPX network on the different datasets.

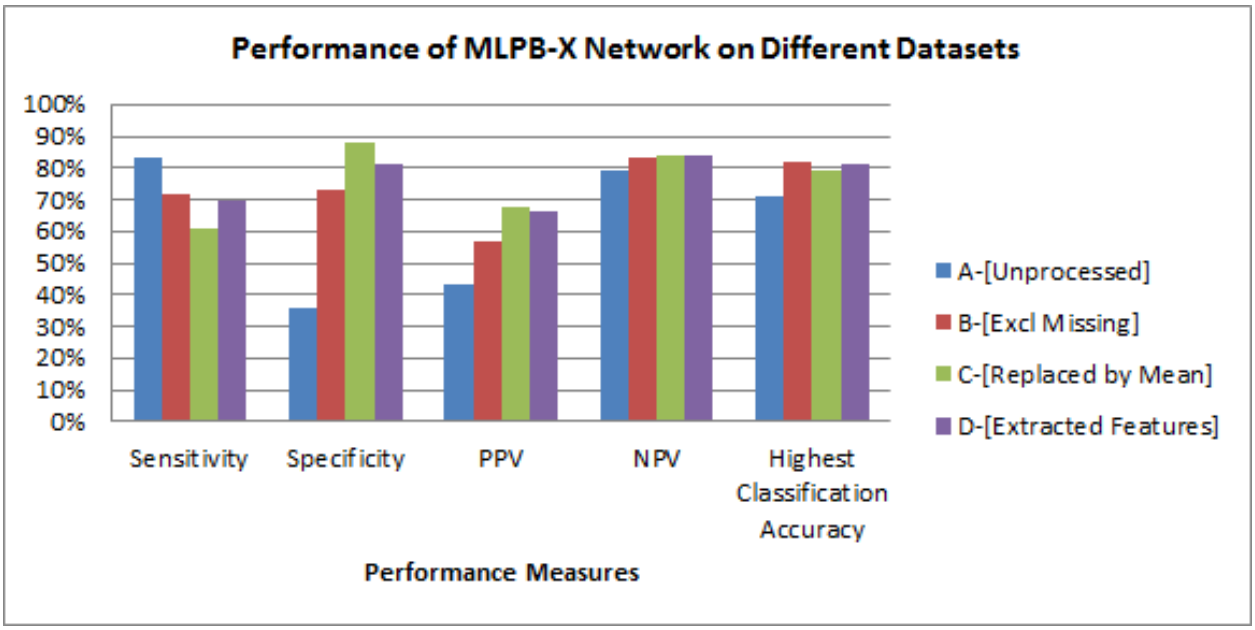

Figure 4.21 Performance measures for MLP-BPX network

Figure 4.22 gives the comparison of classification accuracies achieved by MLP-BPX on test instances from different datasets during cross validation. Figure 4.23 provides a performance comparison between different training sets during cross validation.

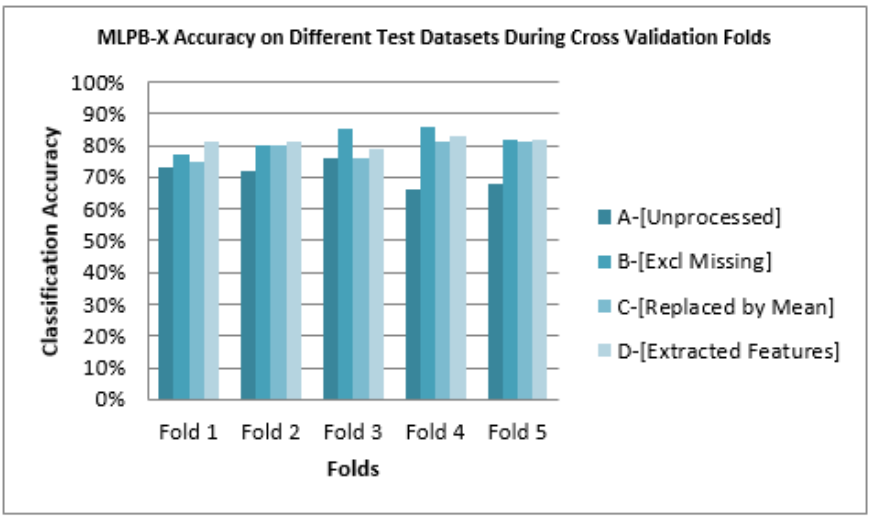

Figure 4.22 MLP-BPX accuracy on different test sets for each fold

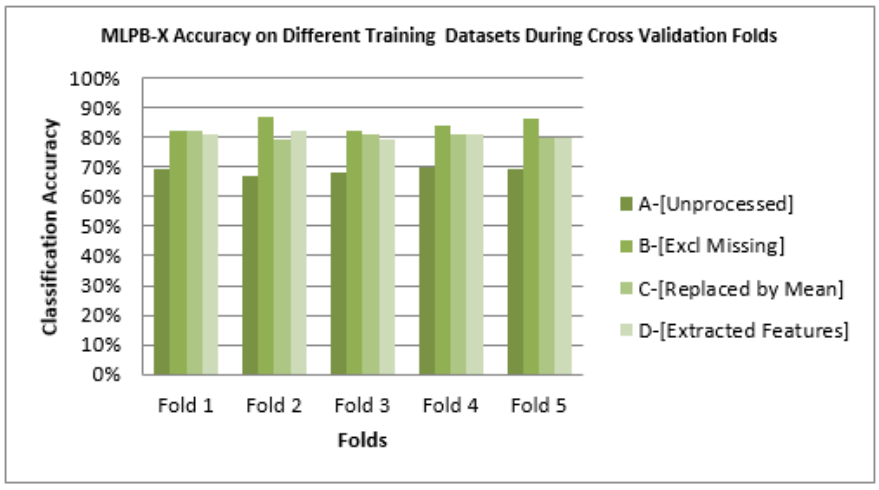

Figure 4.23 MLP-BPX accuracy on different train sets for each fold

Figure 4.24, 4.25, 4.26, 4.27, and 4.28 show a cross validation training and test error trend of the network on dataset B-[Excl Missing] in which the network achieved its highest classification accuracy.

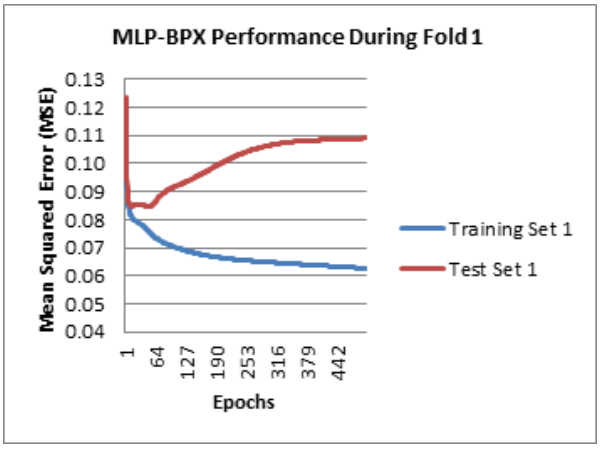

Figure 4.24 MLP-BPX Fold 1 on B-[Excl Missing]

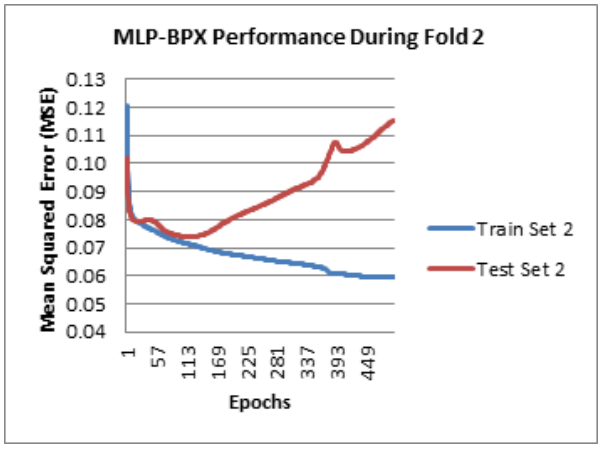

Figure 4.25 MLP-BPX Fold 2 on B-[Excl Missing]

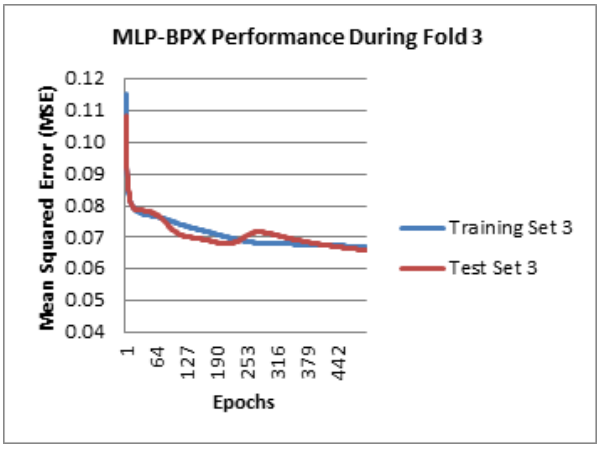

Figure 4.26 MLP-BPX Fold 3 on B-[Excl Missing]

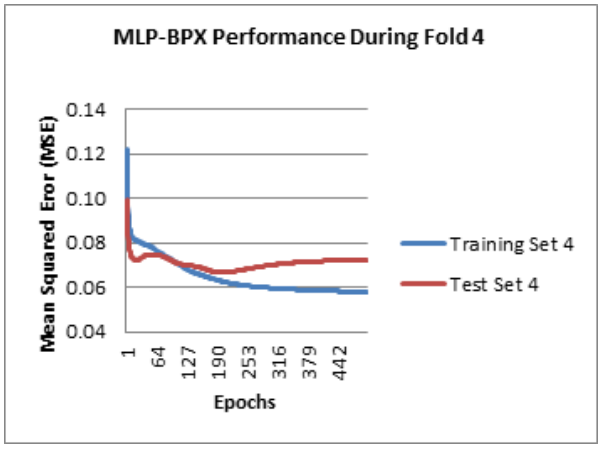

Figure 4.27 MLP-BPX Fold 4 on B-[Excl Missing]

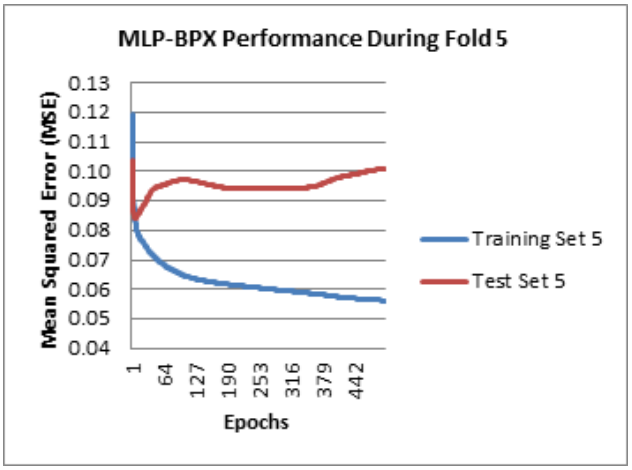

Figure 4.28 MLP-BPX Fold 5 on B-[Excl Missing]

In these charts one can observe that the Mean Squared Error (MSE) stops decreasing and starts increasing after a certain number of iterations. This is due to over-fitting on the test sets. It is important to note that the MLP-BPX network was programmed to only consider the best solution that was obtained before over-fitting commenced. Hence, the trends are only used to show the behaviour of the network on the test data without stopping the learning when over-fitting starts occurring. All the training trends show how the network consistently learned the data until the smallest error was reached. Figure 4.29, 4.30, 4.31, 4.32, and 4.33 give a cross validation training and test error trend of the network on dataset D-[Extracted Features] in which the network achieved the second highest classification accuracy.

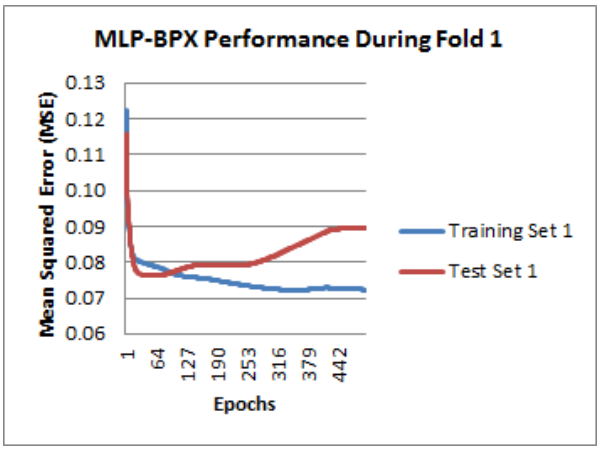

Figure 4.29 MLP-BPX Fold 1 on D-[Extracted Features]

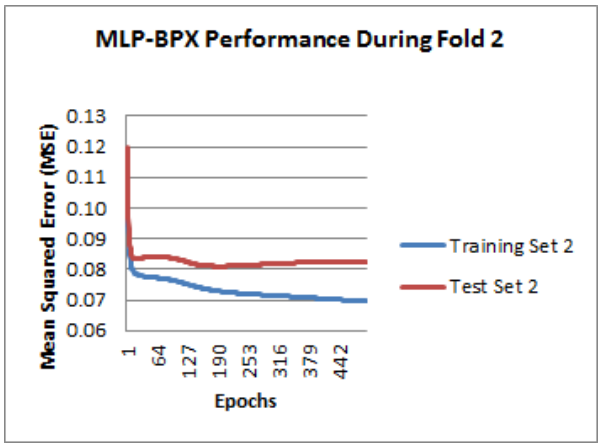

Figure 4.30 MLP-BPX Fold 2 on D-[Extracted Features]

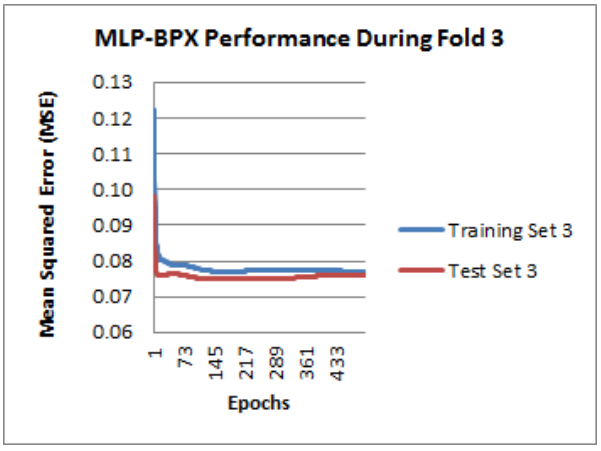

Figure 4.31 MLP-BPX Fold 3 on D-[Extracted Features]

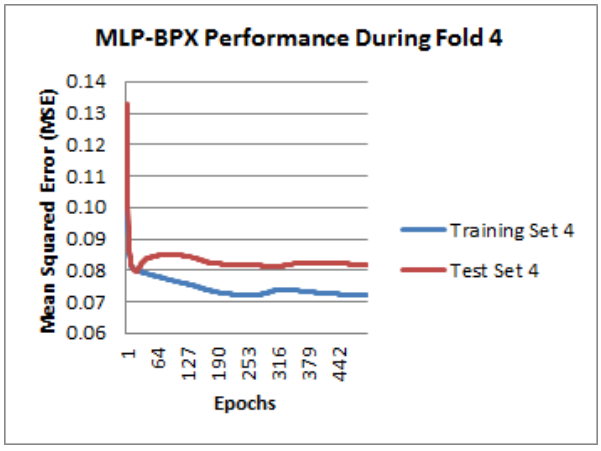

Figure 4.32 MLP-BPX Fold 4 on D-[Extracted Features]

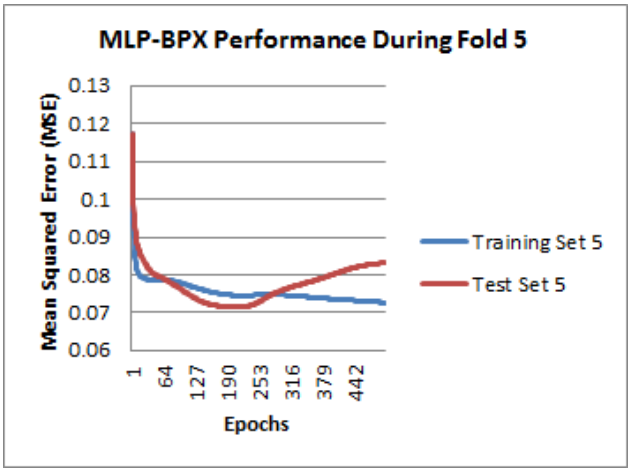

Figure 4.33 MLP-BPX Fold 5 on D-[Extracted Features]

Figure 4.34, 4.35, 4.36, 4.37, and 4.38 give a cross validation training and test error trend of the network on dataset C-[Replaced by Mean] in which the network achieved the third highest classification accuracy.

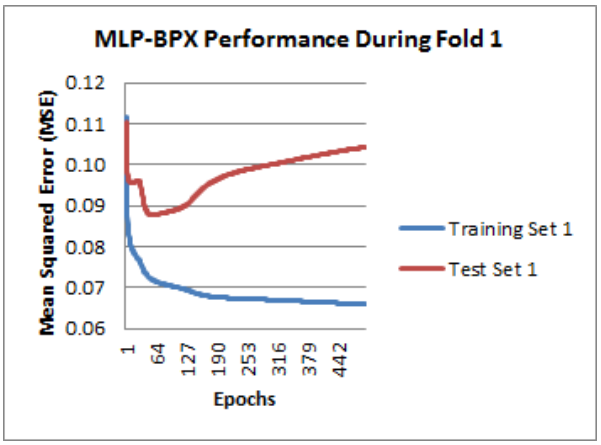

Figure 4.34 MLP-BPX Fold 1 on C-[Replaced by Mean]

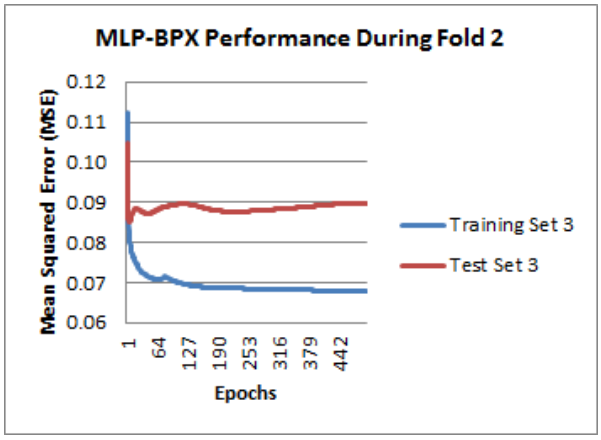

Figure 4.35 MLP-BPX Fold 2 on C-[Replaced by Mean]

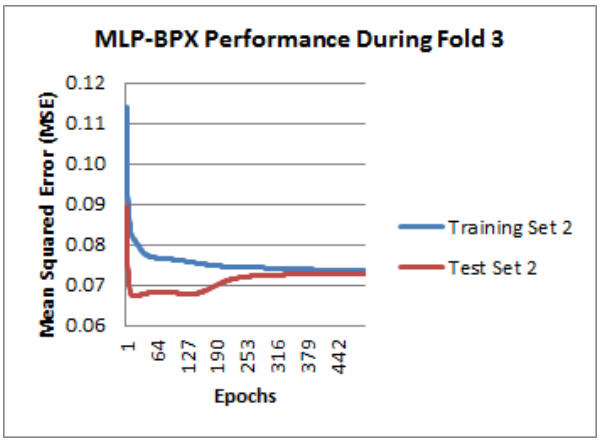

Figure 4.36 MLP-BPX Fold 3 on C-[Replaced by Mean]

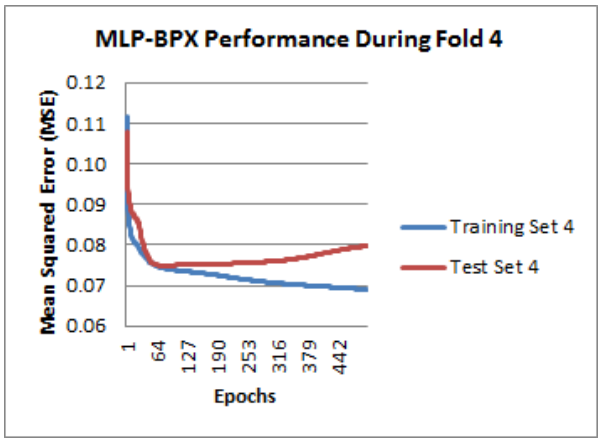

Figure 4.37 MLP-BPX Fold 4 on C-[Replaced by Mean]

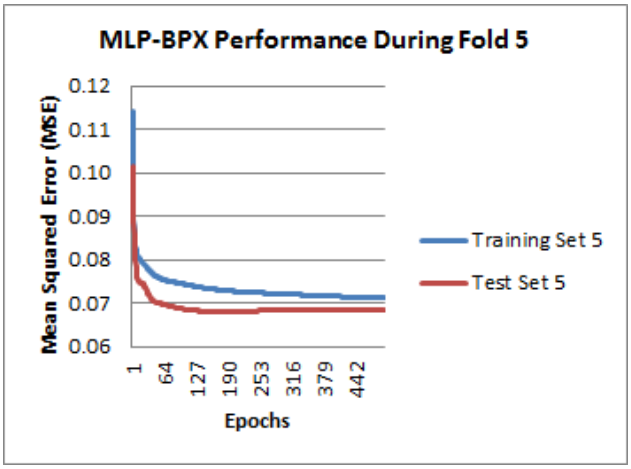

Figure 4.38 MLP-BPX Fold 5 on C-[Replaced by Mean]

Figure 4.39, 4.40, 4.41, 4.42, and 4.43 provide a cross validation training and test error trend of the network on dataset A-[Unprocessed] in which the network achieved the lowest classification accuracy. The network was trained using this dataset with a learning rate of 0.01 and 60 epochs. With this dataset the network was not able to converge using a high learning rate (0.67) that was used for other datasets. The learning rate and number of epochs were chosen through experimentation.

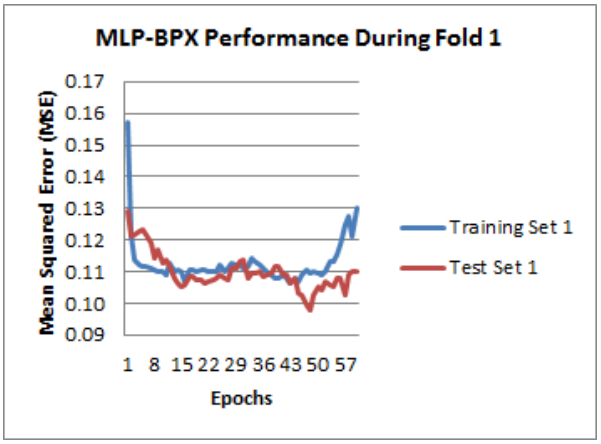

Figure 4.39 MLP-BPX Fold 1 on A-[Unprocessed]

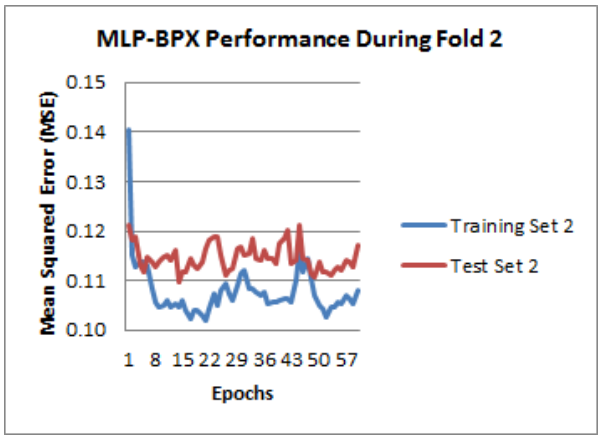

Figure 4.40 MLP-BPX Fold 32 on A-[Unprocessed]

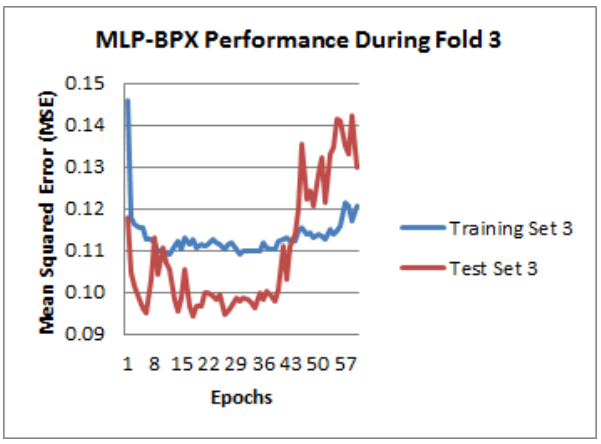

Figure 4.41 MLP-BPX Fold 3 on A-[Unprocessed]

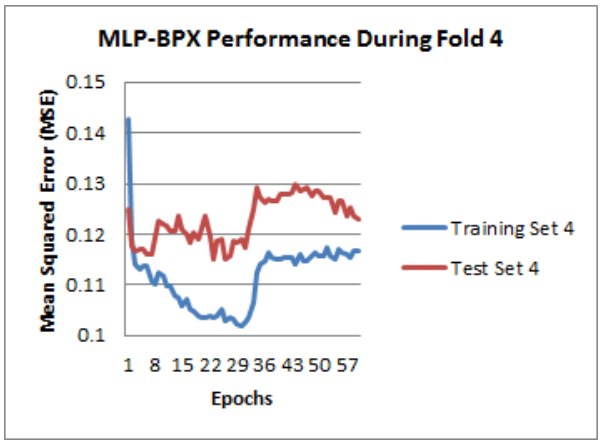

Figure 4.42 MLP-BPX Fold 4 on A-[Unprocessed]

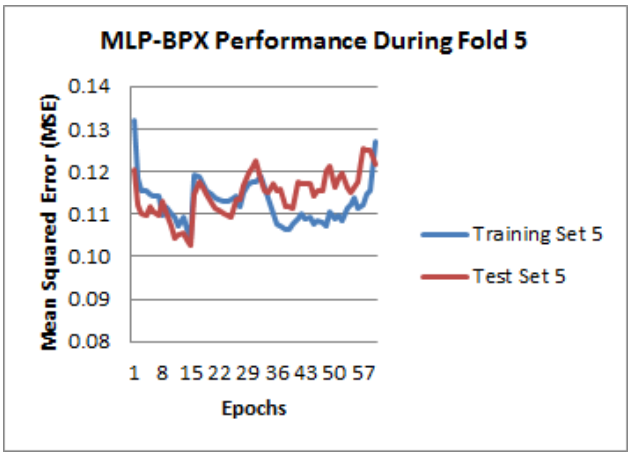

Figure 4.43 MLP-BPX Fold 5 on A-[Unprocessed]

The outcomes of testing the performance of MLP-BPX network using different configurations on dataset B-[Excl Missing] are given in figure 4.44 below.

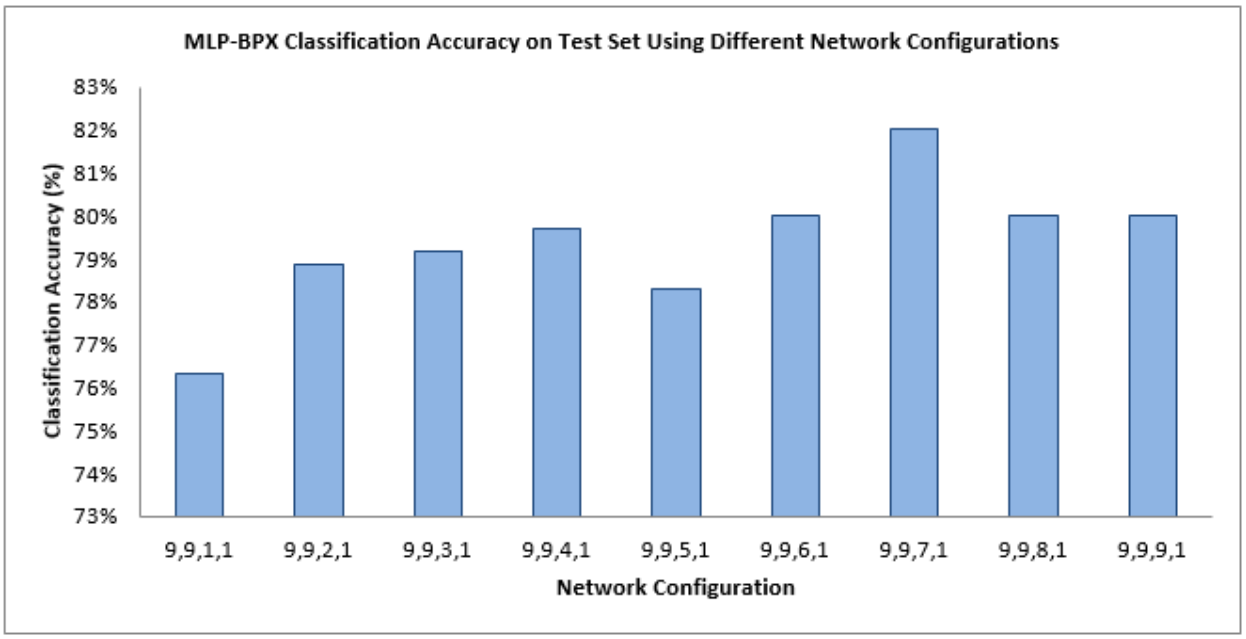

Figure 4.44 MLP-BPX classification accuracy using different configurations on dataset B-[Excl Missing].

In Figure 4.44, the network with configuration {9,9,7,1} has the best classification of 82% on the test set, while configurations {9,9,8,1} and {9,9,9,1} have the second best classification accuracy

of 80%. Configuration {9,9,1,1} has the lowest classification accuracy of 76%. The otherconfigurations have classification accuracies that range between 77% and 79%. Table 4.8 below provides detailed results on the performance of the network with these configurations. It is important to note that the lowest MSE errors reported in Table 4.8 are errors of the fold with the lowest error during 5-fold cross validation. Therefore it is possible for a network configuration with the best overall accuracy to have the lowest training or test error that is higher than the configuration with the lower overall accuracy. The last column in Table 4.8 gives the epoch number where the best solution was found on each fold/iteration of 5-fold cross validation during testing.

Network Configuration	Num of Epochs in CV	Time (Seconds)	Lowest Training Error	Lowest Test Error	Classification Accuracy	Best Epoch in Each Fold (1,..,5)
9,9,1,1	500	1.268	0.076	0.078	76.33%	2,119,31,9,296
9,9,2,1	500	1.958	0.066	0.075	78.87%	2,98,192,128,40
9,9,3,1	500	2.893	0.070	0.074	79.15%	14,463,19,92,60
9,9,4,1	500	3.485	0.069	0.072	79.71%	288,3,9,6,27
9,9,5,1	500	4.196	0.067	0.078	78.30%	106,7,5,109,7
9,9,6,1	500	4.783	0.064	0.068	80.00%	294,72,8,160,7
9,9,7,1	500	5.662	0.059	0.060	82.00%	33,12,470,173,4
9,9,8,1	500	6.369	0.064	0.068	80.00%	84,38,267,5,8
9,9,9,1	500	7.026	0.068	0.052	80.00%	72,55,60,151,8

Table 4.8 MLP-BPX performance using different configurations on dataset B-[Excl Missing].

The ROC curve for MLP-BPX Network is given in Figure 4.45. The AUC of the MLP-BPX ROC is 80.4%, this allows the network to be called a good classifier. The ROC curve was plotted by training and testing the network using dataset B-[Excl Missing], 500 training epochs and a learning rate of 0.67. Statistical analysis on the classification performance of MLP-BPX Network is given in Table 4.9. The level of significance was selected to be 0.05. This means that the null hypothesis is rejected if the computed p-value is less than 0.05. The dataset B-[Excl Missing] was used for the statistical analysis. The statistical results show that only the third model (3rd fold) did not fit the diabetes data well even though a classification accuracy of 85% was achieved by the model on the test data. This model did not classify a sufficient number of expected positive instances.

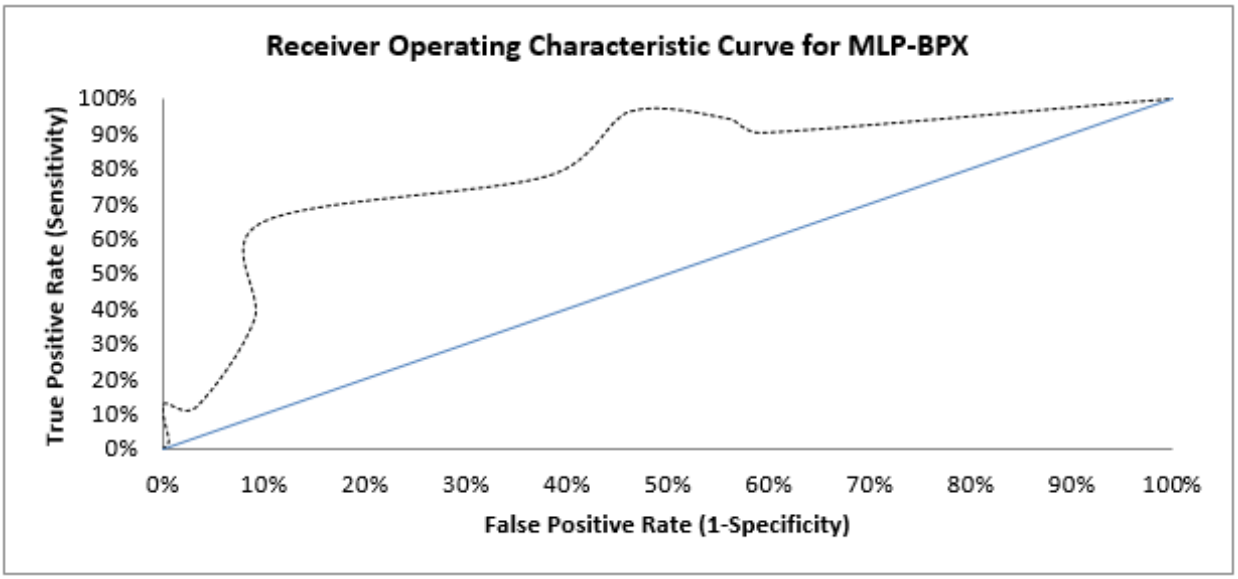

Figure 4.45 ROC Curve for MLP-BPX Network

The null hypothesis was accepted in all other four models during cross validation, meaning that they classified a significant number of expected diabetic and non-diabetic instances in the test set. Hence, the hull hypothesis was accepted.

Fold Number	Expected (Non Diabetic, Diabetic)	Observed (Non Diabetic, Diabetic)	Degrees of Freedom	Chi-square	p-value:	Result
Fold 1	46,25	49,22	1	0.56	0.46	Accept H_0
Fold 2	49,22	44,27	1	1.65	0.20	Accept H_0
Fold 3	43,28	53,18	1	5.32	0.02	Reject H_0
Fold 4	50,21	52,19	1	0.27	0.60	Accept H_0
Fold 5	45,26	44,27	1	0.06	0.81	Accept H_0

Table 4.9 Goodness-of-fit test for 5-fold CV MLP-BPX Network

Table 4.10 provides a comparison of the MLP-BPX with recent studies in literature which were outperformed by the MLP-BPX techniques in terms of classification accuracy.

Year	Reference	ML Technique	Accuracy	Learning Rate	Dataset
2017	**This study**	**MLP-BPX**	**82%**	**0.67**	**B-[Excl Missing]**
2016	Poonkuzhali et al. [98]	MLP-BP	80%	0.33	Pima Indian
2016	Karumuri et al. [99]	Mathlab MLP	80%	0.3	Pima Indian
2016	Jan et al. [100]	CV-MLP	75%	Not Reported	Pima Indian
2011	Pradhan et al. [101]	MLP-GA	72%	Not Reported	Pima Indian

Table 4.10 Performance comparison of different types of MLP Neural Networks

4.4 Statistical Techniques

On each iteration (restart) of the LR-n technique, the dataset was randomly partitioned into five samples of equal size, and four of these samples were used to create four LR models. Each sample was used to create a distinct LR model. To create a LR model, each sample was partitioned 60:40 into a training set and a validation set. The 5-th sample was used as the test sample for the model with the best classification accuracy on its validation set. The following parameters were used to train and test the LR-n technique on all datasets: number of restarts = 5, number of samples = 5, number of epochs for GSA algorithm = 300, and a learning rate = 0.15.

4.4.1 Improved Logistic Regression with $n-$restarts (LR-n)

The LR-n technique achieved the highest classification accuracy of 84% with 72% sensitivity and 100% specificity on dataset B-[Excl Missing]. The Logistic Regression equations for the model with best validation accuracy on each restart were obtained and are as follows:

$$E(y) = h_w(0.38 * PREG + 1.53 * GLUC - 0.31 * PRESS + 1 * SKIN + 1.98 * INSU + 0.41 * BMI + 1.49 * PDF + 1.36 * AGE) \quad (4.2)$$

$$E(y) = h_w(0.15 * PREG + 1.68 * GLUC - 0.41 * PRESS + 1.47 * SKIN + 1.2 * INSU - 0.66 * BMI - 0.06 * PDF + 1.91 * AGE) \tag{4.3}$$

$$E(y) = h_w(0.85 * PREG + 2.43 * GLUC - 0.47 * PRESS - 1.17 * SKIN + 1.67 * INSU - 0.61 * BMI + 1.53 * PDF + 1.1 * AGE) \tag{4.4}$$

$$E(y) = h_w(-0.2 - 3.38 * PREG + 0.58 * GLUC + 3.34 * PRESS - 0.47 * SKIN + 1.2 * INSU - 0.29 * BMI + 1.27 * PDF + 0.37 * AGE) \tag{4.5}$$

$$E(y) = h_w(-0.16 * PREG + 1.34 * GLUC + 0.58 * PRESS + 1.7 * SKIN + 1.05 * INSU + 1.14 * BMI + 0.65 * PDF + 1.32 * AGE) \tag{4.6}$$

$$E(y) = h_w(0.04 * PREG + 2.38 * GLUC - 1.2 * PRESS + 0.48 * SKIN - 0.24 * INSU + 0.12 * BMI + 0.64 * PDF + 2.22 * AGE) \tag{4.7}$$

The algorithm converts the output of h_w into crisp values 0 or 1 using the cut off value of 0.5 (i.e. if output of h_w is less than 0.5 then output = 0 else output = 1). The chart in Figure 4.46 shows the classification accuracy of each model that was picked to classify the final test set per restart. One can observe from the results in Figure 4.46 that LR-*n* achieved the lowest average classification accuracy of 64% on dataset A-[Unprocessed], followed by average accuracy of 80% on dataset C-[Replaced by Mean], and then the second highest average classification accuracy of 81% on dataset D-[Extracted Features]. All these results were obtained using similar parameters that were used to find the best classification accuracy of 84% using dataset B-[Excl Missing].

The proposed LR model achieved the lowest classification accuracy on the unprocessed dataset A-[Unprocessed] compared to all techniques proposed in this study. This suggests that the logistic model does not deal well with missing data or attribute values of varying scales to learn effectively. This is supported by the lowest sensitivity of 7% that was achieved by the model on the unprocessed dataset A-[Unprocessed]. Figure 4.47 provides performance results of the proposed LR-*n* model on different datasets. The ROC curve for LR-*n* technique is given in Figure 4.48. The AUC of the LR-*n* ROC curve is 77% which allows the LR-*n* technique to be regarded as a fair classifier. The

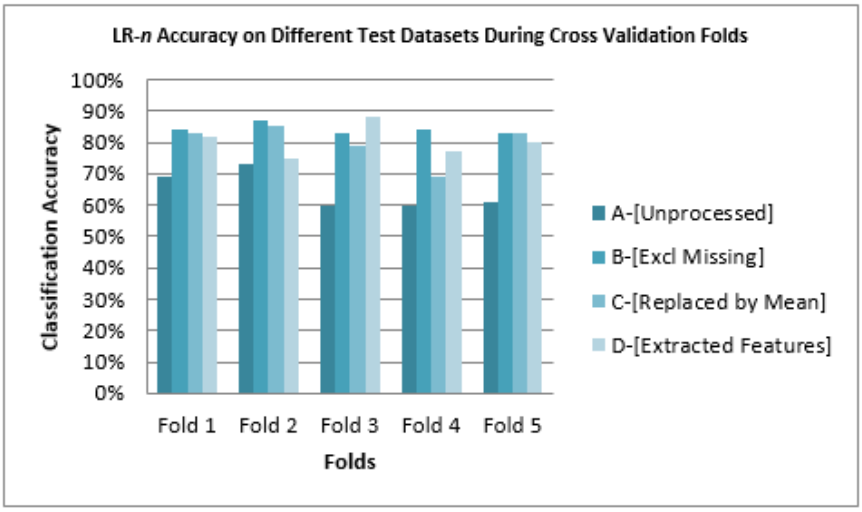

Figure 4.46 Performance of best LR models on final test sets on each restart

ROC curve was plotted by training and testing the technique with dataset B-[Excl Missing], using a learning rate of 0.15 and 300 epochs for SGA algorithm.

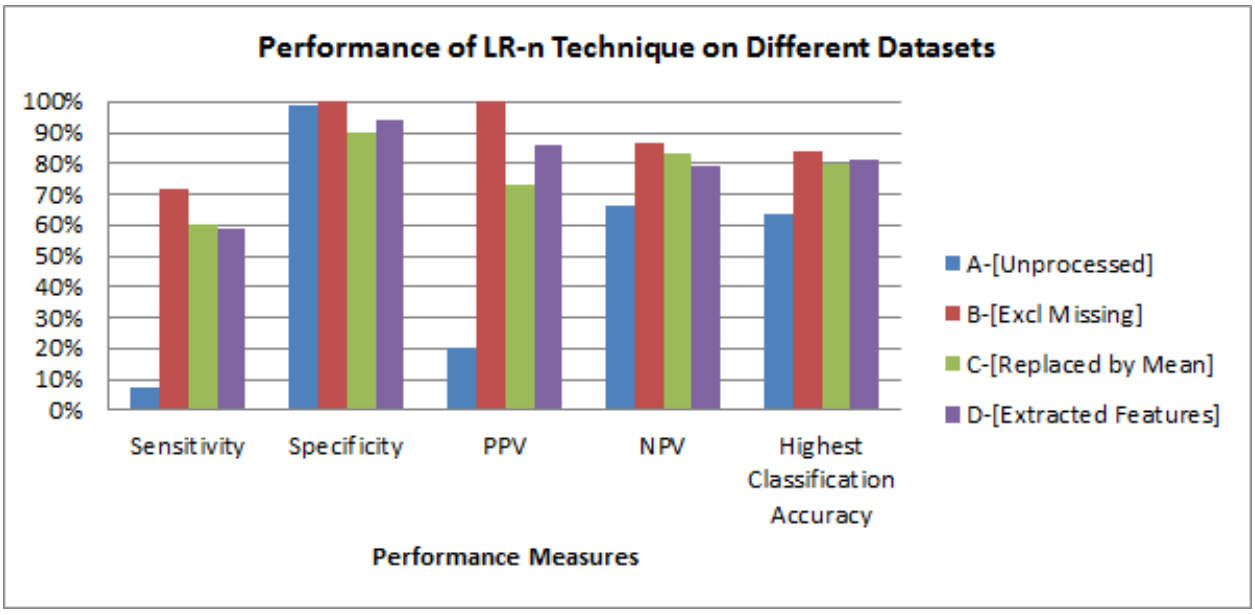

Figure 4.47 Performance measures for LR-*n* technique.

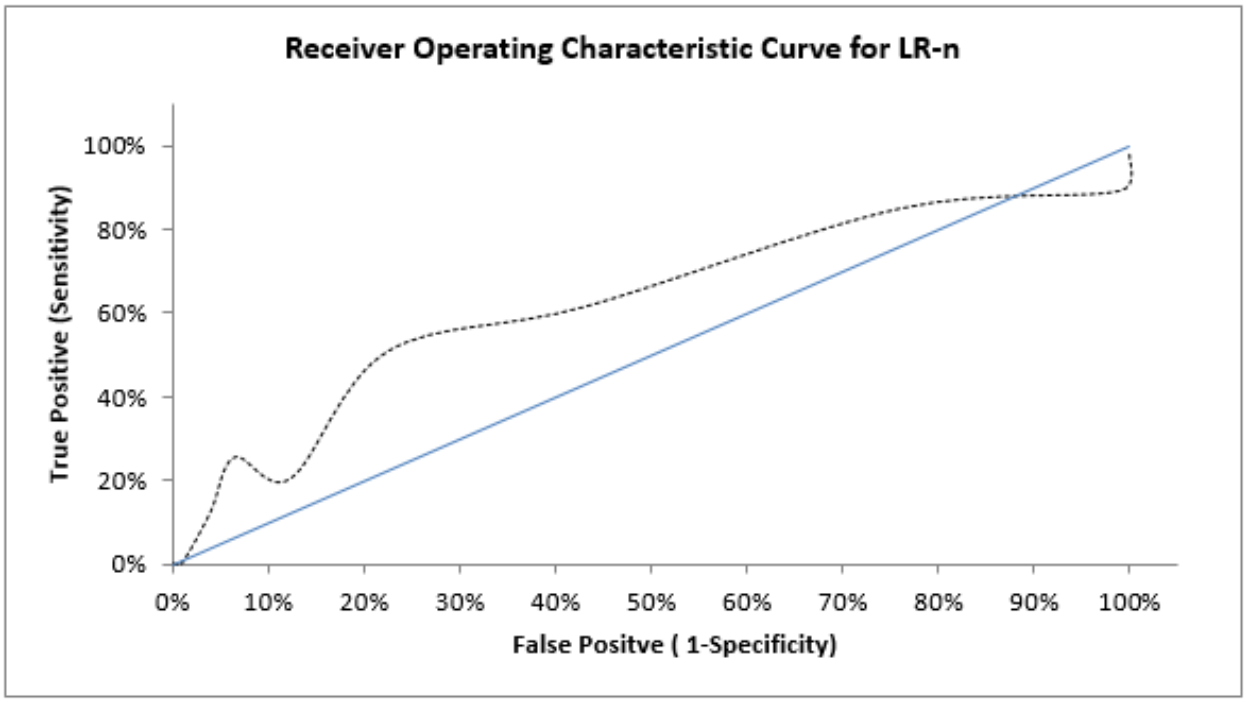

Figure 4.48 ROC Curve for LR-*n* model

The statistical analysis of the classification performance of LR-*n* technique on dataset t B-[Excl Missing] is given in Table 4.11. The level of significance was selected to be 0.05. According to Chi-Square results in Table 4.11, the fourth model (best model in the 4-th restart) did not fit the diabetes data well. This model did not classify a sufficient number of expected diabetic and non diabetic instances. The null hypothesis is accepted for the four models, meaning that they classified a significant number of expected diabetic and non-diabetic instances on the test set. Hence, the hull hypothesis was accepted.

The standard LR model was also trained and tested in this study. This standard model achieved the classification accuracy of less than 68% in all datasets utilized in this study. It is evident from the results that LR-*n* improved the accuracy of the standard LR model by approximately 17%. LR-*n* technique also performed better than other variants of Logistic Regression techniques that were proposed in literature. Table 4.12 provides a comparison of the LR-*n* technique with similar

Model	Expected (Non Diabetic, Diabetic)	Observed (Non Diabetic, Diabetic)	Degrees of Freedom	Chi-square	p-value:	Result
Model 1	20,8	20,8	1	0.00	1.00	Accept H_0
Model 2	23,5	22,6	1	0.24	0.62	Accept H_0
Model 3	19,9	21,7	1	0.66	0.42	Accept H_0
Model 4	23,5	18,10	1	6.09	0.01	Reject H_0
Model 5	18,10	17,11	1	0.16	0.69	Accept H_0

Table 4.11 Goodness-of-fit test for n models in LR-*n* algorithm

techniques in literature which were outperformed by the LR-*n* in terms of classification accuracy.

Year	Reference	ML Technique	Accuracy	Dataset
2017	**This study**	**LR-*n***	**84%**	**B-[Excl Missing]**
2015	Nongyao Nai-Aruna, et al. [87]	LR with Boosting (BT+LR)	82%	Sawanpracharak RH
2013	Bassam et al. [143]	LR	81%	Kuwait NHN
2016	Devi et al. [144]	LR	79%	AR Hospital - Madurai
2004	Cabrera et al [90]	Carabera's LR (CLR)	78%	Pima Indian

Table 4.12 Comparison of LR variants

4.5 Summary of Results

Table 4.13 provides an overall summary of all the results obtained in this study. Figure 4.48, 4.49, 4.50, 4.51 show the performance comparison of these techniques in graphical form.

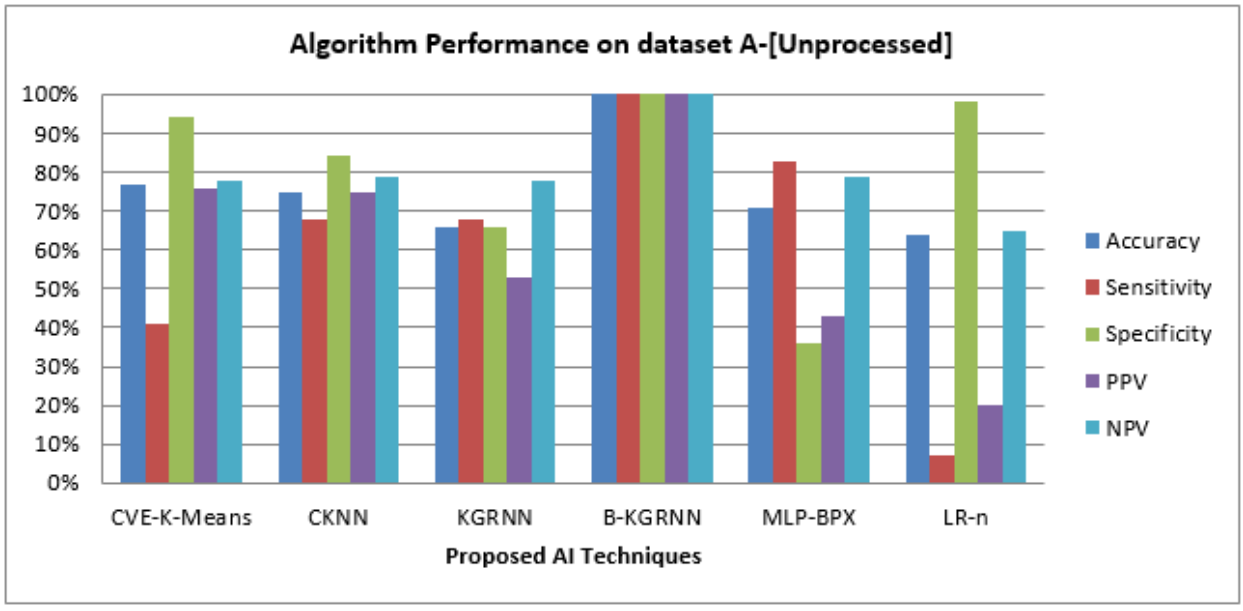

Figure 4.49 Performance Comparison of proposed ML techniques on dataset A-[Unprocessed]

Dataset	ML technique	Accuracy	Sensitivity	Specificity	PPV	NPV	Area Under the ROC Curve (AUC)
A-[Unprocessed]	CVE-K-Means	77%	41%	94%	76%	78%	-
	CKNN	75%	68%	84%	75%	79%	-
	KGRNN	66%	68%	66%	53%	78%	-
	B-KGRNN	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
	MLP-BPX	71%	83%	36%	43%	79%	-
	LR-*n*	64%	7%	98%	20%	65%	-
B-[Excl Missing]	CVE-K-Means	80%	61%	90%	60%	90%	-
	CKNN	81%	**89%**	88%	73%	**96%**	-
	KGRNN	**85%**	67%	92%	75%	89%	-
	MLP-BPX	82%	72%	73%	57%	83%	**80%**
	LR-*n*	84%	72%	**100%**	**100%**	87%	77%
C-[Replaced by Mean]	CVE-K-Means	78%	59%	88%	70%	81%	-
	CKNN	78%	56%	**98%**	**93%**	82%	-
	KGRNN	71%	65%	76%	65%	76%	-
	MLP-BPX	79%	61%	88%	68%	**84%**	-
	LR-*n*	**80%**	**72%**	73%	73	83	-
D-[Extracted Features]	CVE-K-Means	79%	67%	86%	71%	73%	-
	CKNN	80%	71%	90%	83%	83%	-
	KGRNN	**86%**	**83%**	87%	77%	**91%**	87%
	MLP-BPX	81%	70%	81%	66%	84%	-
	LR-*n*	81%	59%	**95%**	**85%**	79%	-

Table 4.13 Summary of results obtained by ML techniques proposed in this study

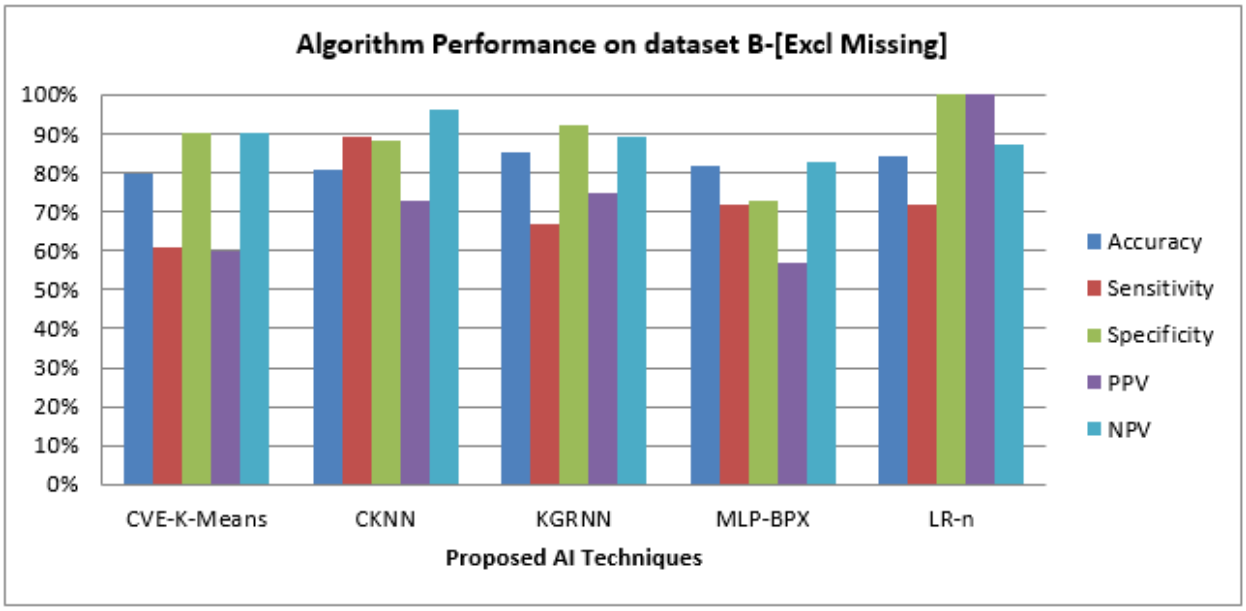

Figure 4.50 Performance Comparison of proposed ML techniques on dataset B-[Excl Missing]

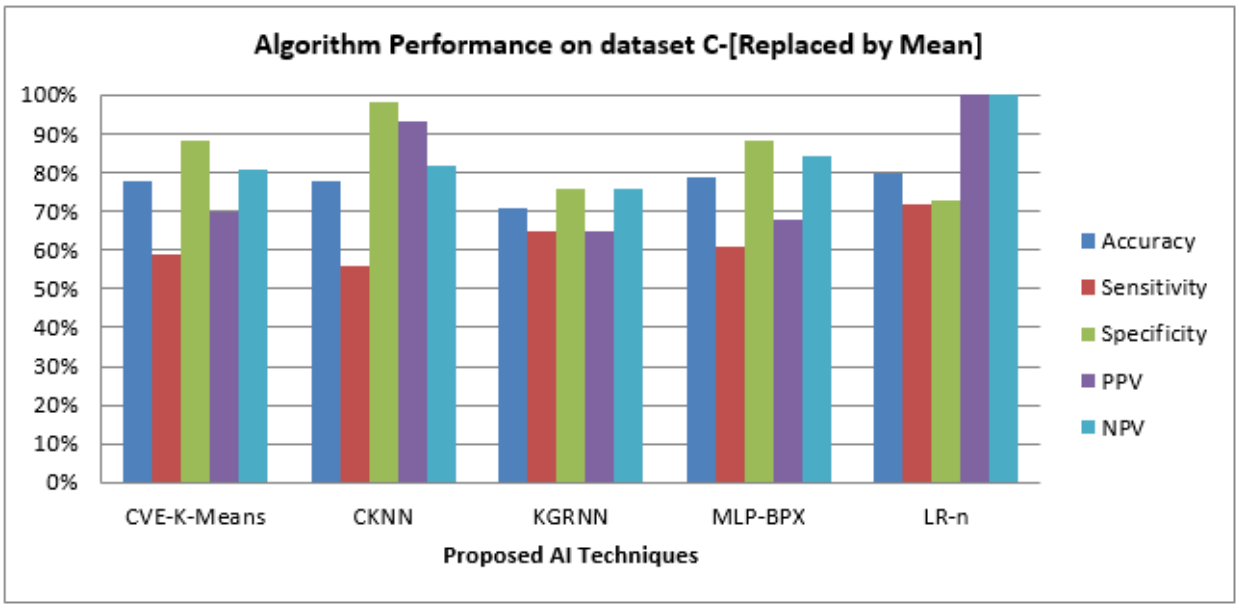

Figure 4.51 Performance Comparison of proposed ML techniques on dataset C-[Replaced by Mean]

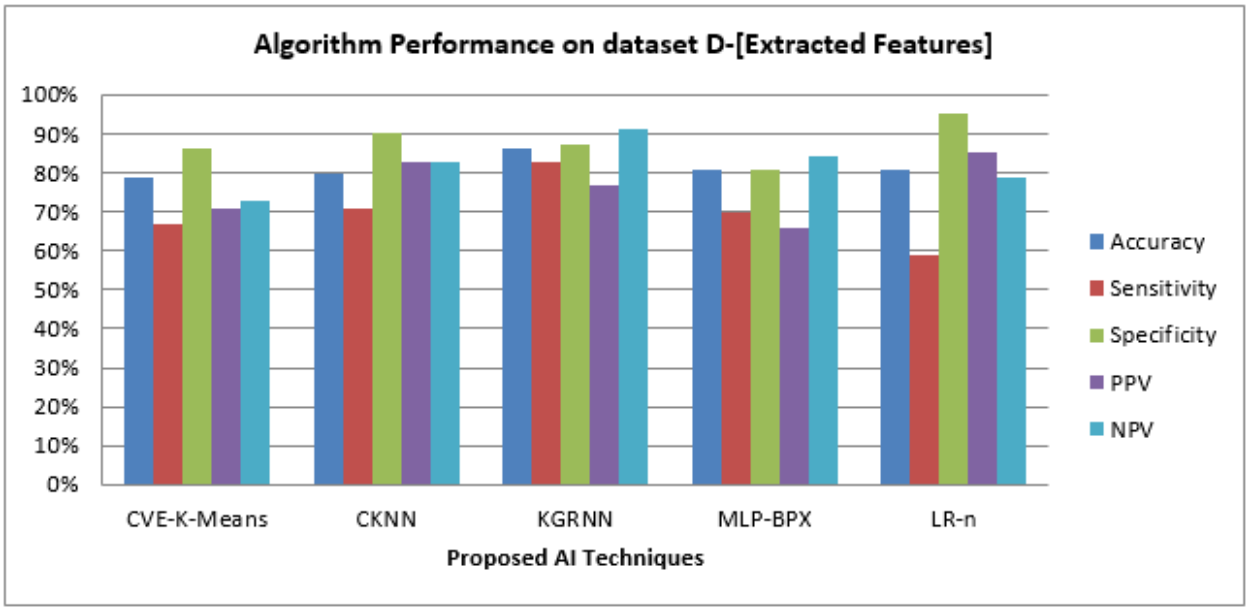

Figure 4.52 Performance Comparison of proposed ML techniques on dataset D-[Extracted Features]

An overview of recent results in literature and the results obtained in this study is provided in Table 4.14.

Algorithm Type	Algorithm	Year	Reference	Accuracy	Learning Rate	Smoothing Factor σ	Num of Neighbours (k)	Clusters
Artificial Neural Networks	**B-KGRNN**	**2017**	**This study**	**100%**	**N/A**	**0.2**	**N/A**	**18**
	KGRNN	**2017**	**This study**	**86%**	**N/A**	**0.19**	**N/A**	**N/A**
	BT+ANN	2015	Nongyao et al. [87]	85%	N/A	N/A	N/A	N/A
	CNN	2016	S. Tharani et al. [83]	83%	N/A	N/A	N/A	N/A
	MLP-BPX	**2017**	**This study**	**82%**	**0.67**	**N/A**	**N/A**	**N/A**
	PNN	2016	Zahed Soltani [82]	81%	N/A	N/A	N/A	N/A
	GRNN	2003	Kayaer et al. [79]	80%	N/A	2.5	N/A	49
	[98] MLP-BP	2016	Poonkuzhali et al. [98]	80%	0.33	N/A	N/A	N/A
	Mathlab MLP	2016	Karumuri et al. [99]	80%	0.30	N/A	N/A	N/A
	GA-MLPNN	2016	Kumar Choubey [84]	79%	N/A	N/A	N/A	N/A
	CV-MLP	2016	Jan et al. [100]	75%	Not Reported	N/A	N/A	N/A
	MLP-GA	2011	Pradhan et al. [101	72%	Not Reported	N/A	N/A	N/A
	RBFN	2017	Cheruku et al. [81]	70%	N/A	N/A	N/A	N/A
	VGRNN	2014	V. Sujatha [135]	57%	N/A	Not Reported	N/A	N/A
Instance-based Techniques	**CVE-K-Means**	**2017**	**This study**	**80%**	**N/A**	**N/A**	**N/A**	**12**
	K-Means Clustering	**2017**	**This study**	**77%**	**N/A**	**N/A**	**N/A**	**12**
	[22] K-Means	2014	Vijayan et al. [22]	77%	N/A	N/A	N/A	Not Reported
	[75] Novel K-Means	2013	Shadi et al. [75]	69%	N/A	N/A	N/A	2
	[74] K-Means	2012	Nidhi et al. [74]	52%	N/A	N/A	N/A	3
	CKNN	**2017**	**This study**	**81%**	**N/A**	**N/A**	**3**	**N/A**
	Amalgam KNN	2014	Vijayan et al. [22]	80%	N/A	N/A	Not Reported	N/A
	Class-wise KNN	2013	Christobel et al. [72]	78%	N/A	N/A	10	N/A
	KNN-Naive-Bayes	2009	L. Jiang et al. [73]	76%	N/A	N/A	10	N/A
	[8] KNN	2014	K. Saxena et al. [8]	75%	N/A	N/A	5	N/A
	kNNModel	2003	Gongde Guo et al. [23]	75%	N/A	N/A	> 5	N/A
Statistical Techniques	**LR-*n***	**2017**	**This study**	**84%**	**0.15**	**N/A**	**N/A**	**N/A**
	LR with Boosting	2015	Nongyao et al. [87]	82%	Not Reported	N/A	N/A	N/A
	Carabera LR (CLR)	2014	Cabrera et al. [90]	78%	Not Reported	N/A	N/A	N/A
	Meng LR (MLR)	2013	Meng et al. [91]	77%	Not Reported	N/A	N/A	N/A

Table 4.14 Comparison of best results with previous studies

Chapter 5

Discussion

The following Machine Learning techniques were applied: the standard K-Means Clustering Algorithm, the CVE-K-Means Clustering Algorithm, a Hybrid of the CVE-K-Means and the KNN Algorithm (CKNN), a Hybrid of CVE-K-Means and GRNN (KGRNN), an Ensemble of CVE-K-Means, Boosting, and GRNN(B-KGRNN), an Improved Multi-Layer Back-Propagation Neural Network (MLP-BPX), and an Improved Logistic Regression with n-restarts (LR-n). The performance results of these techniques and their implications are discussed. The discussion on the impacts of different data preprocessing techniques used and the limitations of classification techniques proposed in this study is then provided. A comparative analysis of the performance between the Machine Learning techniques proposed in this study and those in literature is also provided and discussed in this chapter.

5.1 Data Preprocessing

The Pima Indian diabetes dataset has 65% positive instances and 35% negative instances. This means that the dataset is slightly imbalanced. This dataset has an average feature completeness rate of 90%. Feature completeness rate is the number of instances with a value for the feature divided

by the total number of instances in the dataset. Out of the 9 features in the dataset, 4 of them are 100% complete, meaning that 44% of features in the dataset are complete. These features are PREG, PDF, AGE, and Class Variable. Two out of 9 features (22% of the total) are 99% complete. These features are GLU and BMI. PRESS is 95% complete, SKIN is 70% complete, and INS is 51% complete. The minimal imbalance in the dataset and the high number of incomplete features motivated this study to apply feature extraction techniques and preprocess the dataset to produce other additional datasets of better quality. On average, the proposed ML techniques obtained results which were 12% higher on the derived datasets compared to the original un-preprocessed dataset. This clearly shows that the issue of incomplete feature values and the minimal class imbalance in the dataset was adequately addressed by preprocessing the data and applying feature extraction techniques.

The removal of missing values provided the best results compared to other data preprocessing techniques in this study. This approach ensured that all instances which caused noise, variance and bias in the dataset were removed. Feature extraction followed by data imputation also provided an improvement on the performance of ML techniques. The advantage of imputing the data is that all instances in the dataset are not removed but their quality is improved by substituting erroneous attribute values which cause noise in the data with better values. The advantage of using feature selection was the exclusion of redundant features which made the learning of ML technique slow and inefficient. The balance of feature importance between features whose values are directly influenced by genetics and those which are not was observed. As shown in the feature extraction results in Table 4.2, the combined feature importance percentage for features that are influenced by genetics is 51% (PDF+MMI+PRESS+SKIN) while for those which are not influenced by genetics is 49% (GLU+AGE+PREG+INS = 49%). These results imply that although hereditary characteristics have a significant influence in the development of diabetes, other factors are as equally important as diabetes risk factors.

Although the ML techniques performed well on the derived datasets, the preprocessing techniques that were used to derive them also have limitations. Removing missing values has limitations when the original dataset is very small since it causes a reduction in the total number of instances in the dataset. Similarly for feature extraction, if the dataset has a small number of features then it becomes difficult to obtain exceptional results using a very small number of features. The downside of imputing data is that instances with erroneous attribute values are not removed in the dataset and they still pose of a risk of negatively impacting the performance of ML techniques if the imputation technique was not effective.

5.2 Comparison of Techniques

The algorithm performance evaluation measures such as the classification accuracy, sensitivity, specificity, and ROC were used in this study. These performance measures were used to determine the efficacy of the proposed ML techniques in type II diabetes diagnosis and to compare their performance with each other including other ML techniques proposed in literature. Sensitivity and specificity were used to assess the effectiveness of the classifiers in identifying diabetic and non diabetic instances since the classification accuracy only reveals the number of instances that were correctly classified from all test instances. The ML techniques employed in this study obtained competitive results on the derived datasets, and a majority of them did not perform well on the un-preprocessed dataset. Out of six novel ML techniques proposed in this study, three of them obtained results which were superior to the best results known in literature that were obtained by similar techniques. The proposed ML techniques were superior in terms of classification accuracy and computational time complexity. The proposed ML techniques were applied to each of the following datasets: A-[Unprocessed], B-[Excl Missing], C-[Replaced by Mean], and D-[Extracted Features].

The dataset A-[Unprocessed] is the raw Pima Indian dataset. This dataset had seven hundred and

sixty eight (768) instances, of which 65% of them were not diabetec and 35% were diabetic. Each instance in this dataset had nine attributes. The ML techniques in this study were applied on this dataset without preprocessing it. The best classification accuracy of 100% was obtained by the B-KGRNN technique on this dataset. This was followed by the CVE-K-Means, CKNN, MLP-BPX, KGRNN, and LR-*n* technique which individually obtained the highest classification accuracy of 77%, 75%, 71%, 66%, and 64% respectively. Each of these techniques had the sensitivity of 41%, 68%, 83%, 68%, and 7% respectively. An ensemble of CVE-K-Means, Adaboost, and the KGRNN technique significantly improved the performance of the KGRNN technique on this dataset. The ensemble improved the classification accuracy and sensitivity of the KGRNN technique on dataset A-[Unprocessed] by 34% and 32% respectively. This shows that the B-KGRNN technique was more effective in dealing with noise, bias, and varying attribute value scales in the dataset. Each weak learner in the B-KGRNN technique effectively encoded all training instances which could not be encoded by other learners, thus, allowing the final classifier which is a combination of weak KGRNN techniques to be able to generalize well on the test data.

Although the B-KGRNN technique performed well on dataset A-[Unprocessed] than other techniques, the enhanced Instance-based techniques outperformed the the other ANNs and the Statistical technique. This shows that the enhancements applied on Instance-based techniques enabled them to deal better with noise and variance in the data since these techniques are known to be prone to noise and bias in the data. The LR-*n* technique obtained the lowest sensitivity and classification accuracy on dataset A-[Unprocessed] compared to other techniques on the same dataset. The LR-*n* technique obtained the sensitivity of 7% and the classification accuracy of 64%. The missing data and the presence of slight class imbalance in the dataset inhibited the technique from learning the training data well to be effective in generalizing well on unseen instances. The sensitivity results show that the noise and the presence of slight class imbalance in the dataset favored the learning of a large number of negative instances than positive instances by the LR-*n* technique. Notwith-

standing the poor performance of the majority of ML techniques on dataset A-[Unprocessed], the advantage of using the original dataset without preprocessing it is that it allows the use of all the instances and their features in a dataset without the loss of any essential information which can aid the ML techniques to learn better. This however, comes at a high cost because the disadvantages of using a dataset without preprocessing it outweigh the advantages. Often, raw data is prone to noise and bias due to missing values and class imbalance which significantly affect the efficacy of ML techniques in a negative way.

The dataset B-[Excl Missing] was formed as a result of removing instances with missing values in the Pima Indian dataset. This dataset had three hundred and fifty-nine (359) instances of which 55% of them were not diabetec and 45% were diabetic. Each instance in this dataset had nine attributes. The best classification accuracy of 85% with 67% sensitivity on this dataset was obtained by the KGNN technique compared to other techniques. The CKNN technique obtained the best sensitivity of 89% with 81% classification accuracy on this dataset compared to other techniques. The CVE-K-Means technique obtained the lowest classification accuracy of 80% and 67% sensitivity on this dataset. The results showed that excluding the missing data from the Pima Indian dataset improved the classification accuracy of all the ML techniques employed in this study. The classification accuracy of the LR-*n*, KGRNN, MLP-BPX, CKNN, CVE-K-Means, and the technique improved by 20%, 19%, 11%, 6%, and 3% respectively. The sensitivity for the LR-*n*, CKNN, and the CVE-K-Means technique improved by 65%, 21%, 20%, and respectively. On the other hand, the sensitivity of the KGRNN and the MLP-BPX technique decreased by 1% and 11% respectively.

The removal of the missing data provided a significant increase in accuracy and sensitivity of the LR-*n* technique. This suggested that the LR-*n* technique was only effective when the dataset was cleansed of noise and the slight class imbalance was reduced. Although the KGRNN and the MLP-BPX technique also had the highest improvement in terms of classification accuracy, their

ability to diagnose more diabetic cases slightly decreased. This indicates that the KGRNN and the MLP-BPX were not very effective in classifying a large number of positive instances also in the absence of noise and class imbalance in the data. Instead, the removal of missing data made the KGRNN and the MLP-BPX technique to be more effective in classifying negative instances. The exclusion of missing values also allowed the CVE-K-Means technique to be more effective in classifying positive instances. By considering both the classification accuracy and sensitivity, one can conclude that the CKNN technique performed better than other techniques in diagnosing type II diabetes on the dataset B-[Excl Missing].

The dataset C-[Replaced by Mean] was formed by imputing missing values in the Pima Indian dataset with their respective attribute mean values. The class variable and number of times pregnant (PREG) attribute values were not imputed since missing values were represented by zeros in the data set. The zeros were permissible in these two attributes. The dataset C-[Replaced by Mean] had an equal number of instances as dataset A-[Unprocessed]. The ratio of diabetic and non diabetic instances was also the same. The LR-*n* technique obtained the best classification accuracy of 80% and 72% sensitivity on dataset C-[Replaced by Mean] compared to other techniques. Although most techniques obtained an improved classification accuracy on dataset C-[Replaced by Mean] compared to dataset A-[Unprocessed], their sensitivity significantly decreased on dataset C-[Replaced by Mean]. The sensitivity of the MLP-BPX, CKNN, and the KGRNN technique decreased by 22%, 12%, and 3% respectively. The CVE-K-Means had 18% increase in sensitivity on dataset C-[Replaced by Mean]. The classification accuracy of the LR-*n*, MLP-BPX, KGRNN, CKNN, and the CVE-K-Means technique improved by 20%, 8%, 5%, 5%, and 1% respectively on dataset C-[Replaced by Mean].

The improvement in the classification accuracy of the KGRNN, CKNN and the MLP-BPX technique were mainly due to many negatives instances which were correctly classified. The imputation of missing values with the mean value introduced a new kind of noise which made these techniques

to be more sensitive to the slight class imbalance in the dataset. Therefore, these techniques were more effective in encoding negative instances, which consequently improved their classification accuracy. If the improvement in classification accuracy by the KGRNN, CKNN and the LMP-BPX technique is considered individually, one would conclude that these techniques diagnosed diabetes better on dataset C-[Replaced by Mean] than on dataset A-[Unprocessed]. However, when the classification accuracy is observed in conjunction with sensitivity, it can be noticed that these techniques were less effective in identifying more positive cases on datased C-[Replaced by Mean] than on dataset A-[Unprocessed]. Therefore, evaluating the performance of ML techniques using both metrics leads to more reliable results. The restarts in LR-*n* technique made it to deal well with the new type of noise that was introduced by missing value imputation in the dataset, thus, the technique obtained the best results on datased C-[Replaced by Mean] compared to other techniques.

The dataset D-[Extracted Features] was formed as a result of excluding insignificant features in the Pima Indian dataset using the AHP and the Random Forest algorithm feature extraction technique. This dataset had three hundred and fifty-nine instances (359) of which 55% of them were not diabetec and 45% were diabetic. Each instance in this dataset had five attributes which were selected as significant attributes by the two feature extraction techniques. These attributes were: GLU, PDF, AGE, PREG, and SKIN. The KGRNN technique obtained the best classification accuracy of 86% and 83% sensitivity which were superior to other techniques on this dataset. Feature extraction provided an improvement in the classification accuracy and sensitivity for four out of the five ML techniques. The classification accuracy of the KGRNN, LR-*n*, MLP-BPX, CKNN, and the CVE-K-Means technique improved by 20%, 17%, 10%, 5%, and 2% respectively on dataset D-[Extracted Features]. Their sensitivity of the CKNN, CVE-K-Means, LR-*n*, KGRNN, improved by 42%, 26%, 18%, and 15% respectively. The sensitivity of the MLP-BPX technique was reduced by 13%. The exclusion of insignificant features allowed the KGRNN technique to

effectively learn to classify both negative and positive instances on dataset D-[Extracted Features] unlike in dataset B-[Excl Missing] where it could only encode a majority of negative instances. The combination of both feature extraction and the removal of missing values reduced noise in the dataset and the slight class imbalance, thus, it led to faster learning and better classification. The KGRNN technique performed better on dataset D-[Extracted Features] because it used centroids which were produced from a clean training set which represented the entire dataset well. Hence, it outperformed other ML techniques on this dataset.

The CVE-K-Means, KGRNN and the CKNN technique showed an improvement in computational time during training and testing compared to their respective standard ML techniques. The results in Figure 4.10 showed that the combination of k-fold cross validation and intelligently selecting initial cluster centres provided an improvement in the classification accuracy of the standard K-Means algorithm since the CVE-K-Means algorithm consistently obtained better results than the standard K-Means algorithm for different values of k on the same dataset. Furthermore, the results in Figure 4.11 suggested that there was no correlation between the execution time of the two techniques and their classification accuracy, however, the results showed that the improvements made on the standard K-Means algorithm had a positive impact when it comes to improving the execution time of the algorithm and in obtaining better results. The use of cluster centroids that were intelligently selected using CVE-K-Means algorithm did not show any improvement in the classification accuracy of the CKNN algorithm compared to the standard KNN algorithm. Both techniques obtained similar results across all the datasets except in the computational time during the classification stage. The CKNN technique showed an improvement in computational time complexity during the classification stage since only a small number of instances representing a whole training set were used for training. This is supported by the results in Figure 4.14 which showed that for the CKNN technique, the time to classify test instances was directly proportional to the number of instances used as the training set. The centroids produced by the CVE-K-Means

to train the KGRNN technique ensured that the network obtained improved results in less computational time. When all instances in the training set were used instead of the centroids to train the GRNN, the computation of the Gaussian activation function for every instance in the training set increased the computational time of the network.

The ROC curve area was used to illustrate the classification effectiveness of three classifiers (MLP-BPX, KGRNN, and LR-*n*) as their discrimination threshold varied during classification since they classified instances based on a cut-off value of their output. The ROC results of the KGRNN, MLP-BPX and LR-*n* model showed that these classifiers are suitable to be used for the diagnosis of type II diabetes. Apart from the B-KGRNN which had the best classification accuracy, sensitivity and specificity compared to other techniques. When the results of ML technique on all datasets were considered, the KGRNN technique obtained the best accuracy compared to other techniques. The CKNN technique obtained the best sensitivity, while the LR-*n* technique obtained the best specificity compared to other techniques. The classification accuracy and sensitivity were treated as equally important in evaluating the efficacy of the techniques proposed in this study unlike in many studies in literature which only reported on the classification accuracy of their techniques to demonstrate the efficacy of their techniques. The ability of the classifier to effectively identify positive instances is equally important as its classification accuracy in the diabetes diagnosis problem. In medical diagnosis problems it is important to have a classifier which can identify positive cases exceptionally well because false positive results lead to further tests which ultimately lead to a correct diagnosis, while false negative results lead to failure to treat a disease.

The results have shown that three out of the six Machine Learning techniques proposed in this study (CVE-K-Means, KGRNN, B-KGRNN) obtained results which were superior to the best known results in literature by their similar counterparts. The ROC results in conjunction with the classification accuracy and sensitivity showed that the Artificial Neural Networks are a better choice for type II diabetes diagnosis compared to the Statistical technique and the Instance-based

techniques. The ANNs have shown good performance on un preprocessed dataset when they were hybridized with other ML techniques such as Adaboost whichis specifically designed to deal with noise and class imbalance in a dataset. The ANNs also showed good performance when the combination of missing value and feature extraction were used. Instance-based techniques showed the best performance only when missing values were removed in the dataset. While statistical techniques showed the best performance when missing values were removed and when missing values were imputed with attribute mean in the dataset. This shows that there is no data preprocessing technique which optimally works for all ML techniques. Therefore, the selection of a data preprocessing technique to employ should be carefully decided by also considering the strengths and weaknesses of a ML technique to be applied. The experimental results have shown that the use of various data preprocessing techniques and the hybridization of individually enhanced Machine Learning techniques provided improved results compared to previous studies in literature.

5.3 Method Limitations

Although the proposed ML techniques performed well in diagnosing diabetes using various datasets, they still had weaknesses which were observed during the study. For example, the KGRNN technique performed well on the derived datasets, however, it struggled to obtain good results on the dataset A-[Unprocessed] and dataset C-[Replaced by Mean]. This implies that the network was sensitive to noise and varying attribute scales in the data. The use of Xavier weight initialization technique did not provide any significant benefit to the classification accuracy of the MLP-BPX network, except that the technique enabled the network to converge slightly faster even for smaller values of the learning rate which are known to slow the convergences of network. The classification accuracy of the MLP-BPX network was similar to that of the MLP-BP network with the same configuration and weights initialized using random values between -1 and 1. The lack of impact in the classification accuracy of the MLP-BPX network by the Xavier technique may be attributed

to the configuration of the network since the Xavier technique is known to be more effective when the network has many hidden layers. LR-*n* performed poorly on the A-[Unprocessed] by achieving the lowest sensitivity of 7% compared to all techniques proposed in this study. This implies that LR-*n* could not deal well with the missing data, slight class imbalance and varying scales in the attribute values in the data. Therefore, it is recommended for the datasets to be extensively pre-processed before the logistic regression technique is applied on them. Instance-based techniques were not effective in identifying positive cases, especially on the un-preprocessed dataset in which they outperformed other techniques in terms of classification accuracy. These techniques achieved a higher classification accuracy by classifying a high number of negative cases compared to other techniques. This shows that these techniques were susceptible to class imbalance in the dataset since the datasets had more negative instances than positive instances.

5.4 Comparison with Previous Studies

A performance comparison of the proposed ML techniques with previous studies in literature is outlined in Table 5.1. The classification accuracy, sensitivity, specificity, and the ROC area were used for comparison.

Algorithm Type	Algorithm	Year	Reference	Accuracy	Sensitivity	Specificity	ROC Area	Dataset
Artificial Neural Networks	**B-KGRNN**	**2017**	**This study**	**100%**	**100%**	**100%**	**100%**	Pima Indian
	K-Means+C4.5+ANN	2012	Priya et al. [15]	98%	96%	98%	Not Reported	Pima Indian
	ML-BPN	2015	Durairaj and Kalaiselvi [13]	91%	Not Reported	Not Reported	Not Reported	Pima Indian
	KGRNN	**2017**	**This study**	**86%**	**83%**	**87%**	**87%**	Pima Indian
	GRNN	2017	Alby and Shivakumar [140]	85%	91%	73%	Not Reported	Pima Indian
	GA-ML-BPN	2011	Karegowda et al. [14]	84%	Not Reported	Not Reported	Not Reported	Pima Indian
	GRNN	2011	Adeyemo and Akinwonmi [102]	84%	Not Reported	Not Reported	Not Reported	Wesley Guild Unit - Nigeria
	MLP-BPX	**2017**	**This study**	**82%**	**72%**	**73%**	**80%**	Pima Indian
	DTDN	2012	Mehmet et al. [55]	76%	53%	88%	Not Reported	Pima Indian
	PNN	2012	Mehmet et al. [55]	72%	63%	76%	Not Reported	Pima Indian
Instance-based Techniques	KNN	2014	Saxena et al. [8]	97%	Not Reported	Not Reported	N/A	LARS Diabetes
	GA-KNN-Kmeans	2012	Karegowda et al. [20]	89%	Not Reported	Not Reported	N/A	Pima Indian
	CKNN	**2017**	**This study**	**81%**	**89%**	**88%**	**N/A**	Pima Indian
	CVE-K-Means	**2017**	**This study**	**80%**	**61%**	**90%**	**N/A**	Pima Indian
	KNN-K-Means	2014	Vijayan et al. [22]	80%	Not Reported	Not Reported	N/A	Pima Indian
	K-Means Clustering	**2017**	**This study**	**77%**	**55%**	**94%**	**N/A**	Pima Indian
	K-Means	2014	Vijayan et al. [22]	77%	Not Reported	Not Reported	N/A	Pima Indian
	KNN	2014	Vijayan et al. [22]	73%	Not Reported	Not Reported	N/A	Pima Indian
Statistical Techniques	LR+BSF	2016	Devi et al. [144]	90%	Not Reported	Not Reported	Not Reported	AR Hospital - Madurai
	LR	2013	Bassam et al. [143]	84%	Not Reported	Not Reported	Not Reported	Kuwait NHN
	LR-*n*	**2017**	**This study**	**84%**	**72%**	**100%**	**77%**	Pima Indian
	LR	2015	Nai-aruna et al. [87]	82%	Not Reported	Not Reported	Not Reported	Sawanpracharak RH
	LR+Boosting	2015	Nai-aruna et al. [87]	78%	Not Reported	Not Reported	Not Reported	Sawanpracharak RH
	LR+Bagging	2015	Nai-aruna et al. [87]	77%	Not Reported	Not Reported	Not Reported	Sawanpracharak RH
	LR	2013	Meng et al. [91]	77%	80%	72%	Not Reported	Zhuguang - China
Decision Tree	J48	2015	Habibie et al. [31]	99%	70%	Not Reported	88%	Tabriz EA - Iran
	Improved J48	2014	Kaur and Amit [32]	91%	Not Reported	Not Reported	Not Reported	Pima Indian
	C4.5	2012	Ashwinkumar and Anandakumar [37]	68%	Not Reported	Not Reported	Not Reported	BGS Hospital - India
	CART	2012	Ashwinkumar and Anandakumar [37]	44%	Not Reported	Not Reported	Not Reported	BGS Hospital - India
Support Vector Machines	GDA-LS-SVM	2008	K. Polat et al. [43]	79%	73%	80%	N/A	Pima Indian
	LS-SVM	2008	K. Polat et al. [43]	78%	83%	82%	N/A	Pima Indian
	VaR-SVM	2013	Tsyurmasto et al. [44]	93%	Not Reported	Not Reported	N/A	Pima Indian
	SSVM	2009	Purnami et al. [45]	88%	Not Reported	Not Reported	N/A	Pima Indian
	SVM+MLP-BPN	2012	Zolfaghari [47]	79%	Not Reported	Not Reported	N/A	Pima Indian
Rule-based Systems	Fuzzy-Rule	2010	Aibinu [49]	80%	Not Reported	Not Reported	N/A	Pima Indian
	ANFIS-GMDH	2008	Sharifi et al. [118]	80%	Not Reported	Not Reported	N/A	Pima Indian
	Evolving Rule-Base	2010	Lekkas et al. [120]	83%	Not Reported	Not Reported	N/A	Pima Indian
Population-Based Techniques	GA	2010	Aslam and Nandi [50]	75%	Not Reported	Not Reported	N/A	Pima Indian
	GP+CPS	2010	Aslam and Nandi [50]	82%	Not Reported	Not Reported	N/A	Pima Indian

Table 5.1 Comparison of results with those in literature

It can be observed in Table 5.1 that a majority of studies in literature did not report the sensitivity, specificity and the ROC results of their ML techniques. This poses a challenge when one is comparing their results with other experiments. Most of these ML techniques proposed in literature had high classification accuracies, however, their efficacy in positively identifying diabetic cases is not known due to the exclusion of essential algorithm performance measures such as sensitivity and specificity. One can also observe that the Pima Indian dataset was widely used by a majority of studies.

The comparative analysis in Table 5.1 shows that the ANNs followed by SVMs, Instanced-based techniques, and Statistical techniques performed better than other approaches in terms of classification accuracy. Considering the results in Table 5.1, the average classification accuracy of ANNs, SVMs, Instance-based, and Statistical techniques is 86%, 83%, 82%, and 82% respectively. The average classification accuracy for Rule-based techniques, Population-based techniques, and the Decision Tree is 81%, 78%, and 76% respectively.

The B-KGRNN, KGRNN, and CVE-K-Means technique improved the best classification accuracy obtained by similar techniques in literature by 17%, 1%, and 2% respectively. The B-KGRNN technique also outperformed the best performed ANN known in literature by 2%. The *n* model and the CKNN technique were outperformed by the best known similar techniques in literature by 6% and 16% respectively in terms of classification accuracy. These results and the comparative analysis in Table 5.1 show that the ANNs and Instance-based techniques are more effective in solving the diabetes diagnosis problem compared to other methods proposed in literature.

Chapter 6

Conclusion and Future Work

In this study, the Type II diabetes diagnosis problem was presented and addressed using six enhanced Machine Learning techniques. Three different types of enhanced Artificial Neural Networks, a Statistical Regression model, and Instance-based techniques were applied to solve the type II diabetes diagnosis problem. The efficacy of these techniques in diagnosing type II diabetes on various datasets was compared between them and also with previous studies in literature. The classification accuracy, sensitivity, specificity, NPV, PPV, Chi-square test and the ROC were used to evaluate the efficacy of the Machine Learning techniques on various datasets. The Machine Learning techniques proposed in this study obtained very competitive results, and some of them outperformed other Machine Learning techniques proposed in literature.

The Pima Indian diabetes dataset was preprocessed using different data preprocessing and feature extraction techniques to form new datasets which were used to train and evaluate the performance of ML techniques proposed in this study. The Min-Max normalization technique, AHP, Random Forest feature extraction, missing value removal, and missing value imputation were employed as data preprocessing techniques. The ML techniques in this study showed a significant improvement in their performance on preprocessed datasets. Most of these techniques could not reach their optimal performance on the original un-preprocessed dataset with missing values and the dataset

with imputed data. The results showed that the proposed techniques achieved lower results on these two datasets compared to the datasets where instances with missing values were removed or relevant features were extracted. Therefore, one can conclude that data imputation is not effective as feature extraction and removing instances with missing values or a combination of both.

The literature review showed that most studies only used the classification accuracy to show the efficacy of their Machine Learning techniques instead of incorporating other performance measures to clearly demonstrate the effectiveness of their Machine Learning approaches. This study has addressed this by using various performance measures to show the true efficacy of the ML techniques in diagnosing type II diabetes. The ANNs in this study obtained the best results compared to Instance-based techniques and the Statistical technique. The ANNs obtained the highest classification accuracy of 100% which is an optimal result. The Instance-based techniques obtained the best classification accuracy of 81% with 89% sensitivity and 88% specificity. The statistical technique obtained the highest classification accuracy of 84% with 72% sensitivity and 100% specificity. The ANNs have showed good performance on un-preprocessed dataset and also when the combination of missing value and feature extraction was used. Instance-based techniques showed the best performance only when missing values were removed in the dataset. The Statistical technique showed the best performance when missing values were removed and when missing values were imputed with attribute mean in the dataset. The B-KGRNN, CVE-K-Means, and KGRNN outperformed similar techniques in literature in terms of classification accuracy by 17%, 2% , and 1% respectively.

Based on the results obtained in this study and those in previous studies, it can be concluded that the Artificial Neural Networks are more effective in diagnosing type II diabetes compared to other Machine Learning approaches. It can also be concluded that combining the removal of instances with missing values and feature extraction is more effective in improving the performance of ML techniques than other types of data preprocessing methods. In conclusion, this study has shown

that the use of ML techniques which were enhanced using novel approaches, their hybridization, and efficiently preprocessing the dataset provided promising results in type II diabetes diagnosis, of which one of them was an optimal result, i.e., 100% classification accuracy.

A direction of future work in this problem would be to employ the proposed data preprocessing techniques on different datasets in different domains to improve the performance of ML techniques. It would be beneficial as future work to perform a time complexity analysis on the performance of the techniques proposed in this study. Another future work would be to employ the proposed Machine Learning techniques in other domains to ascertain whether improved results could be obtained.

Bibliography

[1] World Diabetes Foundation, International Diabetes Federation, Novo Nordisk, Diabetes South Africa, *Diabetes: the hidden pandemic and its impact on Sub-Saharan Africa* - document prepared for the Diabetes Leadership Forum Africa 2010 - (Edited by Prof Ayesha Motala and Dr Kaushik Ramaiya, 2010).

[2] The National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), *Causes of diabetes*, Website: http://diabetes.niddk.nih.gov/dm/pubs/causes/, Accessed: 2015/02/08.

[3] American Diabetes Association, *Diagnosis and Classification of Diabetes Mellitus*, Website: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2613584/, Accessed: 2015/02/08.

[4] International Diabetes Federation, *Follow-up to the political declaration of the high-level meeting of the General Assembly on the prevention and control of non-communicable diseases*, IDF Diabetes Atlas 6th Edition, 2013.

[5] Diabetes Atlas 6th Edition Committee, *IDF Diabetes Atlas 6th Edition Poster Update*, 2014.

[6] Shankaracharya, Devang Odedra, Medhavi Mallick, Prateek Shukla, Subir Samanta, and Ambarish Vidyarthi, *Java-Based Diabetes Type 2 Prediction Tool for Better Diagnosis*, Diabetes technology and therapeutics, Volume 14, Number 3, 2012.

[7] Nahla H. Barakat, Andrew P. Bradley, and Mohamed Nabil H. Barakat, *Intelligible Support Vector Machines for Diagnosis of Diabetes Mellitus*, IEEE transactions on information technology in biomedicine, Vol. 14, Number 4, July 2010.

[8] Krati Saxena, Zubair Khan, and Shefali Singh, *Diagnosis of Diabetes Mellitus using K Nearest Neighbor Algorithm*, International Journal of Computer Science Trends and Technology (IJCST) - Volume 2 Issue 4, July-Aug 2014.

[9] N. Lavrac, E. Keravnou, and B. Zupan, *Intelligent data analysis in medicine*, Encyclopedia of Computer Science and Technology, Vol. 42, 2000, pp. 113-157.

[10] Najmeh Hosseinpour, Saeed Setayeshi, Karim Ansari-asl and Mohammad Mosleh, *Diabetes Diagnosis by Using Computational Intelligence Algorithms*, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 12, December 2012.

[11] P. Cunningham, Matthieu Cord, and Sarah Jane Delany, *Machine Learning Techniques for Multimedia, Case Studies on Organization*, 2008.

[12] Murali Shanker, M.Y. Hu, M.S. Hung, *Estimating Probabilities of Diabetes Mellitus, Using Neural Networks*, November 13, 1999.

[13] M.Durairaj, G.Kalaiselvi, *Prediction of diabetes using back propagation algorithm*, International Journal of Emerging Technology and Innovative Engineering Volume 1, Issue 8, August 2015 (ISSN: 2394 - 6598).

[14] Asha Gowda Karegowda, A.S. Manjunath, and M.A. Jayaram, *Application Of Genetic Algorithm Optimized Neural Network Connection Weights For Medical Diagnosis Of Pima Indians Diabetes*, International Journal on Soft Computing (IJSC), Vol. 2, No. 2, May 2011.

[15] S. Priya and R.R. Rajalaxmi, *An Improved Data Mining Model to Predict the Occurrence of Type-2 Diabetes using Neural Network*,International Journal of Computer Applications (0975 - 8887), Volume 95 - No.17, 2012.

[16] Blanca S. Leona, Y. Alma., N. Alanisb, N. Edgar, O. Sanchea Fernando, and V. EduardoRuiz, *Inverse optimal neural control of blood glucose level for type 1 diabetes mellitus patients*, Journal of the Franklin Institute, Vol. 349, pp. 1851-1870, 2012.

[17] Sebastian Polak and Aleksander Mendyk, *Artificial neural networks based Internet hypertension prediction tool development and validation*, Applied Soft Computing , pp. 734âĂŞ739, 2008.

[18] B.M. Patil, R.C. Joshi, and Durga Toshniwal, *Hybrid Prediction Model for Type-2 Diabetic patients*, Expert Systems with Applications, Science direct, pp. 8102-8108, 2010.

[19] C. Yue, L. Xin, X. Kewen, and S. Chang, *An Intelligent Diagnosis to Type 2 Diabetes Based on QPSO algorithm and WLS-SVM*, IEEE International Symposium on Intelligent Information Technology Application Workshops, 2008.

[20] Asha Gowda Karegowda, M.A. Jayaram, and A.S. Manjunath, *Cascading K-means Clustering and K-Nearest Neighbor Classifier for Categorization of Diabetic Patients*, IJEAT Vol.1, No.3, pp 147-151, 2012.

[21] University of Carlifornia Irvine, *UCI machine learning repository and archives*, website: https://www.ics.uci.edu/ml/datasets.html, Accessed: 20/03/2016.

[22] Veena Vijayan and Aswathy Ravikumar, *Study of Data Mining Algorithms for Prediction and Diagnosis of Diabetes Mellitus*, International Journal of Computer Applications (0975-8887), Volume 95 - No.17, June 2014.

[23] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer, *KNN Model-Based Approach in Classification*.

[24] G. Gates, *The Reduced Nearest Neighbour Rule*, IEEE Transactions on Information Theory, Vol. 18, pp. 431-433, 1972.

[25] E. Alpaydin, *Voting Over Multiple Condensed Nearest Neighbors*, Artificial Intelligence Review, Vol. 11, pp. 115-132, 1997.

[26] M. Kubat, *Voting Nearest-Neighbour Sub-classifiers*, Poceedings of the 17-th International Conference on Machine Learning, pp. 503-510, Stanford, CA, June 29 -July 2, 2000.

[27] P. Hart, *The Condensed Nearest Neighbour Rule*, IEEE Transactions on Information Theory, Vol. 14, pp. 515-516, 1968.

[28] Murali S. Shanker, *Using Neural Networks to predict the onset of diabetes mellitus*, Journal of Chemical Information and Computer Sciences, Vol. 36, 1996.

[29] Padraic G. Neville, *Decision Trees for Predictive Modeling*, AS Institute Inc, 4 August 1999.

[30] Yan-yan Song and Ying Lu, *Decision tree methods: applications for classification and prediction*, Shanghai Archives of Psychiatry, Vol. 27, No. 2, 2015.

[31] Shafi Habibi, Maryam Ahmadi, and Somayeh Alizadeh3, *Type 2 Diabetes Mellitus Screening and Risk Factors Using Decision Tree: Results of Data Mining*, Global Journal of Health Science; Vol. 7, No. 5; Published by Canadian Center of Science and Education, 2015.

[32] Gaganjot Kaur and Amit Chhabra, *Improved J48 Classification Algorithm for the Prediction of Diabetes*, International Journal of Computer Applications, Volume 98, No. 22, July 2014.

[33] Korting Thales Sehn, *C4.5 algorithm and Multivariate Decision Trees*, Image Processing Division, National Institute for Space Research - INPE.

[34] A. Nadali, E.N. Kakhky, and H.E. Nosratabadi, *Evaluating the success level of data mining projects based on CRISP-DM methodology by a Fuzzy expert system*, Electronics Computer Technology (ICECT), 3rd International Conference, vol.6, pp. 161-165, 8-10 April 2011.

[35] Aiswarya Iyer, S. Jeyalatha and Ronak Sumbaly, *Diagnosis of diabetes using classification mining techniques*, International Journal of Data Mining and Knowledge Management Process (IJDKP) Vol.5, No. 1, January 2015.

[36] Saba Bashir, Usman Qamar, Farhan Hassan Khan, and M. Younus Javed, *An Efficient Rule-Based Classification of Diabetes Using ID3, C4.5, and CART Ensembles*, Conference paper, 12-th International Conference on Frontiers of Information Technology (FIT), At Islamabad Pakistan, Vol. 12.

[37] U.M. Ashwinkumar and K.R. Anandakumar, *Predicting Early Detection of Cardiac and Diabetes Symptoms using Data Mining Techniques*, 2nd International Conference on Computer Design and Engineering, 2012.

[38] P. Radha and B. Srinivasan, *Predicting Diabetes by co-sequencing the various Data Mining Classification Techniques*, IJISET - International Journal of Innovative Science, Engineering Technology, Vol. 1 Issue 6, August 2014.

[39] R. Ali, M.H. Siddiqi, M. Idris, and B.H. Kang, *Prediction of Diabetes Mellitus Based on Boosting Ensemble Modeling*, 8-th International conference on Ubiquitous computing Ambient intelligence, 2014.

[40] K. Rajesh and V. Sangeetha, *Application of Data Mining Methods and Techniques for Diabetes Diagnosis*, International Journal of Engineering and Innovative Technology (IJEIT), Vol 2 Issue 3, September 2012.

[41] K. Soman and V.A. Loganathan, *Machine Learning with SVM and other Kernel Methods*, Prentice Hall of India, 2009.

[42] Datong Chen and Jean-Marc Odobez, *Comparison of Support Vector Machine and Neural Network for text texture classification.*

[43] K. Polat, S. Gunes and A. Arslan, *A cascade learning system for classification of diabetes disease: Generalized Discriminant Analysis and Least Square Support Vector Machine*, Expert Systems with Applications, Vol. 34, No. 1, pp. 482-487, 2008.

[44] Peter Tsyurmasto, Michael Zabarankin, and Stan Uryasev, *Value-at-risk Support Vector Machine: Stability to outliers*, 2013.

[45] Santi Wulan Purnami, Abdullah Embong, Jasni Mohd Zain and S.P. Rahayu, *A New Smooth Support Vector Machine and Its Applications in Diabetes Disease Diagnosis*, Journal of Computer Science, Vol. 5, No. 12, pp. 1003-1008, 2009.

[46] Ming-Hsuan Yang and Antoine Cornu ejols, *Introduction to Support Vector Machines*, Website: http://u.cs.biu.ac.il/ haimga/Teaching/AI/saritLectures/svm.pdf, Accessed: 2016/10/22.

[47] Rahmat Zolfaghari, *Diagnosis of Diabetes in Female Population of Pima Indian Heritage with Ensemble of BP Neural Network and SVM*, IJCEM International Journal of Computational Engineering Management, Vol. 15 Issue 4, July 2012.

[48] M.F. Ganji and M.S. Abadeh, *Using fuzzy ant colony optimization for diagnosis of diabetes disease*, 18-th Proceedings of Iranian Conference on Electrical Engineering (ICEE), pp. 501-505, 2010.

[49] A. Aibinu, M.J. Salami and A.A Shafie, *Application of modeling techniques to diabetes diagnosis*, Proc. IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES), pp. 194-198, 2010.

[50] M.W. Aslam and A.K. Nandi, *Detection of Diabetes Using Genetic Programming*, Proc. 18-th European Signal Processing Conference, pp. 1184-1188, 2010.

[51] M. Pasrapoor and U. Bilstrup, *An Emotional Learning-inspired Ensemble Classifier (ELiEC)*, Proc. Conference on Computer Science and Information Systems (FedCSIS), pp. 137-141, 2013.

[52] Jack W. Smith, J.E. Everhart, W.C. Dicksont, W.C. Knowler, and R.S. Johannes, *Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus*.

[53] Ronald K. Pearson, *The Problem of Disguised Missing Data*, Website: http://preview.kdd.org/explorationfiles/12-Pearson.pdf, Accessed: 2016/10/23

[54] J. Breault, *Data mining diabetic databases: Are rough sets a useful addition?*, Proc. 33rd Symposium on the Interface, Computing Science and Statistics, Fairfax, VA, 2001.

[55] Mehmet Recep and Cengiz Sertkaya, *Comparison of different methods for determining diabetes*, Turkish Journal of Electrical Engineering and Computer Sciences, 2014.

[56] E.D. Ubeyli, *Modified mixture of experts for diabetes diagnosis*, Journal of Medical Systems Vol. 33, pp. 299-305, 2009.

[57] Shankaracharya, Devang Odedra, Subir Samanta, and Ambarish S. Vidyarthi, *Computational Intelligence in Early Diabetes Diagnosis: A Review, the review of diabetes studies*, Vol. 7, No. 4 , 2010.

[58] B.Y. Baha, G. M. Wajiga, N. V. Blamah, and A. O. Adewumi, *Analytical hierarchy process model for severity of risk factors associated with type 2 diabetes*, Academic Journals, Vol 8 (39), pp. 1906 - 1910, October 2013.

[59] T.L. Saaty, *Decision making with the analytic hierarchy process*, International Journal of Services Sciences, Vol. 1, No. 1, pp. 83-98, 2008.

[60] World Health Organization, *Global report on diabetes*, 2016, ISBN 978 92 4 156525 7.

[61] Vidar Hjellvik, Solveig Sakshaug, and Hanne Strom, *Body mass index, triglycerides, glucose, and blood pressure as predictors of type 2 diabetes in a middle-aged Norwegian cohort of men and women*, Clinical epidemiology journal, 17 August, 2012.

[62] World Health Organization, *Global health risks global health risks who mortality and burden of disease attributable to selected major risks*, 2009, ISBN 978 92 4 156387

[63] M. Jolly, N. Sebire, J. Harris, S. Robinson, and L. Regan, *The risks associated with pregnancy in women aged 35 years or older*, Human Reproduction ; Vol. 15 (11), pp. 2433-2437, DOI: 10.1093/humrep/15.11.2433, 2000.

[64] Blood pressure UK, *Diabetes and high blood pressure*, Website: http://www.bloodpressureuk.org/BloodPressureandyou/Yourbody/Diabetes, Accessed: 2016/12/10.

[65] NHS Choices UK, *High blood pressure: does it lead to diabetes?*, Website: http://www.nhs.uk/news/2015/10October/Pages/high-blood-pressure-does-it-leads-to-diabetes.aspx, Accessed: 10 January 2017.

[66] Janos Fulop, *Introduction to decision making methods*, Laboratory of Operations Research and Decision Systems, Computer and Automation Institute, Hungarian Academy of Sciences.

[67] Evangelos Triantaphyllou and Stuart H. Mann, *USing the Analytic Hierarchy Process for decision making in engineering applications: some challenges*, International Journal of Industrial Engineering: Applications and Practice, Vol. 2, No. 1, pp. 35-44, 1995.

[68] Pawel Cabala, *Using the Analytic Hierarchy Process in evaluating decision alternatives*, 2010.

[69] Madhu Yedla, Srinivasa Rao Pathakota, and T.M. Srinivasa, *Enhancing K-means Clustering Algorithm with Improved Initial Centers*, International Journal of Computer Science and Information Technologies, Vol. 1 (2), pp. 121-125, 2010.

[70] T. Santhanam and M.S Padmavathi, *Application of K-Means and Genetic Algorithms for Dimension Reduction by Integrating SVM for Diabetes Diagnosis*, Procedia Computer Science, Vol. 47, pp. 76 - 83, 2015.

[71] Jason Brownlee, *Machine learning mastery - Introduction to Machine Learning*, Website: http://machinelearningmastery.com /, Accessed: 01 July 2016.

[72] Y. Angeline Christobel and P.Sivaprakasam, *A New Classwise k Nearest Neighbor (CKNN) Method for the Classification of Diabetes Dataset*, International Journal of Engineering and Advanced Technology (IJEAT), Vol. 2 Issue 3, pp. 2249 - 8958, February 2013.

[73] L. Jiang, D.Wang, Z. Cai, S. Jiang, and X. Yan, *Scaling up the accuracy of k-nearest-neighbour classifiers: a naive-bayes hybrid*, International Journal of Computers and Applications, Vol. 31, No. 1, 2009.

[74] Nidhi Singh and Divakar Singh, *Performance Evaluation of K-Means and Hierarchical Clustering in Terms of Accuracy and Running Time*, International Journal of Computer Science and Information Technologies, Vol. 3 (3), pp. 4119-4121, 2012.

[75] Shadi I. Abudalfa and Mohammad Mikki, *K-means algorithm with a novel distance* , Turkish Journal of Electrical Engineering and Computer Sciences, 2013.

[76] Agatonovic-Kustrin and R. Beresford, *Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research*, Journal of Pharm Biomed Anal, Vol. 22(5),m pp. 717-27, Jun 2000.

[77] Graham Templeton, *Artificial neural networks are changing the world. What are they?*, Website: https://www.extremetech.com/extreme/215170-artificial-neural-networks-are-changing-the-world-what-are-they, Accessed: 20 February 2017

[78] D.F. Specht, *A general regression neural network*, IEEE Transactions on Neural Networks Vol. 2(6), pp. 568-576, 1991.

[79] Kamer Kayaer and Tulay Yildirim, *Medical diagnosis on Pima Indian diabetes using general regression neural networks*, Researchgate, 17 September 2014.

[80] S. Alby and B.L. Shivakumar, *A prediction model for type 2 diabetes using adaptive neuro-fuzzy interface system*, Biomedical Research 2017; Special Issue, Article Accepted Accepted on February 16, 2017.

[81] Ramalingaswamy Cheruku, Damodar Reddy Edla, and Venkatanareshbabu Kuppili, *Diabetes Classification using Radial Basis Function Network by Combining Cluster Validity Index and BAT Optimization with Novel Fitness Function*, International Journal of Computational Intelligence Systems, Vol. 10, pp. 247-265, 2017.

[82] Zahed Soltani, *A New Artificial Neural Networks Approach for Diagnosing Diabetes Disease Type II*, International Journal of Advanced Computer Science and Applications, Vol. 7, No. 6, 2016.

[83] S. Tharani and C. Yamini, *Classification using Convolutional Neural Network for Heart and Diabetics Datasets*, International Journal of Advanced Research in Computer and Communication Engineering ISO 3297, Vol. 5 Issue 12, December 2016.

[84] Dilip Kumar Choubey, *GAMLP NN: A Hybrid Intelligent System for Diabetes Disease Diagnosis*, International Journal of Intelligent Systems and Applications, Vol. 1, pp. 49-59, 2016.

[85] Chris McCormick, *AdaBoost Tutorial*, Website: http://mccormickml.com/2013/12/13/adaboost-tutorial/, Accessed: 10 April 2017.

[86] Yoav Freund and Robert E. Schapire, *A Short Introduction to Boosting*, Journal of Japanese Society for Artificial Intelligence, Vol. 14(5), pp. 771-780, September, 1999 (In Japanese, translation by Naoki Abe.).

[87] Nongyao Nai-Aruna and Rungruttikarn Moungmaia, *Comparison of Classifiers for the Risk of Diabetes Prediction*, 7-th International Conference on Advances in Information Technology, Procedia Computer Science, Vol. 69, pp. 132 - 142, 2015.

[88] Andrew Ng, *Regression with Numerical Optimization - Logistic regression*, CSG220 Machine Learning Fall 2008.

[89] Jeff Howbert, *Introduction to Machine Learning âĂŞ Logistic Regression*, Winter 2012, Website: http://courses.washington.edu/css490/2012.Winter/lecture$_s$$lides$/05$b_l$$ogistic$regression.pdf, Accessed: 14 May 2017.

[90] Javier Cabrera, Venkata Sasikiran Goteti, Michael Pannucci, and Juan Zhang, *Logistic Regression Analysis of the Occurrence of Diabetes in Pima Indian Women*, March 24, 2004, Website: www.rci.rutgers.edu/ cabrera/587/pima.pdf, Accessed: 14 May 2017.

[91] Xue-Hui Meng, Yi-Xiang Huang, Dong-Ping Rao, Qiu Zhang, and Qing Liu, *Comparison of three data mining models for predicting diabetes or prediabetes by risk factors*, Kaohsiung Journal of Medical Sciences, Vol. 29, pp. 93-99, 2013.

[92] Wei Lu, *Neural Network Model for Distortion Buckling Behaviour of Cold-Formed Steel Compression Members*, Helsinki University of Technology Laboratory of Steel Structures Publications 16, 2000.

[93] Alex Castrounis, *Deep Learning Explained - Neural Networks*, Website: www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html, Accessed: 15 May 2017.

[94] J.G. Makin, *Backpropagation*, February 15, 2006, Website: www.cs.cornell.edu/courses/cs5740/2016sp/resources/backprop.pdf, Accessed: 16 May 2017.

[95] UCC Ireland, *Computer Science Notes âĂŞ the back propagation algorithm*, Website: http://www.cs.ucc.ie/dgb/courses/ai1/07-notes.pdf, Accessed: 16 May 2017.

[96] Gustav Larsson, *Initialization of deep networks*, Website: http://deepdish.io/2015/02/24/network-initialization/, Accessed: 19 May 2017.

[97] Prateek Joshi, *Understanding Xavier Initialization In Deep Neural Networks*, March 29 2016, Website: https://prateekvjoshi.com/2016/03/29/understanding-xavier-initialization-in-deep-neural-networks/, Accessed: 21 May 2017.

[98] S. Poonkuzhali, N. Deepika, J. Jeyalakshmi, and S. Sreeshuba, *Intelligent Data Analytics System to Predict Diabetes using Artificial Neural Network*, Journal of Chemical and Pharmaceutical Sciences, JCHPS Special Issue 9, December 2016.

[99] Venkata Akhil Karumuri, *A Comparative Study of ID3 and MLP Algorithms, International journal of innovative technology and research*, Vol. 4, Issue 1,pp. 2684 - 2688, December - January 2016.

[100] Sabreena Jan and Vinod Sharma, *A Study of various data mining techniques for diabetic prognosis*, International Journal of Modern Computer Science (IJMCS), Vol. 4, Issue 3, June 2016.

[101] Manaswini Pradhan and Ranjit Kumar Sahu, *Predict the onset of diabetes disease using Artificial Neural Network (ANN)*, International Journal of Computer Science Emerging Technologies, Volume 2, Issue 2, April 2011.

[102] A.B. Adeyemo and A.E. Akinwonmi, *On the Diagnosis of Diabetes Mellitus Using Artificial Neural Network Models*, African Journal of Comp ICT, Vol. 4., No. 2 Issue 1, 2011.

[103] P. Radha and B. Srinivasan, *Predicting Diabetes by co-sequencing the various Data Mining Classification Techniques*, International Journal of Innovative Science, Engineering Technology, Vol. 1, Issue 6, August 2014.

[104] Max Bramer, *Undergraduate topics in computer science*, Principles of data mining, 2nd Edition, pp. 2.

[105] Nicholas Gould, *An introduction to algorithms for continuous optimization*, Oxford University Computing Laboratory.

[106] Xin-She Yang, *Metaheuristic Optimization*, Scholarpedia 2011, Website: http://www.scholarpedia.org/article/MetaheuristicOptimization, Accessed: 20 May 2017.

[107] Neos Optimization Guide, *Types of Optimization Problems*, Website: https://neos-guide.org/optimization-tree, Accessed: 23 June 2017.

[108] Leo Liberti and Sergei Kucherenko, *Comparison of deterministic and stochastic approaches to global optimization*, 16 May 2005.

[109] Deeplearning4J, *A Beginner's Guide to Recurrent Networks and LSTMs*, Website: https://deeplearning4j.org/lstm.html, Accessed: 12 May 2017.

[110] Brian Leke, *HIV analysis using Computational Intelligence*, Phd Computer Science Thesis, February 2008.

[111] John McCullock and Elman,*Neural Networks Architectures*, Website: http://mnemstudio.org/neural-networks-elman.htm, Accessed: 13 April 2017.

[112] Bo Jiang, Yi Zhang, Shunlin Liang, Xiaotong Zhang, and Zhiqiang Xiao, *Surface Daytime Net Radiation Estimation Using Artificial Neural Networks*, Remote Sens. 2014, Vol. 6(11).

[113] M.F. Wilkins, *Identification of Phytoplankton from Flow Cytometry Data by Using Radial Basis Function Neural Networks*, Appl. Environ. Microbiol., pp. 4404-4410, 1999.

[114] I. Yousif and A.L. Mashhadany, *Recurrent Neural Network with Human Simulator Based Virtual Reality*.

[115] Learn by Marketing, *K-Means Clustering - What it is and How it Works*, Website: http://www.learnbymarketing.com/methods/k-means-clustering/, Accessed: 12 April 2017.

[116] X. Gu, T. Ni, and H.Wang, *New Fuzzy Support Vector Machine for the Class Imbalance Problem in Medical Datasets Classification*, The Scientific World Journal, pp. 1-12, 2014.

[117] R. Yager and L. Zadeh, *An introduction to fuzzy logic applications in intelligent systems*, Kluwer Academic Publishers Norwell, MA, USA, 1992.

[118] A. Sharifi, A.Vosolipour and M. Teshnehlab, *Hierarchical Takagi-Sugeno type fuzzy system for diabetes mellitus forecasting*, Proc. International Conference on Machine Learning and Cybernetics, pp. 1265-1270, 2008.

[119] M.F. Ganji and M.S. Abadeh, *Using fuzzy ant colony optimization for diagnosis of diabetes disease*, Proc. 18-th Iranian Conference on Electrical Engineering (ICEE), pp. 501-505, 2010.

[120] S. Lekkas and L. Mikhailov, *Evolving fuzzy medical diagnosis of Pima Indians diabetes and of dermatological diseases*, Artificial Intelligence in Medicine, Vol. 50, No. 2, pp. 117-126, 2010.

[121] R. Mallika and V. Saravanan, *An SVM Based Classification Method For Cancer Data Using Minimum Microarray Gene Expressions*, World Academy Of Science, Engineering And Technology 62, 2010.

[122] World Health Organization, *Cardiovascular Diseases (CVDs)- Fact Sheet*, May 2017, Website: http://www.who.int/mediacentre/factsheets/fs317/en/, Accessed: 01 June 2017.

[123] Sellappan Palaniappan and Rafiah Awang, *Intelligent Heart Disease Prediction System Using Data Mining Techniques*, International Journal of Computer Science and Network Security, Vol .8, No.8, August 2008.

[124] Y. Kangwanariyakul, N. Chanin, T. Tanawut, and N. Thanakorn, *Data Mining of Magnetocardiograms for Prediction of Ischemic Heart Disease*, EXCLI Journal, Vol. 33(9), pp. 82-95, 2010.

[125] World Health Organization, *Cancer- Fact Sheet*, February 2017, Website: http://www.who.int/mediacentre/factsheets/fs297/en/, Accessed: 02 June 2017.

[126] Ying Lu and Jiawei Han, *Cancer classification using gene expression data*, Eslevier Journal of Information Systems, Vol. 28, Issue 4, pp. 243-268, June 2003.

[127] Zhenyu Wang and V. Palade, *A Comprehensive Fuzzy-Based Framework for Cancer Microarray Data Gene Expression Analysis*, Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering, 2007.

[128] Pooja Agrawal, Suresh kashyap, Vikas Chandra Pandey, and Suraj Prasad Keshri, *Knowledge Patterns in Clinical Data through Data Mining: A Review on Cancer Disease Prediction*, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 4, April 2013.

[129] A.M. De Silva and P.H.W. Leong, *Grammar-Based Feature Generation for Time Series Prediction*, pp. 99, 2015.

[130] Hani Tamim, *Receiver Operating Characteristic*, Department of Internal Medicine American University of Beirut, Website: https://www.aub.edu.lb/sharp/Documents/ROC.pdf, Accessed: 15 June 2017.

[131] K.Vembandasamy and T.Karthikeyan, *Classification Using Data Mining Novel Outlier Detection In Diabetics*, International Journal of Applied Engineering Research ISSN 0973-4562, Vol. 11, No. 2, pp pp. 1400-1403, 2016

[132] K.M. Embrechts, B. Szymanski, K. Sternickel, T. Naenna and R. Bragaspathi, *Use of machine learning for classification of magnetocardiograms*, Systems Management and Cybernetics, IEEE International Conference Vol. 5, 2003.

[133] P. Disorntetiwat, *Global stock index forecasting using multiple generalized regression neural networks with a gating network*, PhD Thesis, University of Missouri-Rolla, USAoastal and Shelf Sciences.

[134] Halil Bisgin, Orhan Usame Kilinc, Ahmet Ugur and Volkan Tuzcu *Diagnosis of long QT syndrome via support vector machines classification*, Journal of biomedical science and engineering, PP:264-271, January 2011.

[135] V. Sujatha, *An Intelligent Expert Based System Neural Network For The Diagnosis Of Type2 Diabetes Patient*, International Journal of Innovative Research in Advanced Engineering Volume 1 Issue 2 (April 2014).

[136] Hossein Hakimpoor, Khairil Anuar Bin Arshad, Huam Hon Tat, Naser Khani, and Mohsen Rahmandoust, *Artificial Neural Networks' Applications in Management*, World Applied Sciences Journal 14 (7): 1008-1019, 2011.

[137] Hasan Temurtas, Nejat Yumusak, and Feyzullah Temurtas,*A comparative study on diabetes disease diagnosis using neural networks*, Expert Systems with Applications 36 (2009)

[138] Yoichi Hayashi and Shonosuke Yukita, *Rule extraction using Recursive-Rule extraction algorithm with J48graft combined with sampling selection techniques for the diagnosis of type2 diabetes mellitus in the Pima Indian dataset*, Informatics in Medicine Unlocked issue 2, pp. 92-104 (2016)

[139] David Page, *Lecture Notes in CS 760: Machine Learning (2017) - Evaluating Machine Learning Methods*, University of Wisconsin-Madison, Website: www.cs.wisc.edu/ dpage/cs760/, Date Accessed: 09 May 2018

[140] S. Alby, B. Shivakumar, *A prediction model for type 2 diabetes using adaptive neuro-fuzzy interface*, Journal of Biomedical Research, Special Issue: S69-S74 (2018)

[141] S. Ramesh, H. Balaji, *Optimal predictive analytics of pima diabetics using deep learning*, International Journal of Database Theory and Application (2017)

[142] Sushruta Mishra, Brojo Kishore Mishra, Bijalayaxmi Panda Panda, *Impact of Swarm Intelligence Techniques in Diabetes Disease Risk Prediction*, International Journal of Knowledge Discovery in Bioinformatics, Vol 6 (2016)

[143] Bassam Farran, Arshad Mohamed Channanath, Kazem Behbehani, Thangavel Alphonse Thanaraj, *Predictive models to assess risk of type 2 diabetes, hypertension and comorbidity: machine-learning algorithms and validation using national health data from Kuwait - a cohort study*, BMJ Open, Vol 3(5), 2013

[144] M. Nirmala Devi, Appavu Alias Balamurugan, M. Reshma Kris, *IDeveloping a Modified Logistic Regression Model for Diabetes Mellitus and Identifying the Important Factors of Type II DM*, Indian Journal of Science and Technology, Vol 9(4), 2016

[145] Saed Sayad, *Real Time Data Mining*, Self-Help publishers, Cambridge-Ontario, pp: 145 - 147 (2011)

[146] K. Thangadurai and N.Nandhini, *Comparison of datamining algorithms for prediction and diagnosis of diabetesmellitus*, International Journal of Scientific Engineering Research, Vol 7 (2016)

[147] H. Liu, H. Motoda, R. Setiono, and Z. Zhao, *Feature selection: An ever evolving frontier in data mining*, In Proc. The Fourth Workshop on Feature Selection in Data Mining, Vol 4, pp: 4-13 (2010)

[148] A.R. Hadadian and Ali Rasoulian, *Using Analytic Hierarchy Process (AHP) for Selecting the Appropriate Country for Economic Integration (Case of IranâĂŹs Foreign Trade with OIC Countries)*, International Research Journal of Finance and Economics Issue 162 (2017)

[149] Xavier Glorot and Yoshua Bengio, *Understanding the difficulty of training deep feedforward neural networks*, Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp: 249-256 (2010)

[150] Andrew Ng, *Deep Learning*, Stanford University - CS229 Lecture Notes (2017), website: cs229.stanford.edu/notes/, Date Accessed: December 2017

[151] Wen Zhu and Nancy Zengm *Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS Implementations*, NESUG proceedings: Health Care and Life Sciences, Baltimore, Maryland (2010)

[152] O. Caelen, *A Bayesian interpretation of the confusion matrix*, Annals of Mathematics and Artificial Intelligence, Volume 81(3), pp: 429-450 (2017)

[153] Anthony K. Akobeng, *Understanding diagnostic tests 1: sensitivity, specificity and predictive values*, Acta Paediatr vol 96, pp: 338âĂŞ41 (2007)

[154] RL Plackett, *Karl Pearson and the Chi-squared test*, International Statistical Review, pp: 59-72 (2006), website: https://umanitoba.ca/faculties/medicine/units/pediatrics/sections/.

Appendix A

Feature Selection with Weka Machine Learning Tool

To perform feature selection in Weka, follow the following intial steps:

1. Open the Weka GUI Chooser
2. Click the “Explorer” button to launch the Explorer
3. Open your desired dataset
4. Click the “Preprocess” tab to see data preprocessing options available
5. Convert the class variable from numeric to binary by clicking the “Filter” option using “NumericToBinary” filter as shown in figure A.1
6. Click the “Select attributes” tab to access the feature selection methods as shown in figure A.2.

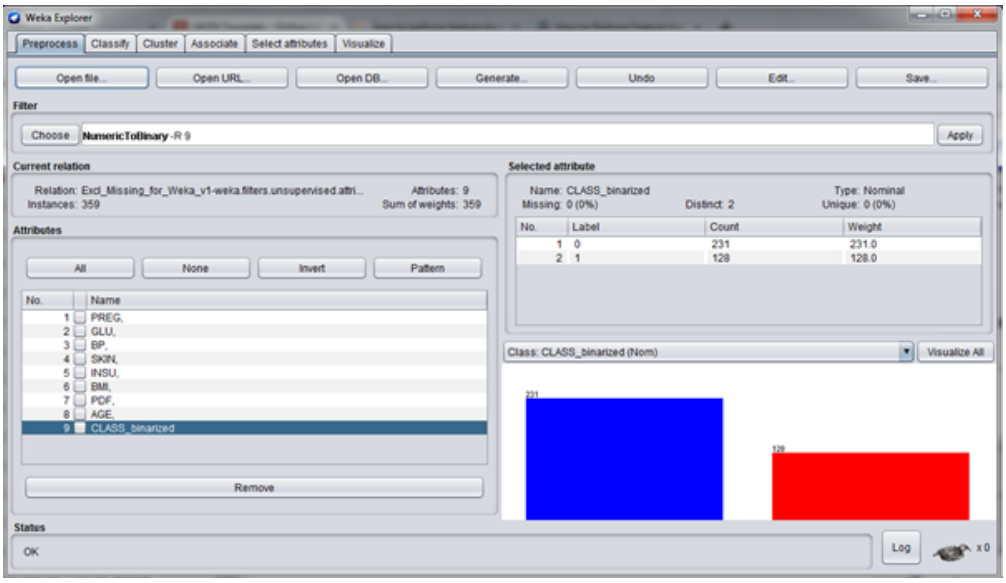

Figure A.1 Weka preprocess tab

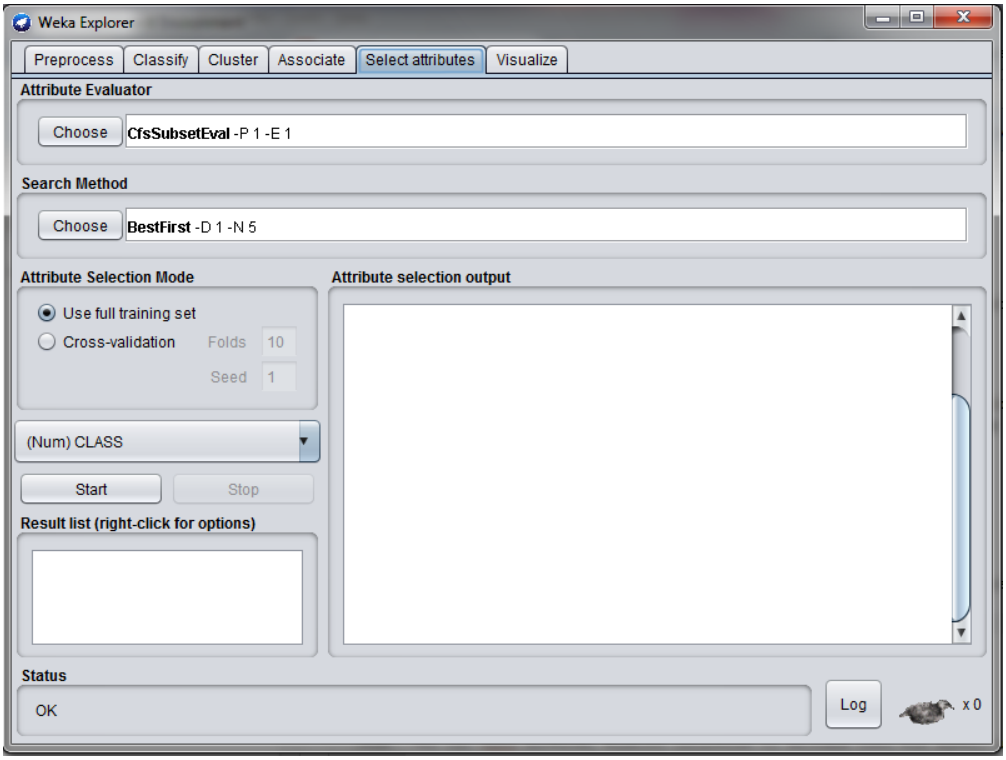

Figure A.2 Weka feature selection initial tab

The way Weka performs feature selection is divided into two parts: Attribute Evaluator and Search Method. The difference between the two parts is that an attribute evaluator evaluates the significance of a single attribute in the context of the output variable while the Search Method evaluates different combinations of attributes in order to find a set of attributes which are significant than others in the context of the output variable such as a class variable. Weka allows the configuration of an attribute evaluator and a search method.

7. Under “Attribute Evaluator” option, select “WrapperSubsetEval” option as the attribute evaluator, “BestFirst” as Search Method and then select “Random Forest” as the learner as shown in figure A.3.

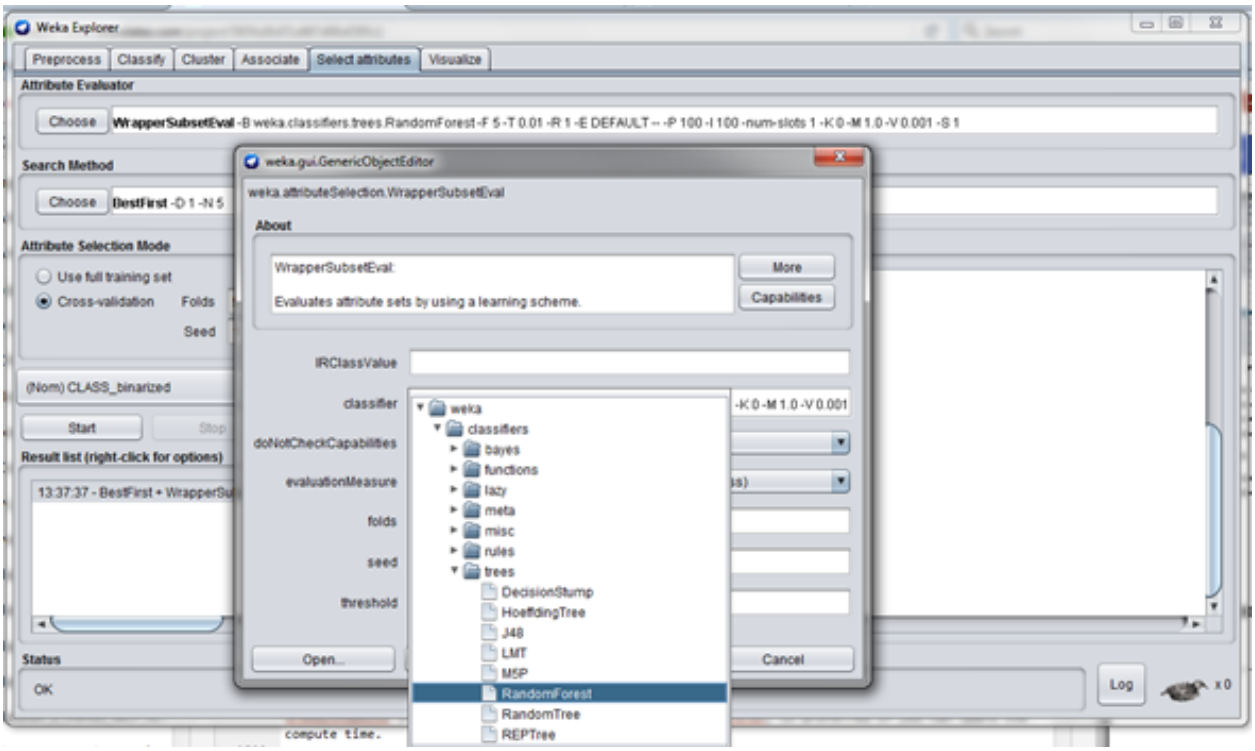

Figure A.3 Weka Selection of Attribute Evaluator, Search Method and Learner

8. Insert number of folds for CV under “Attribue Selection Mode”

9. Click on “Start” button, then Weka will start selecting features on each fold as shown in figure A.4 (this may take several minutes depending on your dataset size)

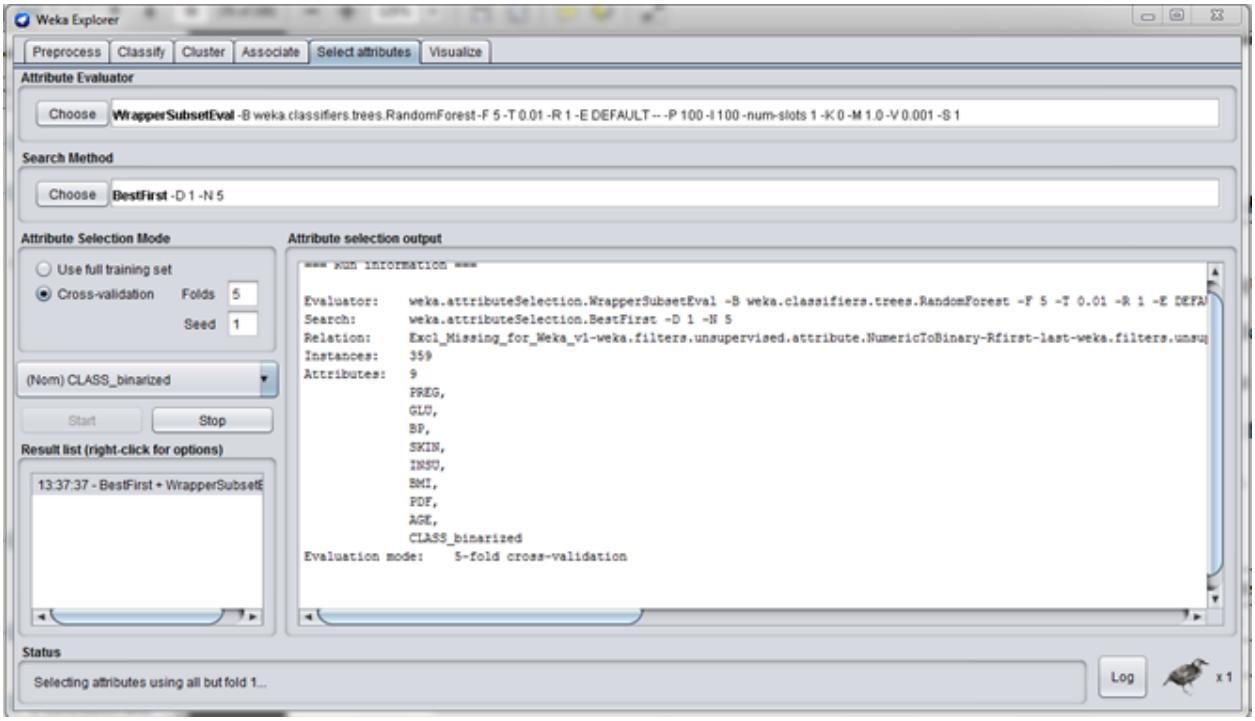

Figure A.4 Weka feature selection execution

10. The results of the feature selection technique are obtained and presented as shown in figure A.5.

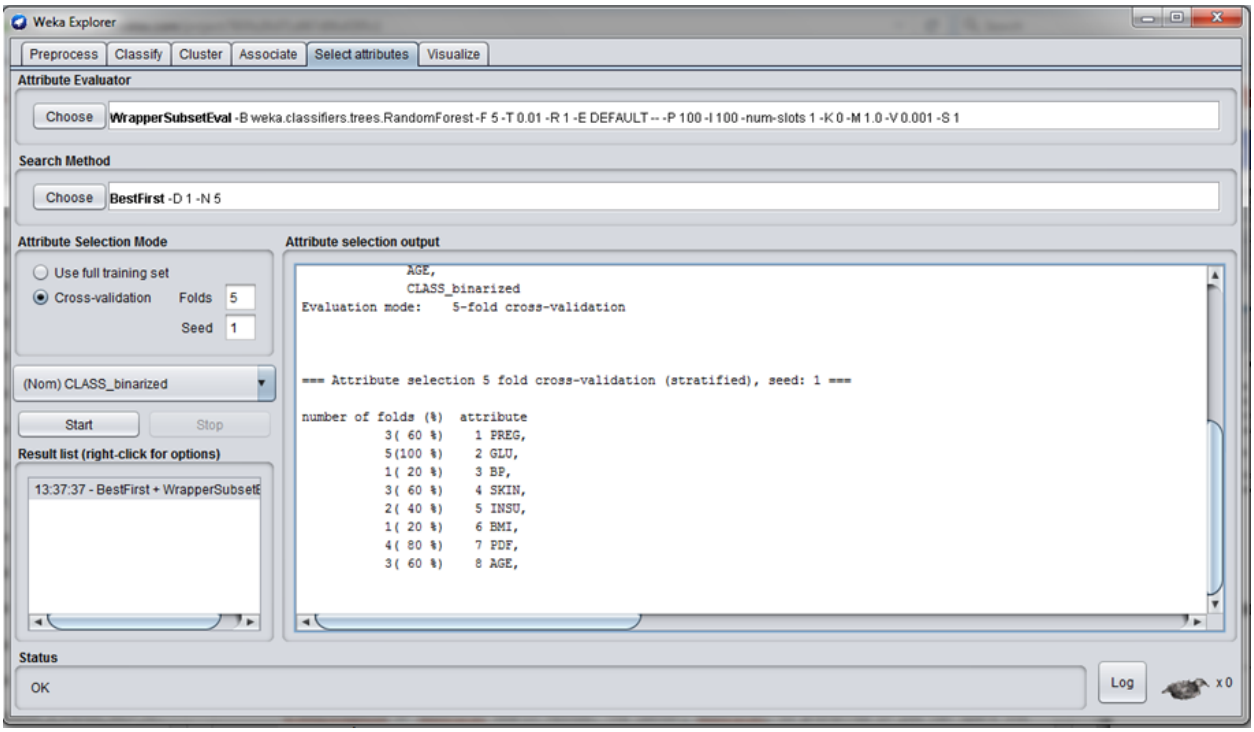

Figure A.5 Weka Learner-based feature selection results

Appendix B

User Manual

An easy to use Java GUI-based System was developed in this study for training and testing all the algorithms proposed in this study.

B.0.1 Diabetes Classification System Requirements

In order to be able to run the diabetes classification system, a testing device should be running any Windows OS software which supports JAVA 1.6 software. It is necessary for the java software to be installed before running the classification system.

B.0.2 Running the Application

To run the application, one must locate the jar file named "DiabetesClassifier.jar" in the JAR folder that's within the "Executables" folder on files submitted with this dissertation. To run the application, a uers must right click it and select "Open" option. An application will start and show the main window that allows a user to select a desired ML technique as shown in figure B.1.

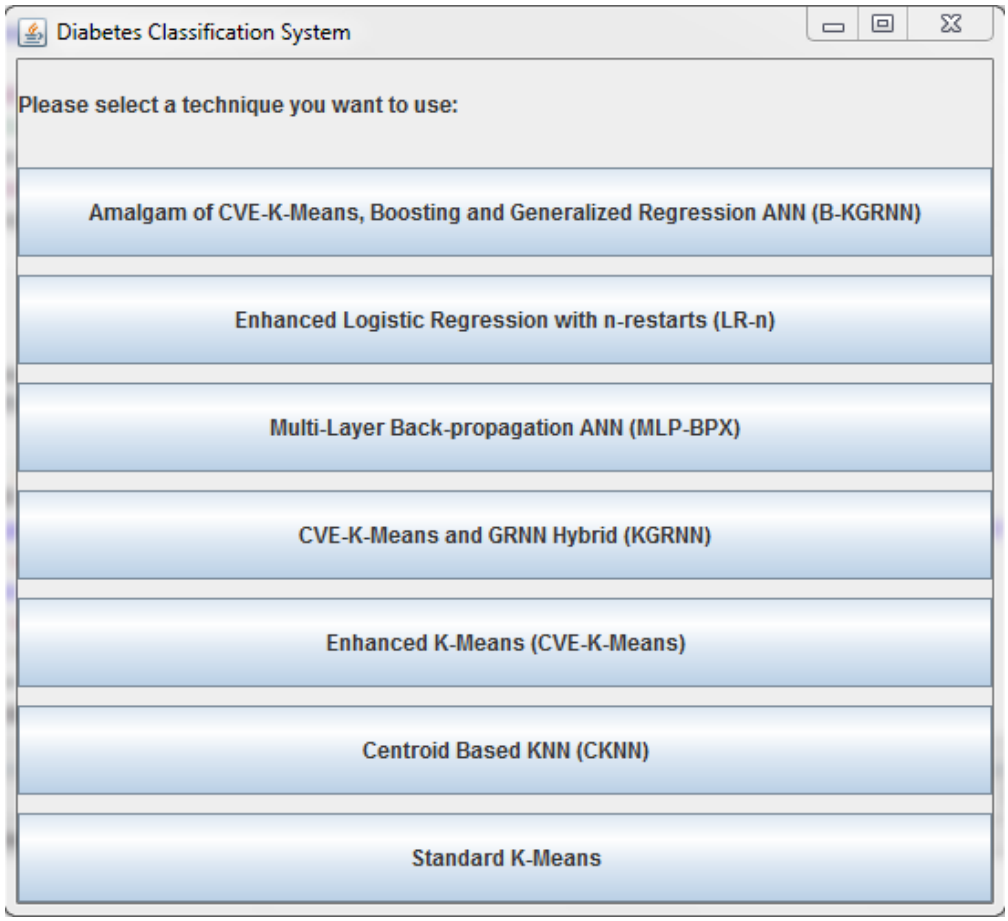

Figure B.1 Diabetes Classification System Main Window

B.0.3 Selecting ML technique and Uploading The Dataset

In order to run a desired ML technique, a user must first select an ML technique of choice by clicking a button on the main window. Upon the selection of a technique, a new window with options to upload a dataset, to select parameters, to start the algorithm, and visualizing the output of the algorithm will appear. The new window also has an option to view instructions on how to execute the desired algorithm. To choose a dataset, a user must select the "File" option at the top left corner of the new window then navigate to the desired folder and file then click on "Open" button as shown in figure B.2. All the datasets for this study are stored under Datasets folder on the files submitted with this dissertation.

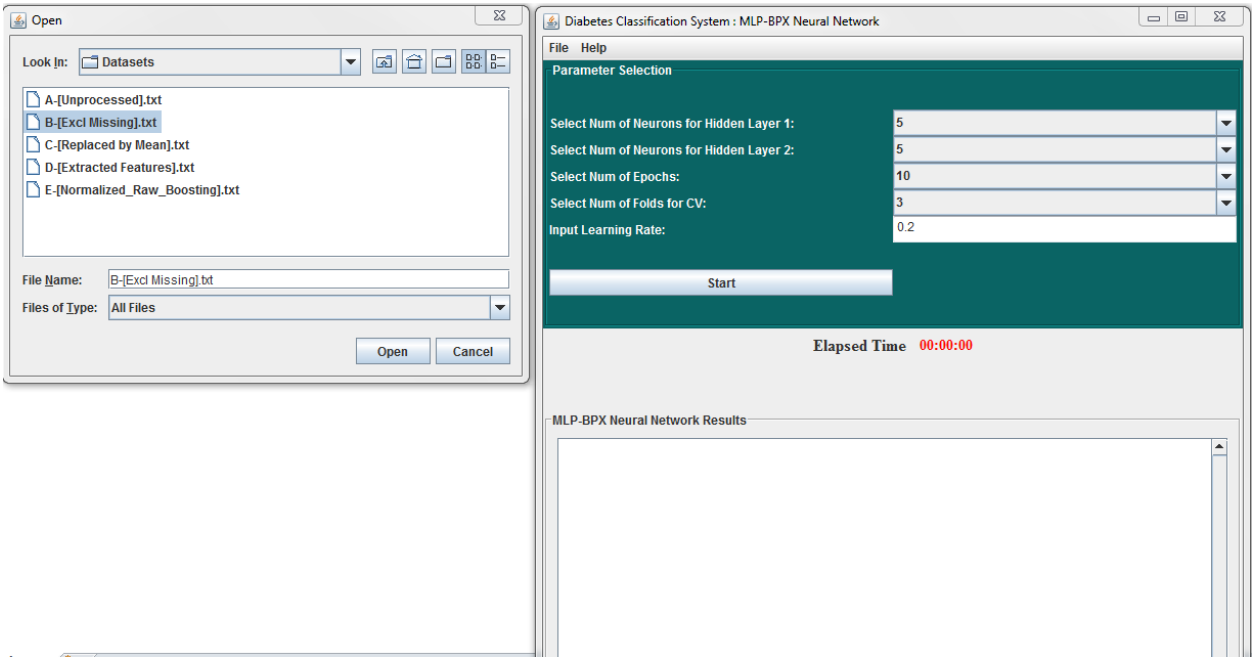

Figure B.2 Dataset Selection Demonstration

B.0.4 Selecting Parameters and Executing Selected Technique

One a user has uploaded a desired dataset, they must select or input all parameters necessary to run the algorithm. Once all parameters are selected, a user must click on "Start" button to initiate the algorithm. If one of the parameters is not entered correctly or a dataset is not selected before clicking the "Start" button, an error message will be displayed as shown in figure B.3 and figure B.4.

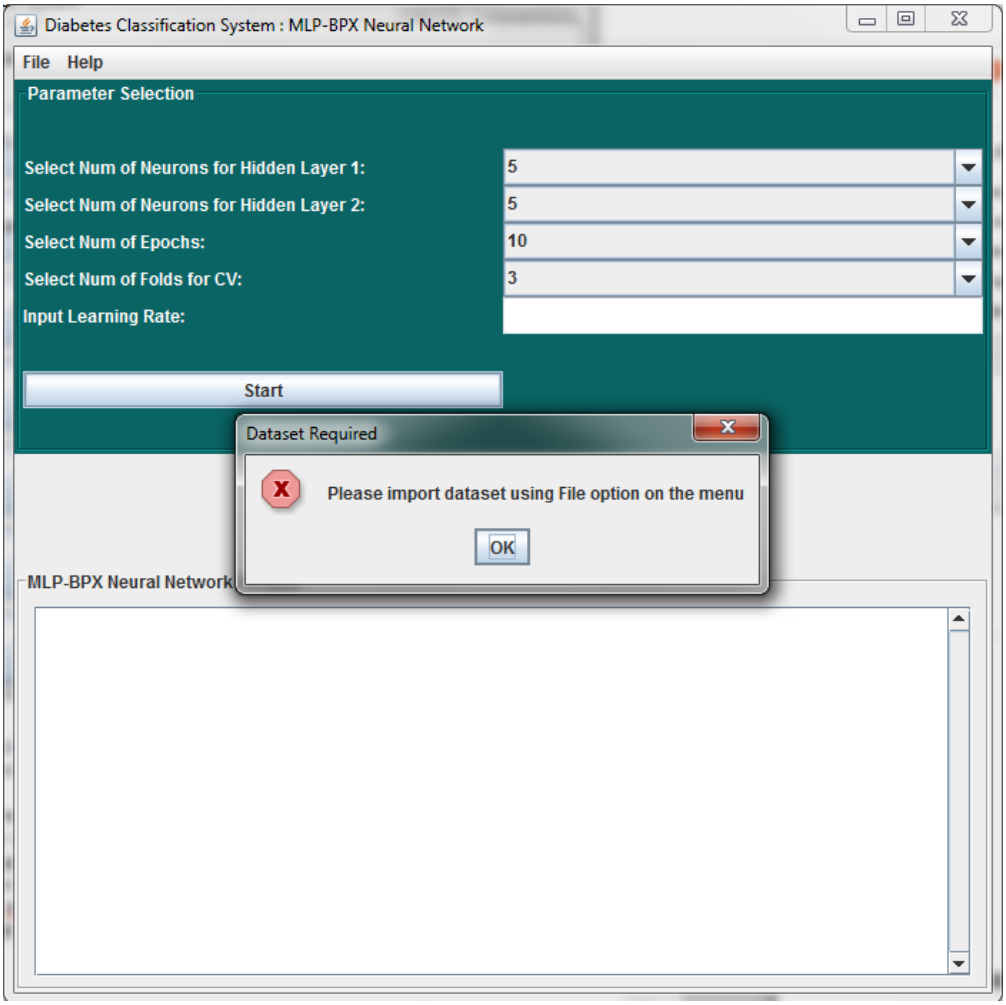

Figure B.3 Error Message for Missing Dataset

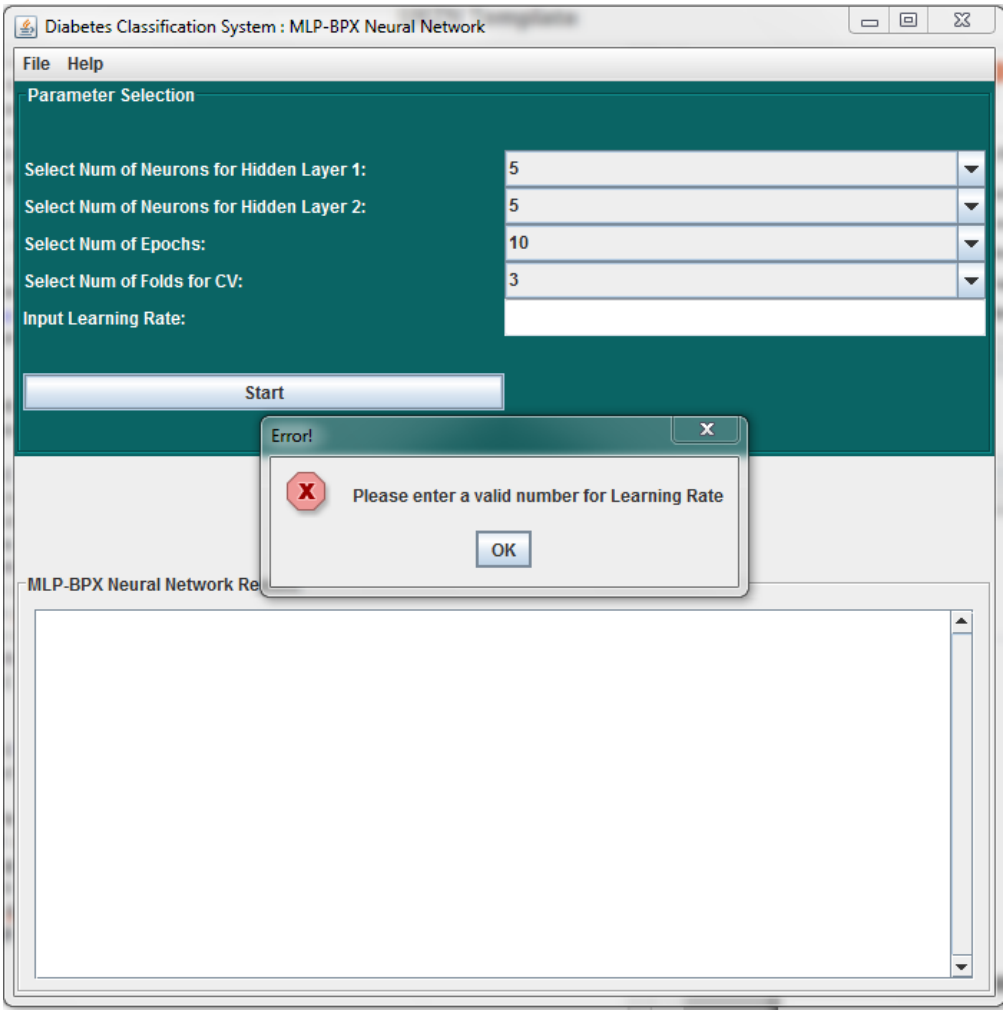

Figure B.4 Error Message for Invalid Parameter Input

B.0.5 ML technique Execution and Results

Upon starting the selected technique, a timer will be shown on the window opened in section **B.0.3** in order to constantly update the user about the execution progress of the selected technique as shown in figure B.5.

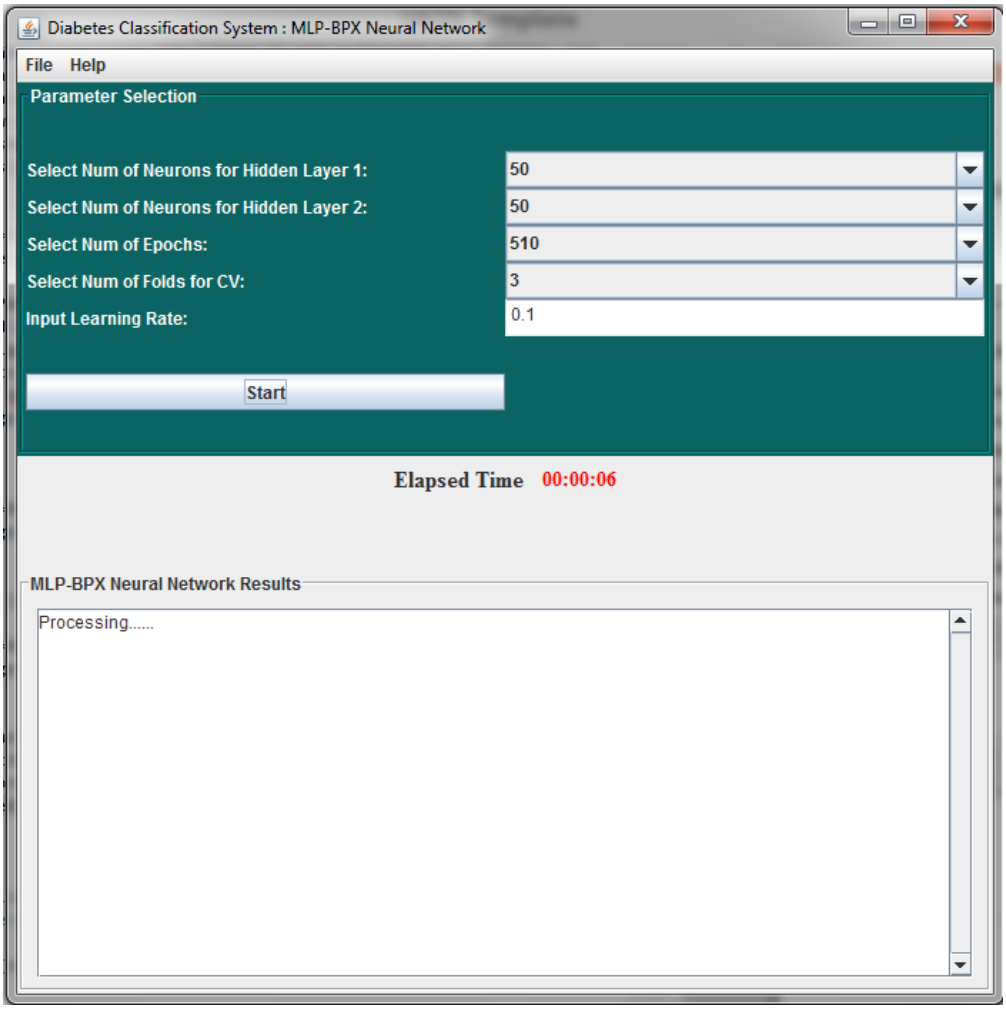

Figure B.5 Algorithm Execution Progress

Once the selected techniques finishes executing, the results are displayed at the bottom of the window with title “"Tehchnique Results"” as shown in figure B.6.

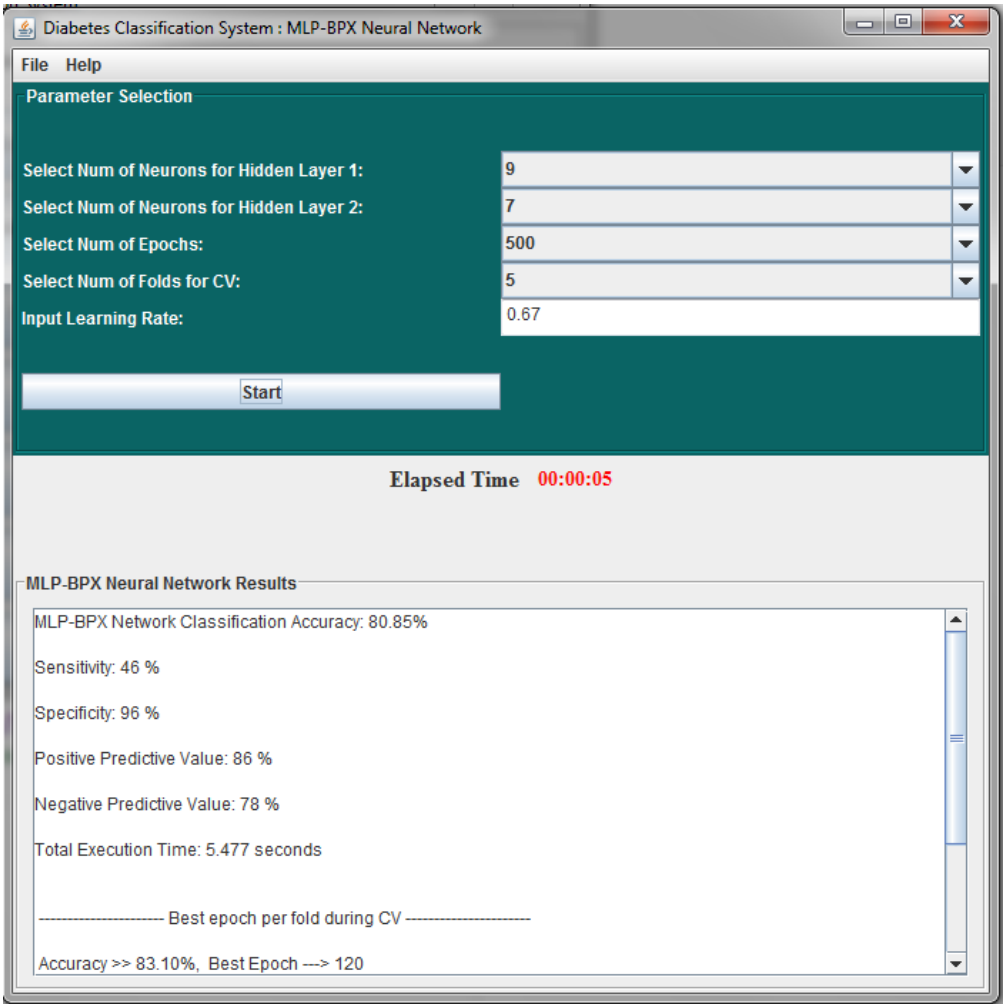

Figure B.6 Selected ML technique Results

After the results are displayed on the window, a user can copy and save the results including the parameters used for future use.

B.0.6 Getting Help

In order to get help on how to utilize the system, a user can click the "Help" option on the window that appears after selecting a desired ML technique from the main window. Upon clicking a sub-option of the help option, instructions will be displayed on the pop up window as shown in figure B.7. All the steps described in the preceding sections apply to all ML techniques proposed in this study.

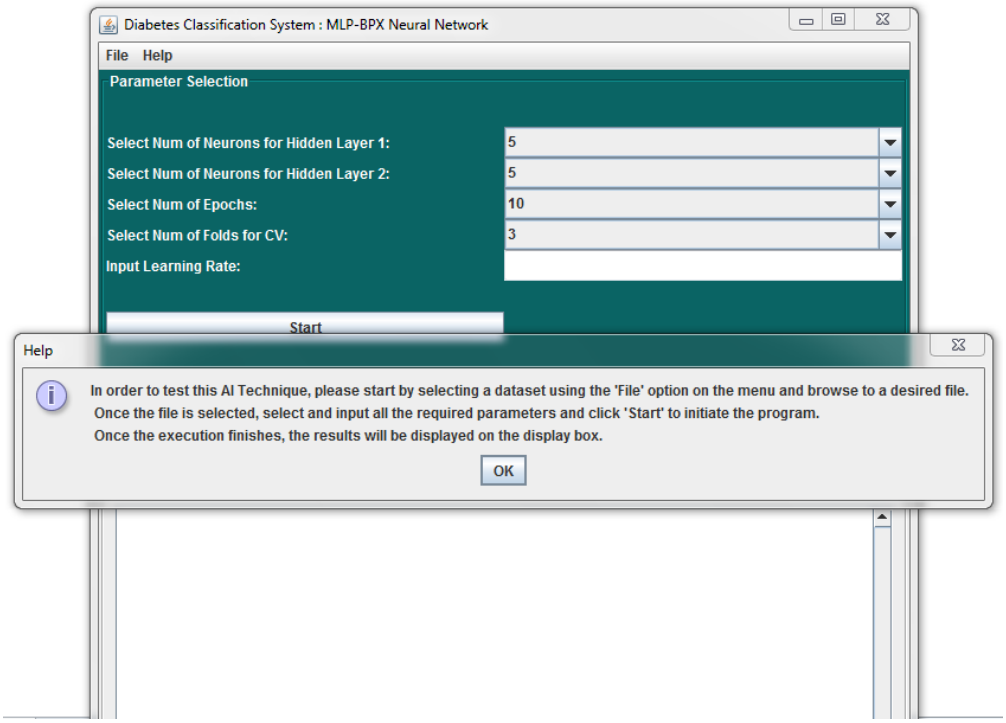

Figure B.7 Instructions for Executing Desired ML technique