Data Classification Using Various Learning Algorithms

Usman Ahmad Baba

A00018852



In partial fulfillment of the requirement for the award of degree of Master of Science (M.Sc.) in Computer Science submitted to

The School of Graduate Studies (SGS), American University of Nigeria.

FALL 2017

Certification

I certify that the work in this document has not been previously submitted for a degree nor has it been submitted as a part of a requirement for a degree except fully acknowledged within this text.

---------- ----------

Student Date

Usman Ahmad Baba

---------- ----------

Supervisor Date

Dr. Augustine Shey Nsang

---------- ----------

Dean SITC Date

Dr. Mathias Fonkam

---------- ----------

Dean SGS Date

Dr. Charles Nche

Dedication

With sincere modesty and genuine humility, I thank you O Allah for your aid and guidance. If it is worth dedicating, bless this piece with your approval, it is dedicated to You, O Allah, al-adzeem; The Great One; The Mighty; The One deserving the attributes of Extolment, Glory, and Purity from all imperfection.

Acknowledgement

In the name of Allah, the most gracious, the most merciful. All praise is for Allah, lord of the worlds. The most gracious, the most merciful. Master of the Day of Judgment. You (alone) we worship, and you (alone) we ask for help. Guide us to the straightway; the way of those upon whom you have bestowed grace, not of those who earned your anger, nor of those who went astray.

There is none whom I wish to thank more – in this world – than my parents, the two lighthouses from which I take my bearings; Prof. Ahmad Tela Baba and Hajiya Habiba Abubakar. I cannot thank you enough, for your erudite parental guidance which made me into the person I proudly am today, your invaluable support, advice, and for being the perfect parents I am extremely proud of having. My thanks as well to my sisters; Meenat and Nana, and my brothers; Musaddiq and Hafiz, for their unfailing support and motivation. May God almighty grant you all success in this world and entry into *Jannah* through the gate of your choice in the hereafter. Ameeen.

My sincere gratitude goes to an authority in dimensionality reduction, whose dimension in research I follow. The reservoir of knowledge, from whom I have tapped; my supervisor, Dr. Augustine Shey Nsang; whose verve of insights are endless. For his invaluable suggestions and contributions to the success of this thesis, I am grateful. Thank you for your expertise, quality reviews, and for providing me with the support, inspiration and courage to continue "the walk", amidst the confusion and frustrations I had on the path. May you continue to thrive in the path of success.

Finally, I cannot thank enough, for their unending support and generosity, all the people that have elevated from being just my friends to family, my friends, the community of scholars - from which I have learnt a lot, and all who have helped me in one way or the other, either directly or indirectly, to succeed. May God reward you abundantly.

Abstract

Dimensionality reduction provides a compact representation of an original high-dimensional data, which means the reduced data is free from any further processing and only the vital information is retained. For this reason, it is an invaluable preprocessing step before the application of many machine learning algorithms that perform poorly on high-dimensional data. In this thesis, the perceptron classification algorithm – an eager learner - is applied to three two-class datasets (Student, Weather and Ionosphere datasets). The k-Nearest Neighbors classification algorithm - a lazy learner - is also applied to the same two-class datasets. Each dataset is then reduced using fifteen different dimensionality reduction techniques. The perceptron and k-nearest neighbor classification algorithms are applied to each reduced set and the performance (evaluated using confusion matrix) of the dimensionality reduction techniques is compared on preserving the classification of a dataset by the k-nearest neighbors and perceptron classification algorithms. This investigation revealed that the dimensionality reduction techniques implemented in this thesis seem to perform much better at preserving K-Nearest Neighbor classification than they do at preserving the classification of the original datasets using the perceptron. In general, the dimensionality reduction techniques prove to be very efficient in preserving the classification of both the lazy and eager learners used for this investigation.

Keywords: Classification, confusion matrix, dimensionality reduction, eager learner, k-nearest neighbors, lazy learner, and the perceptron.

TABLE OF CONTENTS

COVER PAGE.....**Error! Bookmark not defined.**
Certification i
Dedication iii
Acknowledgement iv
Abstract v
TABLE OF CONTENTS vi
LIST OF TABLES viii
LIST OF FIGURES ix
CHAPTER ONE: INTRODUCTION 1
1.1 Background of the Study 1
1.2 Problem Statement 3
1.3 Aim and Objectives 4
1.4 Scope and Limitations 5
1.5 Thesis Structure 6
CHAPTER TWO: LITERATURE REVIEW 7
2.1 Dimensionality Reduction 7
2.2 Machine Learning 8
2.3 Dimensionality Reduction and Machine Learning 10
CHAPTER THREE: METHODOLOGY 12
3.1 Dimensionality Reduction Techniques 12
3.1.1 The New Random Approach 12
3.1.2 Modified New Random Approach 12
3.1.3 Singular Value Decomposition (SVD) 13
3.1.4 Principal Component Analysis (PCA) 14
3.1.5 The Variance Approach 14
3.1.6 The Combined Approach 15
3.1.7 The Direct Approach 16
3.1.8 Random projection (RP) 17
3.1.9 Bottom-Up Approach 17
3.1.10 Top-Down Approach 19
3.1.11 The New Top-Down Approach 19

3.1.12 The New Bottom- Approach 21
3.1.13 First Novel Approach 22
3.1.14 Second Novel Approach 24
3.1.15 Third Novel Approach 24
3.2 Classification Techniques 25
3.2.1 The Perceptron 25
3.2.2 K-Nearest Neighbor 28
3.3 Datasets 30
3.3.1 Ionosphere Dataset 30
3.3.2 Student Dataset 31
3.3.3 Weather Dataset 31
3.4 Confusion Matrix 31
CHAPTER FOUR: RESULTS AND DISCUSSION 35
4.1. The Perceptron Preservation 35
4.2 K-Nearest Neighbor Classification Preservation 44
CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATION 51
5.1 Summary 51
5.2 Conclusion 51
5.3 Recommendation 52
REFERENCES 53
Appendix A 56
Appendix B 71

LIST OF TABLES

Table 3.1: confusion Matrix…….. 32

Table 4.1: Comparing the reduction techniques for the perceptron classification preservation using the weather data set……..……..….35

Table 4.2: Comparing the reduction techniques for the perceptron classification preservation using the student data set……..……36

Table 4.3: Comparing the reduction techniques for the perceptron classification preservation using the ionosphere data set……..……..37

Table 4.4: Comparing the reduction techniques for K-Nearest Neighbor classification preservation using the weather data set.....45

Table 4.5: Comparing the reduction techniques for K-Nearest Neighbor classification preservation using the student data set……46

Table 4.6: Comparing the reduction techniques for K-Nearest Neighbor classification preservation using the ionosphere data set…….……47

LIST OF FIGURES

Figure 3.1 KNN example …..……. 29

Figure 4.1 Confusion matrix showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 18 attributes using direct approach……....38

Figure 4.2 Confusion matrix showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 18 attributes using new bottom up approach…….38

Figure 4.3 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 18 attributes using random approach…….……..39

Figure 4.4 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 18 attributes using third novel, new top down, modified new top down and random approaches……39

Figure 4.5 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 27 attributes using new top down, PCA, combined, direct and new random approaches…….……...40

Figure 4.6 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere dataset from 34 to 26 attributes using random approach……..……...40

Figure 4.7 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere dataset from 34 to 26 attributes using variance approach……..……...41

Figure 4.8 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere dataset from 34 to 26 attributes using PCA approach……..……..……41

Figure 4.9 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere dataset from 34 to 26 attributes using new bottom up approach……..……...42

Figure 4.10 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere dataset from 34 to 26 attributes using second novel approach……..……...42

Figure 4.11 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of weather data set from 30 to 26 attributes using PCA……48

Figure 4.12 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of ionosphere data set from 34 to 30 attributes using second novel approach…….48

Figure 4.13 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of ionosphere data set from 34 to 26, 28 and 30 attributes using new random approach……49

Figure 4.14 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of ionosphere data set from 34 to 26 and 28 attributes using new random projection……...49

Figure 4.15 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of ionosphere data set from 34 to 30and 32 attributes using new random projection……...50

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Data volumes and variety are increasing at an alarming rate making very tedious any attempt to glean useful information from these large data sets. Extracting or mining useful information and hidden patterns from the data is becoming more and more important but can be very challenging at the same time [1]. A lot of research done in domains like Biology, Astronomy, Engineering, Consumer Transactions and Agriculture, deal with extensive sets of observations daily. Traditional statistical techniques encounter some challenges in analyzing these datasets due to their large sizes. The biggest challenge is the number of variables (dimensions) associated with each observation. However, not all dimensions are required to understand the phenomenon under investigation in high-dimensional datasets; this means that reducing the dimension of the dataset can improve accuracy and efficiency of the analysis [2]. In other words, it is of great help if we can map a set of points, say n, in d-dimensional space into a p-dimensional space -where $p << d$-

so that the inherent properties of that set of points, such as their inter-point distances, their labels, etc., does not suffer great distortion. This process is known as Dimensionality reduction [3].

A lot of methods exist for reducing the dimensionality of data. There are two categories of these methods; in the first category, each attribute in the reduced dataset is a linear combination of the attributes of the original dataset. In the second category, the set of attributes in the reduced dataset is a subset of the set of attributes in the original dataset [4]. Techniques belonging to the first category include Random Projection (RP), Singular Value Decomposition (SVD), Principal Component Analysis (PCA), and so on; while techniques in the second category include but are not limited to the Combined Approach (CA), Direct Approach (DA), Variance Approach (Var),

New Top-Down Approach (NTDn), New Bottom-Up Approach (NBUp), New Top-Down Approach (modified version) and New Bottom-Up Approach (modified version) [5].

Dimensionality reduction provides a compact representation of an original high-dimensional data, which means the reduced data is free from any further processing and only the vital information is retained, so it can be used with many machine learning algorithms that perform poorly on high-dimensional data [6]. Calculation of inter-point distances is essential for many machine learning tasks and when the dimensionality increases, it has been proved that "the distance of a sample point to the nearest point becomes very similar to the distance of the sample point to the most distant point", thereby deteriorating the performance of machine learning algorithms [7]. Therefore, dimensionality reduction is an invaluable preprocessing step before the application of many machine learning algorithms.

Machine learning is a scientific field in which computer systems can automatically and intelligently learn their computation and improve on it through experience [8], [9]. Machine learning algorithms are of two main types: supervised learning algorithms and unsupervised learning algorithms. These algorithms have been used in solving a lot of complex real-world problems [10], [11]. In unsupervised learning, the set of observations are categorized into groups (clusters) basing the categorization on the similarity between them. This categorization is otherwise known as clustering [8]. Many clustering algorithms exist, among which k-means clustering is the most famous for a large number of observations [12].

Unlike clustering, classification is a supervised learning method in which the corresponding label for any valid input is predicted based on a number of training examples referred to as "training set," [8], [12]. Classification algorithms can further be categorized into eager and lazy learners, and this investigation considers one from each category. Eager learning algorithms attempt to

construct a general rule or create a generalization during the training phase which can further be used in classifying unseen instances [13]. Example of eager learners includes decision trees, support vector machine, and the perceptron.

The perceptron, an eager learner, is one of the earliest and simplest of all classification algorithms invented by Rosenblatt [14], basically used for classifying each point of a data set into either a positive or a negative label (1 or -1, good or bad, hot or cold, man or woman, etc.) [15]. It is interesting to know that in its basic form, it is still as valid as when it was first published [16].

On the other hand, a lazy learner delays any generalization or model construction until it is presented with an unseen instance to be classified [17]. This idea of not conducting any processing until a lazy learner is presented with an unseen instance makes the learner to require a lot of space in memory for storing the whole of the training instances and processing them each time it is presented with a new unseen instance. Example of a lazy learner is the k-nearest neighbor classifier [18]. In this algorithm, the result/label of any given instance is predicted based on the label most common to its k nearest neighbors, k, in this case, is a user-defined positive integer, normally with a small value [15].

1.2 Problem Statement

In data classification, the corresponding label (class) for any valid input is predicted based on a number of training examples referred to as "training set". This is achieved using a classifier model learning algorithm is applied to a training set made up of past examples having the same set of attributes with the unseen example [8], [12]. However, before starting the training, the label of each example in the "training set" is known [19].

To build a classifier model, an eager learner attempts to construct a general rule in the training phase which will subsequently be used in classifying unseen instances while a lazy learner delays the process until it is presented with an unseen instance [13]. The main disadvantage in eager learning is the long time which the learner takes in constructing the classification model but after the model is constructed, an eager learner is very fast in classifying unseen instances, while for a lazy learner, the disadvantage is the amount of space it consumes in memory and the time it takes during the classification [17]. This makes dimensionality reduction a very crucial preprocessing step because it facilitates classification, and compression of high-dimensional data and thus conserves memory and provides a compact representation of an original high-dimensional data [5].

Researches have been conducted on how dimensionality reduction techniques affect the performance of classifiers [20]–[22]. However, very little attention is given to the extent to which these reduction techniques facilitate and preserve classification. Therefore, this thesis attempts to advance the research by investigating the extent to which dimensionality reduction preserves the classification of weather dataset, student dataset and the ionosphere dataset obtained from "UCI machine learning repository", in order to fill the gap in literature and provide steps for further research in the area of machine learning.

1.3 Aim and Objectives

The aim of this research is to investigate the extent to which dimensionality reduction techniques preserve classification.

The objectives of the research are as follows:

1. Implementation of fifteen dimensionality reduction techniques and using these techniques to reduce the weather and student datasets, as well as the ionosphere dataset obtained from the *UCI machine learning repository* [23].

2. Implementation of the perceptron classification algorithm and using it to classify the data points of a two-class dataset. It shall also be applied to the datasets reduced from this two-class dataset using the techniques above, and comparisons will be made to determine the extent to which the reduction methods preserve the classification of the original dataset using the perceptron.

3. Implementation of the k-Nearest Neighbors classification algorithm and comparing the performance of the dimensionality reduction techniques on preserving the classification of a dataset by the k-nearest neighbors and perceptron classification algorithms.

4. Using *confusion matrices* to show the extent to which each dimensionality reduction method preserves classification of the original datasets and make comparisons with each other.

1.4 Scope and Limitations

This project is limited to showing the extent to which each of the dimensionality reduction methods implemented in this thesis preserves the classification of the original datasets by the perceptron and k-Nearest Neighbors classification algorithms. Accuracy will be used as the performance measure for showing the extent of the classification preservation, and this shall be obtained using confusion matrices.

1.5 Thesis Structure

This thesis consists of five chapters. Chapter 1 introduces dimensionality reduction and discusses its importance and applications to machine learning tasks. After presenting the problem to be addressed, the aim of the research is stated and the objectives are outlined.

Chapter 2 presents a review of literature related to dimensionality reduction and machine learning in general. Existing literature on single layer neural network is reviewed.

Chapter 3 describes the methodology used in this thesis. It discusses the methods in detail and explains how they are applied in achieving the objectives of the thesis. The results obtained from the methodology is presented and discussed fully in Chapter 4.

Chapter 5, which is the final chapter, provides a summary of the work and the results obtained in this thesis, concludes the research and also gives a possible recommendation for further research.

CHAPTER TWO

LITERATURE REVIEW

This chapter gives a review of literature related to dimensionality reduction, machine learning, and the application of dimensionality reduction in the machine learning domain with a bias towards the perceptron and K-Nearest Neighbors learning algorithms.

2.1 Dimensionality Reduction

Dimensionality reduction is defined as the mapping of high dimensional data to a low dimensional data, such that the result obtained by analyzing the reduced dataset is a good approximation of the result obtained by analyzing the original high dimensional data [24]. Due to the challenges faced in the analysis of the available large pool of data, there has been prevalence in emphasizing the robustness and importance of dimensionality reduction in literature.

The importance of dimensionality reduction is stressed in [25], where the authors proposed four novel dimensionality reduction techniques; *New Top-Down, New Bottom-Up, Variance – New Top-Down Hybrid, and Variance – New Bottom-Up Hybrid approaches* and used them alongside other existing techniques in reducing images. The authors observed that most of the approaches belonging to the first category (each attribute in the reduced dataset is a linear combination of the attributes of the original dataset) are inefficient in image preservation, while techniques belonging to the second category (the set of attributes in the reduced dataset is a subset of the set of attributes in the original dataset) are reasonably efficient in image preservation. After these observations, the authors proceeded by applying several queries on the reduced image to discover certain features of the original image. They observed that the features of the reduced image corresponds accurately to the attributes of the original image.

The authors of [1] identified some schemes used in reducing the number of features in high dimensional datasets to improve machine learning algorithms. The authors explained the concept of critical dimensions, which is the minimum number of features required for prediction (with high accuracy) in classification algorithms. They presented four dimensionality reduction schemes with their advantages and disadvantages. This provides researchers with the necessary information and direction when choosing a reduction scheme considering the dataset type.

In a recent study [5], the authors proposed a new approach to reducing the dimensionality of data. In this approach, for a reduction to m attributes from the original n attributes, m attributes are randomly selected from those n attributes to form the reduced dataset. This approach proved to be slightly better than some of the most popular dimensionality reduction techniques like Random Projection and Principal Component Analysis in the K-means Clustering preservation of the original datasets. Thus dimensionality reduction is a very fertile area of research and a strong tool for high-dimensional data preprocessing.

The authors in [26] proposed a biologically inspired (from the behavior of ants) dimensionality reduction method called Ant Colony Optimization-Selection algorithm. The authors used five microarray datasets which have a very high dimensionality to prove that the proposed algorithm selects more important genes from the high dimensional data based on some parameters with an excellent classification accuracy.

2.2 Machine Learning

Tom Mitchell in his classic book on machine learning says "*The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience*" [9]. To put it simply, machine learning is a scientific field in which computer

systems can automatically and intelligently learn their computation and improve on it through experience and can be classified into supervised and unsupervised learning algorithms. Machine learning techniques have been and are still used in solving a lot of complex real world problems. There are a lot of classification techniques. since no classifier is considered strictly better than the others [27], the perceptron and K-Nearest Neighbor classifiers are chosen for the purpose of this research.

The perceptron is an Artificial Neural Network that mimics or tries to simulate the activities of the brain with regards to information processing [14]. To do this, we take a weighted sum of inputs and if the sum is greater than some threshold value, it sends an output of one otherwise it sends zero (or -1). The perceptron is made up of a summation processor which takes the dot product of the inputs and the weights and then an activation function also known as threshold, which uses one step function to determine the output of the perceptron [16].

A great work in the domain of machine learning can be seen in [28], where the authors proposed a single layer Artificial Neural Network for the purpose of classifying cancer patients using Chebyshev, Trigonometric and Legendre expansion techniques. The authors concluded that Discrete Cosine Transform feature reduction based Neural Network classifiers perform better than the other classifiers used for the purpose of the research.

Another classification algorithm is the K-Nearest Neighbor algorithm. In this algorithm, the principle is that, the label of any given instance is predicted based on the label most common to its k nearest neighbors [15]. In a study [29], the authors proposed a new method for attribute selection, which selects attributes that complement each other and then tested it on a real dataset. The authors used two classes from the dataset and found that the proposed technique selects subsets of attributes that yields a classification accuracy which is higher than the accuracy obtained by using

the entire set of attributes or even the subset of attributes identified by CART. The classification technique used for the purpose of the investigation is the K-Nearest Neighbor classification algorithm, and a confusion matrix is used to measure the overall accuracy of the classifier.

2.3 Dimensionality Reduction and Machine Learning

Applications of Different dimensionality reduction techniques and machine learning have been explored by several researchers. The authors in [30] used several dimensionality reduction techniques to reduce text documents and then used supervised classification techniques on the original and reduced datasets after which the authors compared the results and justified that dimensionality reduction is a very reliable and important preprocessing stage for improving computational efficiency particularly for k-nearest neighbor classification, which significantly improves the computational complexity after application of dimension reduction.

A major contribution was made by [31] where the authors proposed an approach for dimensionality reduction and compared the proposed method with other existing dimensionality reduction methods on visualization, clustering and classification tasks in terms of accuracy. The authors used rand index as a measure for evaluating the accuracy of some clustering algorithms including k-means clustering and spectral clustering.

The authors in [32] used random projection (a dimensionality reduction technique) to investigate the effect of dimensionality reduction on performance by applying several machine learning techniques including the k-nearest neighbor classification to obtain a classifier for the detection of malicious executable files.

In another study [4], the authors applied dimensionality reduction on an artificial dataset to show the extent to which ensemble clustering on the reduced dataset agrees with the clusters in the

original dataset. In other words, the authors provided an understanding on the extent to which dimensionality reduction preserves clustering.

The authors of [6] also proposed a dimensionality reduction method and applied the method to reduce a publicly available dataset to build classification models by some supervised learning methods and also provided justification on how dimensionality reduction improves accuracy of classifiers.

An analysis was conducted by [33] on a publicly available dataset called chronic kidney disease dataset, obtained from the UCI machine learning repository [23]. The authors used supervised learning algorithms including support vector machine and the perceptron for prediction of chronic kidney diseases and justified that the perceptron gives better classification accuracy.

Another great work on machine learning in the literature is [20] in which the authors conducted a research on medical datasets and used clustering and dimensionality reduction for finding and imputing of missing values in medical records

In a recent study, the authors of [34] evaluated and analyzed (using confusion matrices) the effectiveness of dimensionality reduction techniques in improving the accuracy of classification in predicting the success of surgical operations.

The authors of [35] analyzed and used classification algorithms to predict liver disease. The authors compared the classification algorithms based on accuracy and execution time, then justified that the perceptron has the best classifier accuracy among the algorithms under study considering the liver disorder dataset.

CHAPTER THREE

METHODOLOGY

This chapter gives a detailed description of all the algorithms used in achieving the aim and objectives of this thesis. MATLAB programming is used for the analysis and implementations in this thesis.

3.1 Dimensionality Reduction Techniques

This section gives a description of the dimensionality reduction techniques implemented in this thesis.

3.1.1 The New Random Approach

This is a technique suggested by [5]. With this technique, to reduce a data set D of dimensionality d to one of dimensionality k, a set S_k is formed consisting of k numbers selected at random from the set S shown in equation (3.1).

$$S = \{x \in \mathrm{N} \mid 1 \leq x \leq d\} \tag{3.1}$$

Then, our reduced set, D_R, is shown in equation (3.2).

$$D_R = D(:, S_k) \tag{3.2}$$

That is, D_R is a data set having the same number of rows as D, and if A_i is the i^{th} attribute of D_R, then A_i will be the j^{th} attribute of D if j is the i^{th} element of S_k.

3.1.2 Modified New Random Approach

This technique is a modification of the new random approach proposed by [36]. To reduce a data set D of dimensionality p to one of dimensionality k, with this improved approach, a more efficient

method is utilized in generating the random numbers, i.e. the results are less random. The algorithm is given in Algorithm 3.1.

Algorithm: Modified New Random Approach

Input: *D* (original dataset)

Output: *Dr* (reduced dataset)

M = []

for i = 1 to m do

- Run the New Random Approach to generate k numbers at random in the range 1..p
- Store the list of numbers generated as the ith row of M

end

Generate the one-dimensional matrix M1 with p entries such that M1[p] holds the frequency of the number p in the matrix M

Generate the matrix Result which contains the k entries in M of highest frequency, arranged in ascending order

DR = D(:, Result)

Algorithm 3.1: Modified New Random Approach

3.1.3 Singular Value Decomposition (SVD)

According to [37], *SVD* reduces a data set *D (*given as an n*p matrix) to a data set X *(*with dimensionality q) by computing the singular value decomposition of X, that is, it computes matrices U, S and V as shown in equation (3.3).

$$D = USV^T \tag{3.3}$$

U is an n*n orthogonal matrix whose columns are the left singular vectors of X,

V is a p*p orthogonal matrix whose columns are right singular vectors of X, and

S is a n*p diagonal matrix whose diagonal elements which are the singular values of X.

Then the transformed matrix is computed using equation (3.4). Where U_q is an n*q matrix whose columns are the unit vectors corresponding to the q largest left singular values of D . *SVD* can only work on square matrices.

$$X = D^T U_q, \tag{3.4}$$

3.1.4 Principal Component Analysis (PCA)

PCA follows a similar approach to Singular Value Decomposition. As for *SVD,* to reduce a dataset, *D,* using *PCA:*

We find the *SVD* of D (by decomposing it into three submatrices *U, S* and *V)* as given in equation (3.3). The transformed matrix is computed from equation (3.5) where V_q is a V_q is a p*q matrix whose columns are the unit vectors corresponding to the q largest right singular values of *D* [37].

$$X = D^T V_q, \tag{3.5}$$

Note: Unlike *SVD, PCA* can be used to reduce any *n x m* matrix.

3.1.5 The Variance Approach

As explained by [38], With the Variance approach, to reduce a dataset D to a data set DR, we start with an empty set, I, and then add dimensions of D to this set in decreasing order of their variances. That means that a set *I* of *r* dimensions will contain the dimensions of top r variances.

Thus, let $I_r = \{i_1, \ldots, i_r\} \subset \{1, \ldots, n\}$, the collection of dimensions corresponding to the top *r* variances. That is i_1 denotes the dimension of largest variance, i_2 the dimension of next larger

variance, etc. The reduced database, D_R, in equation (3.6) is obtained by extracting the data corresponding to the selected dimensions.

$$D_R = D(:, I_r) \tag{3.6}$$

Where D_R has the same number of rows as D and r columns: the i^{th} column of D_R is the column of the original database with the i^{th} largest variance .

3.1.6 The Combined Approach

According to [38], like the previous approach, the *Combined* Approach is one approach which reduces a dataset *D* to a subset of the original attribute set. To reduce a dataset D_{nxp} to a dataset containing *k* columns, the *Combined Approach* selects the combination of *k* attributes which best preserve the interpoint distances, and reduces the dataset to a dataset containing only those *k* attributes. To do so, it first determines the extent to which each attribute preserves the interpoint distances. In other words, for each attribute, *x,* in *D,* it computes g_xm and g_xM given by equation (3.7) and equation (3.8) respectively.

$$g_xm = \min\left\{\frac{\| f(u) - f(v) \|^2}{\| u - v \|^2}\right\} \tag{3.7}$$

$$g_xM = \max\left\{\frac{\| f(u) - f(v) \|^2}{\| u - v \|^2}\right\} \tag{3.8}$$

where u and v are any two rows of *D*, and f(u) and f(v) are the corresponding rows in the dataset reduced to the single attribute *x.* The average distance preservation for the attribute *x* is then computed using equation (3.9).

$$g_xmid = (g_xm + g_xM)/2 \tag{3.9}$$

To reduce the dataset *D* from *p* columns to *k* columns, this approach then finds the combination of *k* attributes whose average value of $g_x$mid is maximum .

3.1.7 The Direct Approach

The authors of [38] came up with the direct approach which is similar to the *Combined Approach.* According to the authors, to reduce a dataset D_{nxp} to a dataset containing *k* columns, the *Direct Approach* selects the combination of *k* attributes which best preserve the interpoint distances, and reduces the original dataset to a dataset containing only those *k* attributes. To do so, it first generates all possible combinations of *k* attributes from the original *p* attributes. Then, for each combination, *C,* it computes g_cm and g_cM given by equation (3.10) and equation (3.11) respectively.

$$g_c m = \min\left\{ \frac{\| f(u) - f(v) \|^2}{\| u - v \|^2} \right\} \tag{3.10}$$

$$g_c M = \max\left\{ \frac{\| f(u) - f(v) \|^2}{\| u - v \|^2} \right\} \tag{3.11}$$

Where *u* and *v* are any two rows of *D*, and *f(u)* and *f(v)* are the corresponding rows in the dataset reduced to the attributes in C. The average distance preservation for this combination of attributes is then computed using equation (3.12).

$$g_c mid = (g_c m + g_c M)/2 \tag{3.12}$$

As we can see, the difference between the *Combined* and *Direct* Approaches is that for the *Combined Approach,* we first find the average distance preservation for each attribute, and then, for any combination of attributes, we compute its average distance preservation by finding the averages of the distance preservations of the individual attributes. With the *Direct Approach,* on the other hand, to find the average distance preservation for any combination of attributes, *C*, we

reduce the original dataset directly to the dataset containing only the attributes in *C*, and then compute the average distance preservation for this combination using the formulas above

3.1.8 Random projection (RP)

With RP, a given dataset with *d* dimensions is projected onto a lower-dimensional subspace of *k*-dimensions using a random *d*k* matrix *R* whose columns have unit lengths [37].

For instance, assuming D_{n*d} is the given dataset with d-dimensions, then the reduced *k*-dimensional dataset, *X,* is obtained as shown in equation (3.13).

$$X_{nxk}=D_{nxd}*R_{d*k} \tag{3.13}$$

3.1.9 Bottom-Up Approach

This approach, proposed by [24], works by selecting subsets of attributes increased by one attribute at a time. With this technique, assuming we want to reduce a data set of p dimensions to another data set of m dimensions, the process is started with a subset S1, containing a single attribute, say *y,* from the original data set, which best preserves k-means clustering. It then increases to S2, which contains a total of two attributes including y that best preserves k-means clustering. S2 is then increased to another subset S3 that contains three attributes (the two attributes of S2 and another attribute from the original dataset apart from the two attributes of S2) which best preserves k-means clustering. This process continues until S_m (the subset that has the m attributes of the original dataset which best preserves k-means clustering) is obtained. The algorithm is shown in Algorithm 3.2.

Algorithm: Bottom-Up Approach

Input: D (a data set with n rows and p columns)

Output: Dr (a data set with n rows and m columns, $m < p$)

Uses: *Rand*: a clustering performance measure

LC: a list of combinations of attributes of D

AML: a list of numbers; $AML(i)$ is the extent (as a percentage) to which the i*th* combination in LC preserves k-means clustering.

Lo: the combination in LC which best preserves k-means clustering

1. Perform a k-means clustering of D, and store the result in R,

2. Compute AML: $AML(i)$, $i = 1, 2, 3, \ldots, p$, represents the extent to which the i*th* attribute of D preserves the k-means clustering of D (using *Rand*)

3. Find the maximum element of AML, and thus find x, the attribute of D which best preserves the k-means clustering of D.

4. Take $Lo = [x]$, and $t = 1$.

5. While $t < m$:

(a) Generate all $p - t$ combinations of $t + 1$ attributes of D which include the t attributes in Lo. Let LC be the list containing these $p - t$ combinations.

(b) Redefine AML to be the list containing $p - t$ values such that $AML(i)$ represents the extent to which the i*th* combination of attributes in LC preserves k-means clustering (by the *Rand* index)

(c) Find the maximum value of AML, and thus the combination, L, of attributes of D (in LC) that best preserves k-means clustering (by the *Rand* index)

(d) Increase t by 1, and redefine Lo to be the list of attributes in L.

End while;

6. The result, Dr, is given by: $Dr = D(:, Lo)$

Algorithm 3.2: Bottom-Up Approach

3.1.10 Top-Down Approach

This approach is also suggested by [24], it follows almost the same process with the Bottom-up approach discussed above. However, instead of considering the subset of attributes increased by one attribute at a time, the Top-down approach considers the subset of attributes decreased by one attribute at a time. Assuming we want to reduce a data set with p dimensions to one with m dimensions, the Top-Down approach starts by reducing the original dataset to the subset of p-1 attributes which best preserves k-means clustering, then to the subset of p-2 attributes which best preserve k-means clustering. The procedure continues until the subset of m attributes that best preserve k-means clustering of the original data set is obtained. The algorithm is shown in Algorithm 3.3.

3.1.11 The New Top-Down Approach

This technique is proposed in [25], it is a modification of Top-Down approach. In this technique, assuming we want to reduce a data set with p dimensions to one with m dimensions, the process is started by a reduction to the subset of p-1 attributes which best preserves the interpoint distances (instead of k-means clustering as in Top-Down approach described above), then to the subset of p-2 attributes which best preserve interpoint distances, the process continues till the subset of m attributes that best preserve the interpoint distances of the original data set is obtained. The algorithm is shown in Algorithm 3.4.

- **Algorithm: Top-Down Approach**
- **Input:** D (a data set with n rows and p columns)
- **Output:** $D1$ (a data set with n rows and m columns, $m < p$)
- **Uses:** *Rand:* a clustering performance measure

  LC: a list of combinations of attributes of D

  AML: a list of numbers; $AML(i)$ is the extent (as a percentage) to which the *ith* combination in LC preserves k-means clustering.

  Lo: the combination in LC which best preserves k-means clustering

  1. Perform a k-means clustering of D, and store the result in R: R has all the columns of D with an extra column showing the cluster to which each observation in D has been assigned.

  2. Generate LC, the list of the p combinations of p - 1 attributes of D

  3. Assign to t the value of p - 1.

  4. While $t \geq m$:

  (a) Compute, using *Rand*, the extent to which each combination in LC preserves k-means clustering. Store the result in a new list, AML.

  (b) Find the maximum element of AML, and thus the combination, Lo, of t attributes of D which best preserves k-means clustering.

  (c) Reset LC to be the list of the t combinations of t - 1 attributes of Lo

  (d) Reset $t \leftarrow t$ - 1

  Endwhile;

  5. The result, $D1$, is given by: $D1 = D(:, Lo)$

Algorithm 3.3: Top-Down Approach

Algorithm: The New Top-Down Approach

Input: D (a data set with n rows and p columns)

Output: Dr (a data set with n rows and m columns, m < p)

1. Generate LC, the list of the p combinations of p – 1 attributes of D.

2. Assign to t the value of p – 1.

3. While t ≥ m:

(a) Compute the extent to which each combination of attributes in LC preserves the interpoint distances of D. Store the result in a new list, AML

(b) Find the maximum element of AML, and thus the combination L0, of t attributes of D which best preserves the interpoint distances

(c) Reset LC to be the list of the t combinations of t – 1 attributes of L0

(d) Reset t ← t -1

Endwhile;

4. The result, Dr, is given by: Dr = D(:, L0)

Algorithm 3.4: The New Top-Down Approach

3.1.12 The New Bottom- Approach

The New Bottom-Up approach is a modification of Bottom-Up approach [25]. In this approach, suppose we want to reduce a data set of p dimensions to another data set of m dimensions, the process is started with a subset S1, containing a single attribute, say *y,* from the original data set,

which best preserves the interpoint distances (instead of k-means clustering as in Bottom-Up approach described above), it then increases to S2, which contains a total of two attributes including y, that best preserves the interpoint distances. S2 is then increased to another subset S3 that contains three attributes (the two attributes of S2 and another attribute from the original dataset apart from the two attributes of S2) which best preserves the interpoint distances. This process continues until S_m (the subset that has the m attributes of the original dataset which best preserves the interpoint distances) is obtained. The algorithm is shown in Algorithm 3.5.

3.1.13 First Novel Approach

Suppose we want to reduce a data set of *d* dimensions to another data set of *y* dimensions, the first novel approach finds the extent to which interpoint distances is preserved by each attribute, say x, in the original data set. To do so, $g_x m$ and $g_x M$ are computed as in equation (3.14) and equation (3.15).

$$g_x M = \max\{\frac{||f(u)-f(v)||^2}{\{||u-v||^2}\} \tag{3.14}$$

$$g_x m = \min\{\frac{||f(u)-f(v)||^2}{||u-v||^2}\} \tag{3.15}$$

where

u = a row in the original dataset

v = another row in the original dataset

f(u) = the row in the reduced dataset corresponding to u from the original dataset

Algorithm: The New Bottom-Up Approach

Input: D (a data set with n rows and p columns)

Output: Dr (a data set with n rows and m columns, m < p)

1. Compute the vector AML such that AML(i) represents the extent to which the ith attribute of D preserves the interpoint distances of D

2. Find the maximum element of AML, and thus find x, the attribute of D which best preserves the interpoint distances of D

3. Take L0 = [x], and t = 1.

4. While t < m:

(a) Generate all p – t combinations of t + 1 attributes of D which include the t attributes in L0. Let LC be the list containing these p – t combinations

(b) Redefine AML to be the list containing p – t values such that AML(i) represents the extent to which the ith combination of attributes in LC preserves the interpoint distances of D

(c) Find the maximum value of AML, and thus the combination, L, of attributes of D (in LC) which best preserves the interpoint distances of D

(d) Increase t by 1, and redefine L0 to be the list of attributes in L

Endwhile;

5. The result, Dr, is given by: Dr = D(:, L0)

Algorithm 3.5: The New Bottom-Up Approach

f(v) = the row in the reduced dataset corresponding to v in the original dataset

The average of $g_x m$ and $g_x M$ is then computed using equation (3.16).

$$g_x mid = \frac{(g_x m + g_x M)}{2} \tag{3.16}$$

Then to reduce a dataset from the original *d* dimensions to *y* dimensions, the *y* attributes with the largest $g_x mid$ value from the original dataset are selected.

3.1.14 Second Novel Approach

In this technique, just like the first novel approach, the interpoint distance preservation is computed. However, in this approach, for each attribute, *x*, adp_x (the actual extent to which it preserves the interpoint distances) is computed as in equation (3.17).

$$adp_x = \frac{\sum_{u=1}^{n} \sum_{v=u+1}^{n} \frac{||f(u)-f(v)||^2}{||u-v||^2}}{n_r} \tag{3.17}$$

n_r= number of pairs of rows in the original dataset. Formally represented as in equation (3.18).

$$n_r = {}^{n}C_r = \frac{n(n-1)}{2} \tag{3.18}$$

Then, to reduce a dataset from its original *d* dimensions to *y* dimensions, the *y* attributes with the largest adp_x value are selected.

3.1.15 Third Novel Approach

In this approach, to reduce a dataset from its original *d* dimensions to *y* dimensions, after computing the extent to which each attribute in the original dataset preserves k-means clustering, the *y* attributes which best preserve k-means clustering are selected.

3.2 Classification Techniques

This section gives a description of the classification techniques implemented for the purpose of this investigation.

3.2.1 The Perceptron

The perceptron [14], is a supervised learning algorithm used for classifying each point of a data set into either a positive or a negative label [15]. Basically, the perceptron takes a weighted sum of observations (real values) and if the sum is greater than some threshold value, it sends an output of one otherwise it sends zero (or -1) [16]. Unfortunately, in some cases, it takes a long time to train the perceptron because of the process of adjusting the weights until all observations are correctly classified. However, after training, the algorithm is very efficient in using the weights obtained for classification of unseen instances [39].

The perceptron is made up of a summation processor which takes the dot product of the inputs and the weights and then an activation function also known as threshold, which uses one step function (shown in equation (3.19)) to determine the output of the perceptron. Learning by the perceptron is completed when it happens that no error has occurred after an epoch (a complete pass through the training set) during the training phase [16]. When the training is complete, the perceptron will respond, for any input presented to it, with an output that is the same as the output of the observation used in the training phase.

$$(f(x) = \begin{cases} -1 \; if \;\; w.x < 0 \\ 1 \quad if \;\; w.x \geq 0 \end{cases} \tag{3.19}$$

The perceptron algorithm is depicted in Algorithm 3.6.

Algorithm: The Perceptron

PerceptronLearning $[M_+, M_-]$

w = *arbitrary vector of real numbers*

Repeat

For all $x \in M_+$

If $w.x \leq 0$ *then* $w = w + x$

For all $x \in M_-$

If $wx > 0$ *then* $w = w - x$

Until all $x \in M_+ \cup M_-$ *are correctly classified*

Algorithm 3.6: The Perceptron

Algorithm 3.2.1 is implemented and a function called *tclassify* (of which the code is provided in the appendix) is implemented to test the perceptron after the training phase is completed. This is illustrated in example 3.1.

Example 3.1 Consider the dataset, D, given in the form of a matrix; the last column represents the class (label) of each observation (row).

$$D = \begin{bmatrix} 0 & 1.8 & 1 \\ -1.2 & 1.4 & -1 \\ 2 & 0.6 & 1 \\ 0.4 & -1 & -1 \end{bmatrix}$$

In the first iteration, the weight vector is initialized as **w** = [1, 1]. Taking the dot product of the weight vector and the first input, x, which is (0, 1.8), gives a value greater than zero (1.8) and the label assigned to that input is a positive label (1), therefore, the input (0, 1.8) is correctly classified as positive.

The process is then continued by taking the dot product of the weight vector and the next input (-.12, 1.4) which yields a positive value (0.2). This means that the input is incorrectly classified as having a positive label. It is given in the perceptron algorithm above that, for all inputs belonging to the negative class, if the dot product of the weight vector and that input is greater than zero (say 0.2 as in the case of the input (-1.2, 1.4) above), then, the weight vector, w, will be updated to $w = w - x$, i.e.

$$w = (1, 1) - (-1.2, 1.4)$$

Which yields w= (2.2, 0.4). The updated weight vector is now taken as (2.2, 0.4) and if the dot product of the weight vector and the input (-1.2, 1.4) is taken then it will obviously correctly classify the input as negative. The process continues with the iteration through the loop (which in this case is five iterations) of the learning algorithm until the final weight vector, in this case (2.2, 1.2), which correctly classifies all given instances, is obtained. After the completion of this process,

training is said to be completed and the perceptron, built using the training set, can now be used to classify any similar previously unseen observation.

3.2.2 K-Nearest Neighbor

K-Nearest Neighbor is a supervised learning algorithm and also one of the simplest machine learning algorithms [24], [40] . In this classification technique, the result/label of any given instance is predicted based on the label most common to its k nearest neighbors. K in this case is a user-defined positive integer, normally with a small value [15]. Example 3.2 explains the concept of k-Nearest neighbor. The principle here is that, all observations that are close will have the same label.

K-Nearest Neighbor classifier measures the distance between the test data and each observation in the training set to determine the observations in the training set that are closest to the test data. For numerical data, which is the type of data used in this investigation, Euclidean distance is the most widely used distance metric. It performs relatively better than the cosine and Minkowsky distance metrics in K-Nearest Neighbor classification algorithm [41]. Therefore, Euclidean distance is considered the distance metric of choice for this investigation.

K-Nearest Neighbor is described by Algorithm 3.7

Note: k in algorithm 3.7 is user-defined.

Algorithm: K-Nearest Neighbor

Input: D (a training data set)

D1 (a data point or a set of data points whose classes are to be predicted)

Output: D2 (a data set containing the test set and their corresponding predicted labels)

1. Compute the similarity between an observation in the test set and each observation in the training set.
2. Select the k nearest neighbors of an observations among the training samples
3. Assign to the test data the class which is most common among its k nearest neighbors.

Algorithm 3.7: K-Nearest Neighbor

Example 3.2

In Figure 3.1, Suppose the boxes are considered to be data points belonging to a single class, say **Box**, the triangles belonging to another class, say **triangle**, and we want to predict, to which class (either box or triangle) the circle belongs. Assuming k is three (i.e. we are going to assign a class to the circle based on its 3 nearest neighbors) then, the circle will be classified as **triangle** because its closest neighbors in the inner circle are 2 triangles and 1 box**.**

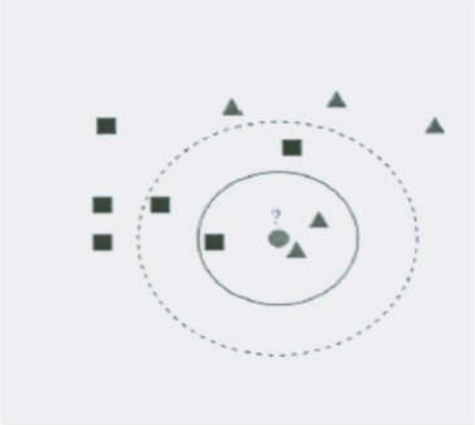


Figure 3.1 KNN example

Similarly, if k = 5 then it will be classified as a box because its 5 nearest neighbors are 3 boxes and 2 triangles inside the outer circle.

3.3 Datasets

A lot of research, especially research in the domain of data mining, machine learning, and artificial intelligence, is conducted using online datasets, extracts from online datasets and some using artificial datasets, which is a common practice in the domain of machine learning [1], [5], [13], [24], [33], [42], [43]. To achieve the objectives of this investigation, three datasets are used, namely, Student dataset, Weather dataset, and Ionosphere dataset. The three datasets have different characteristics but the values are all numeric.

3.3.1 Ionosphere Dataset

The Ionosphere dataset is a publicly available dataset obtained online from the "UCI machine learning repository". The observations are categorized into two classes, either "Good" or "Bad". The original ionosphere dataset contains 34 attributes and 351 observations. For this investigation, due to the time it takes to train the perceptron, 100 different observations are selected at random from the original ionosphere dataset, with almost a ratio of 50:50 for Good and Bad labels. The labels are renamed as 1 and -1 (for Good and Bad respectively) for use with the perceptron, and renamed to 1 and 2 (for Good and Bad respectively) for use with the K-Nearest Neighbor classifier.

The observations are split in the ratio of 70:30 for training and testing sets respectively, in both the perceptron and K-Nearest Neighbors. The training set contains 36 good examples and 34 bad examples while the test set contains 18 good examples and 12 bad examples.

3.3.2 Student Dataset

The student dataset is a dataset (obtained from MATLAB and modified) that models the score/ exam grades of students and classify them as either brilliant or dull, for this study, the classes are renamed to 1 for brilliant and -1 for dull. The dataset contains the scores of 32 students and 15 attributes or courses. 20 observations are used for training and 12 are used for testing in both the perceptron and K-Nearest Neighbors classification algorithms.

3.3.3 Weather Dataset

The weather dataset is an artificial dataset presumed to be the weather readings of Yola town in Adamawa state, Nigeria. It contains weather readings for 50 months, and each month is considered to have 30 days, i.e. the dataset has 50 observations and 30 attributes. Each observation is classified as either Cold or Warm (1 or -1 respectively). 35 observations are used for training and 15 for testing in both the perceptron and K-Nearest Neighbors.

3.4 Confusion Matrix

A confusion matrix is a table containing the description of the actual and predicted classifications performed by a classifier. It is a widely used and effective metric for evaluating the performance of classifiers [29]. The data contained in the matrix is used in evaluating the performance of classifiers [44]. In the context of this study, the four parameters that constitute a confusion matrix has the following meaning:

a is the number of negative observations correctly predicted as negative

b is the number of positive observations predicted as negative

c is the number of negative observations predicted as positive and

d is the number of positive observations correctly predicted as positive.

Table 3.4 is commonly used to show the confusion matrix of a classifier model having two classes.

According to [45], some of the evaluation measures that can be obtained from the confusion matrix include:

Accuracy (AC): this is a measure of the number of correct predictions made by the classifier and is given by

$$AC = \frac{a + d}{a + b + c + d}$$

CONFUSION MATRIX		Predicted	
		Negative	Positive
Actual	Negative	**a**	**b**
	Positive	**c**	**d**

Table 3.1: confusion Matrix

True Positive rate (TP)/recall: This is a measure of positive instances correctly classified as positive and can be calculated as:

$$TP = \frac{d}{c + d}$$

True Negative rate (TN): This is a measure of negative instances correctly predicted as negative and can be calculated as:

$$TN = \frac{a}{a + b}$$

False Positive rate (FP): This is a measure of negative instances incorrectly predicted as positive and can be calculated as:

$$FP = \frac{b}{a + b}$$

False Negative rate (FN): This is a measure of positive instances incorrectly predicted as positive and can be calculated as:

$$FN = \frac{c}{c + d}$$

Precision (P): This is a measure of correctly classified positive instances and can be calculated as:

$$P = \frac{d}{b + d}$$

In this investigation, after partitioning a dataset, D, into training and test sets, a perceptron is built on the training set, and the weight vector obtained from the training phase is then used to classify the data points of the test set and the result of the classification is stored as **Result1**. The original dataset, D, is then reduced to various numbers of attributes (the number of reduced sets depends on the dataset used) using each of the fifteen dimensionality reduction techniques described above. A perceptron is also built for each of the reduced datasets (using the same size of training and test sets as in the original dataset, D) and then the weight vector obtained during the training phase of

each of the reduced set is used to classify the data points of the corresponding test set. The result of the classification is then saved as **Result2.**

A confusion matrix is then used for the comparison of **Result1** and **Result2** to see the extent to which dimensionality reduction techniques preserve classification using the perceptron, considering the predicted labels of the original data set, D, as the basis for the evaluation.

As regards to K-Nearest Neighbors classification, all the labels of the negative classes (-1) from the original datasets are renamed to a class label "2", where 2 in the ionosphere dataset represents the class "bad", in students dataset it means "dull" and in weather dataset it means "warm". As it was done with the perceptron above, after obtaining the result of the classification from the original test set, each dataset is also reduced using each of the dimensionality reduction techniques above and results of classifying the reduced test sets are compared with the result of the original test set to see the extent to which the techniques preserve classification using K-Nearest Neighbors, taking the predicted labels of the original dataset as the basis for evaluation

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1. The Perceptron Preservation

In this section, the results of the comparison amongst dimensionality reduction techniques (explained in chapter three of this thesis) for the perceptron classification preservation of the weather, student and ionosphere datasets are presented. To achieve these results, the training set in each of the three datasets for both the original and reduced datasets is used in training the perceptron, the weight vector obtained from the training phase is then used in classifying the test sets of both the original and reduced datasets. The obtained results are shown in tables 4.1, 4.2 and 4.3.

Table 4.1: Comparing the reduction techniques for the perceptron classification preservation using the weather data set.

Reduction Techniques	18 Attributes	21 Attributes	24 Attributes	27 Attributes
Variance	80%	86.7%	86.7%	86.7%
Novel approach 3	93%	86.7%	86.7%	86.7%
Novel approach 2	86.7%	80%	86.7%	93%
Novel approach 1	86.7%	93%	86.7%	93%
New bottom-up approach modified	86.7%	86.7%	86.7%	93%
New top-down approach modified	93%	100%	93%	93%
New bottom-up approach	80%	80%	86.7%	93%
New top-down approach	93%	93%	86.7%	100%
Principal component analysis	100%	100%	100%	100%
Direct approach	93%	93%	86.7%	100%
Combined approach	80%	86.7%	80%	100%
Modified New random approach	86.7%	93%	86.7%	100%
New Random approach	93%	86.7%	86.7%	93%
Random projection	80%	80%	86.7%	86.7%

Table 4.2: Comparing the reduction techniques for the perceptron classification preservation using the student data set.

Reduction Techniques	12 Attributes	13 Attributes	14 Attributes
Variance	83.3%	91.7%	91.7%
Novel approach 3	100%	100%	91.7%
Novel approach 2	83.3%	83.3%	91.7%
Novel approach 1	83.3%	83.3%	83.3%
New bottom-up approach modified	83.3%	83.3%	66.7%
New top-down approach modified	83.3%	83.3%	91.7%
New bottom-up approach	83.3%	83.3%	83.3%
New top-down approach	75%	66.7%	58.3%
Principal component analysis	58.3%	66.7%	83.3%
Direct approach	58.3%	83.3%	83.3%
Combined approach	83.3%	83.3%	83.3%
Modified New random approach	83.3%	83.3%	91.7%
New Random approach	75%	83.3%	83.3%
Random projection	91.7%	91.7%	75%

Table 4.3: Comparing the reduction techniques for the perceptron classification preservation using the ionosphere data set.

Reduction Techniques	26 Attributes	28 Attributes	30 Attributes	32 Attributes
Variance	90%	83%	90%	90%
Novel approach 3	93%	93%	96.7%	96.7%
Novel approach 2	90%	93%	96.7%	100%
Novel approach 1	83%	93%	96.7%	96.7%
New bottom-up approach modified	90%	93%	96.7%	100%
New top-down approach modified	83%	83%	96.7%	93%
New bottom-up approach	93%	93%	96.7%	96.7%
New top-down approach	73%	90%	96.7%	90%
Principal component analysis	93%	96.7%	86.7%	96.7%
Direct approach	83%	90%	96.7%	93%
Combined approach	80%	83%	96.7%	96.7%
Modified New random approach	90%	90%	90%	100%
New Random approach	96.7%	86.7%	90%	93%
Random projection	86.7%	86.7%	90%	96.7%

The results displayed above imply that, for a reduction of a dataset with p attributes to a dataset with q attributes, where q<<p, the first, second and the third novel approaches perform better than all the other techniques in preserving the perceptron classification. The *confusion matrices* below show the extent to which each dimensionality reduction method preserves the classification of the original datasets. As it is obviously noticed from [24], the most reliable measure for studying the extent of classification preservation is accuracy (as explained above, accuracy is a measure of the

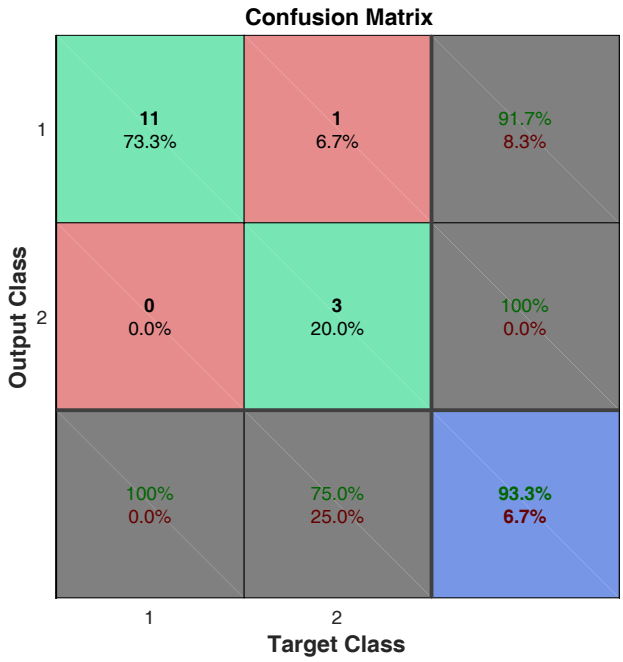


Figure 4.1 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 18 attributes using direct approach.

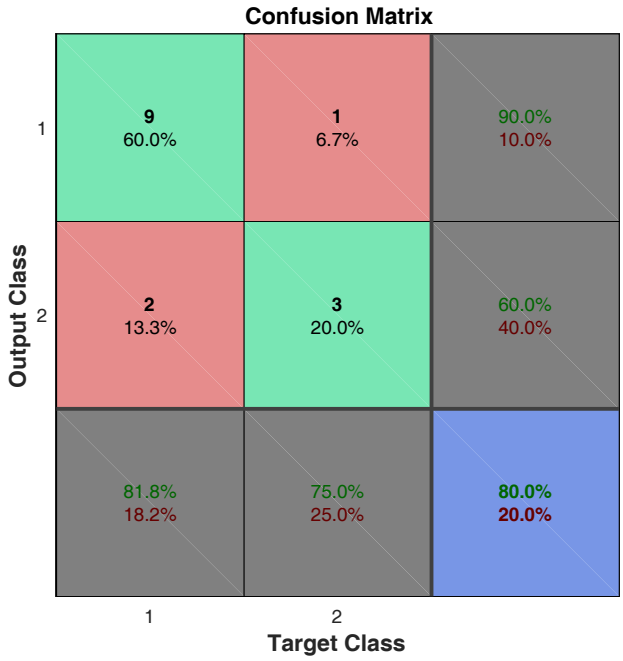


Figure 4.2 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 18 attributes using new bottom up approach.

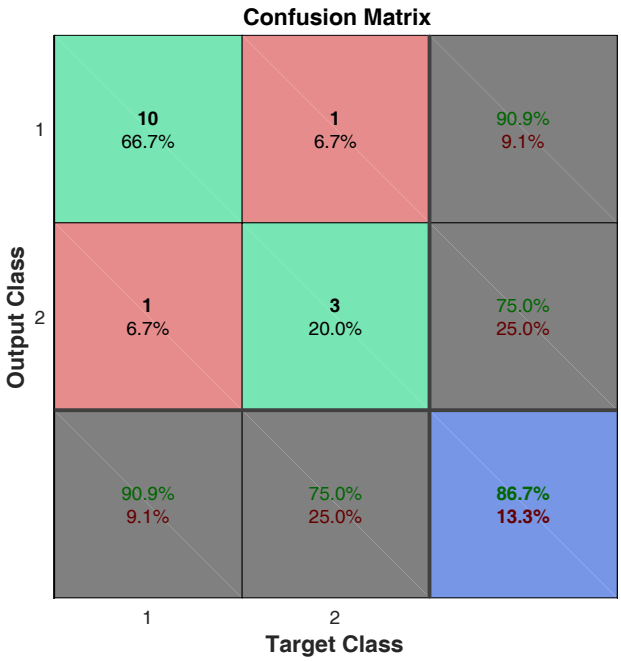


Figure 4.3 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 18 attributes using random approach.

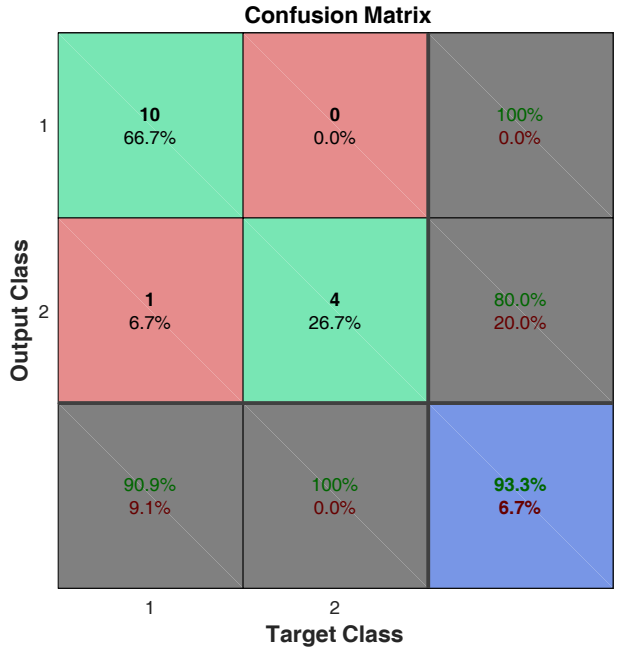


Figure 4.4 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 18 attributes using third novel, new top down, modified new top down and random approaches.

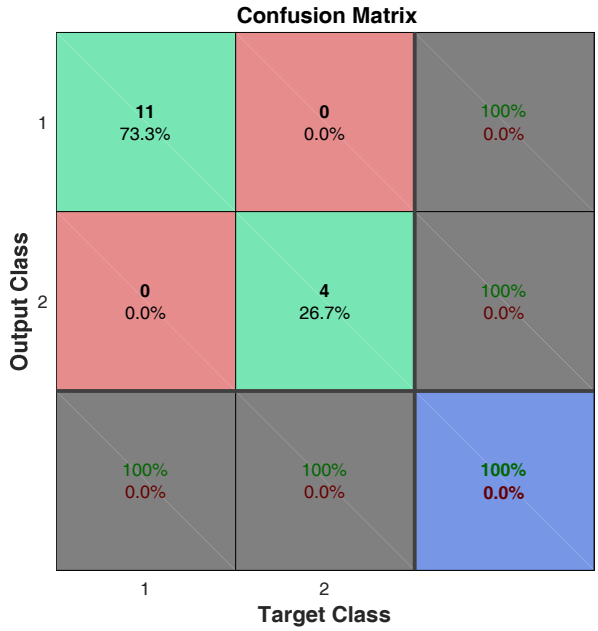


Figure 4.5 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of weather data set from 30 to 27 attributes using new top down, PCA, combined, direct and new random approaches.

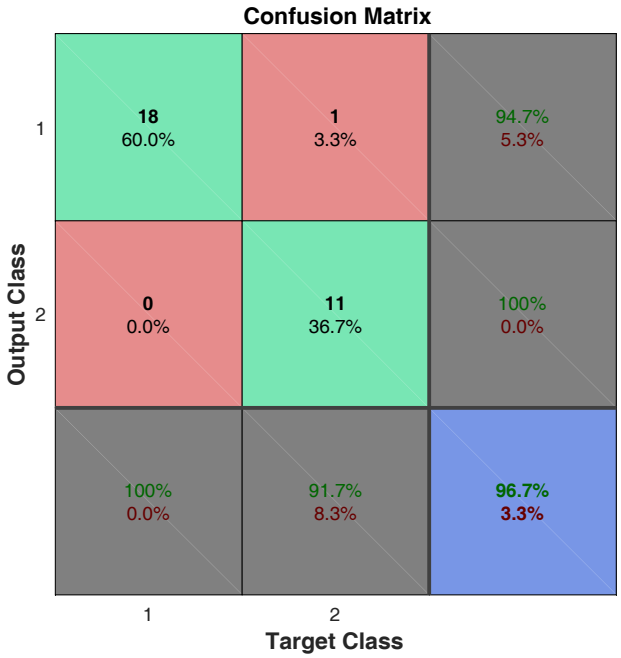


Figure 4.6 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere data set from 30 to 26 attributes using random approach.

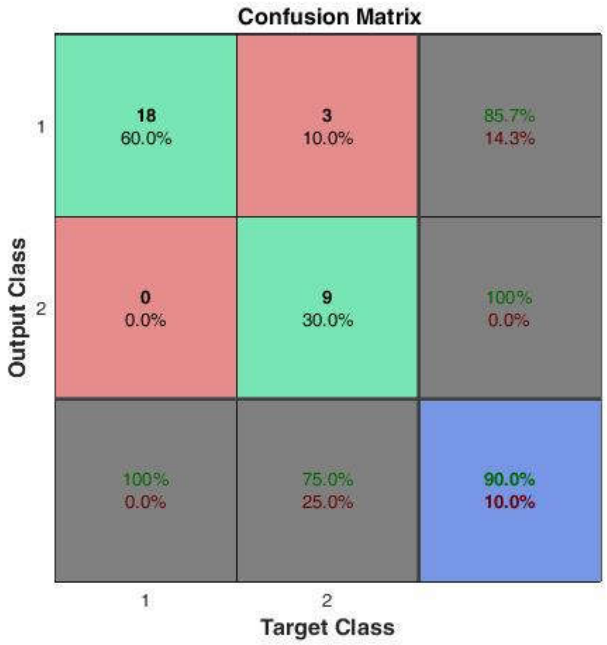


Figure 4.7 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere data set from 30 to 26 attributes using variance approach.

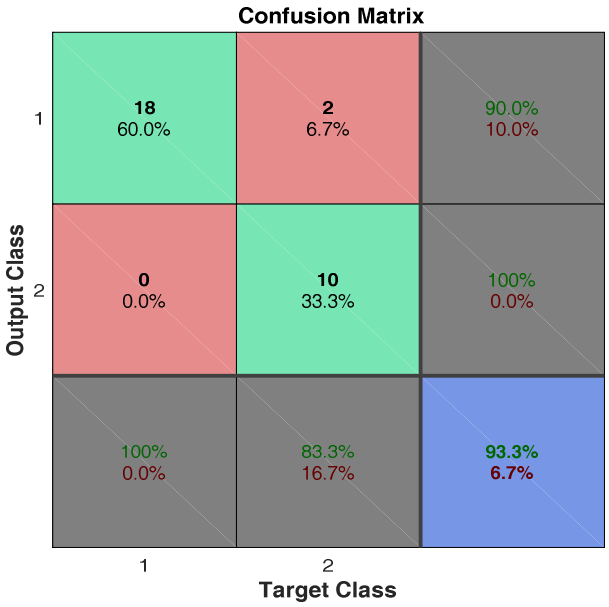


Figure 4.8 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere data set from 30 to 26 attributes using PCA

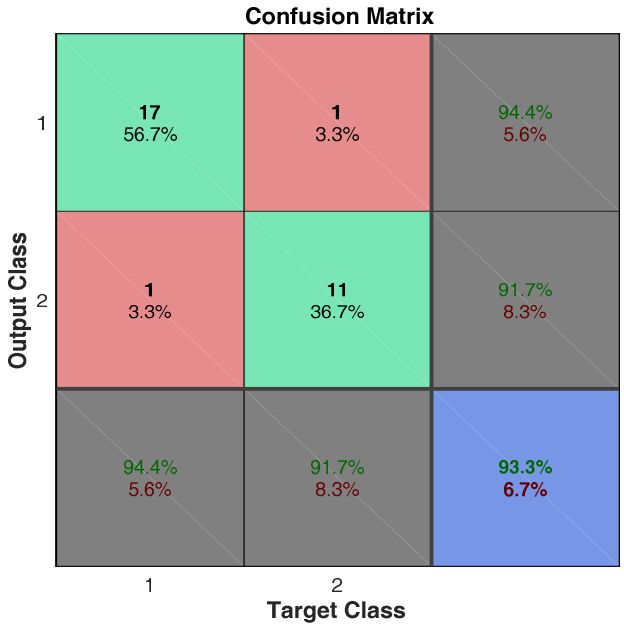


Figure 4.9 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere data set from 30 to 26 attributes using new bottom up.

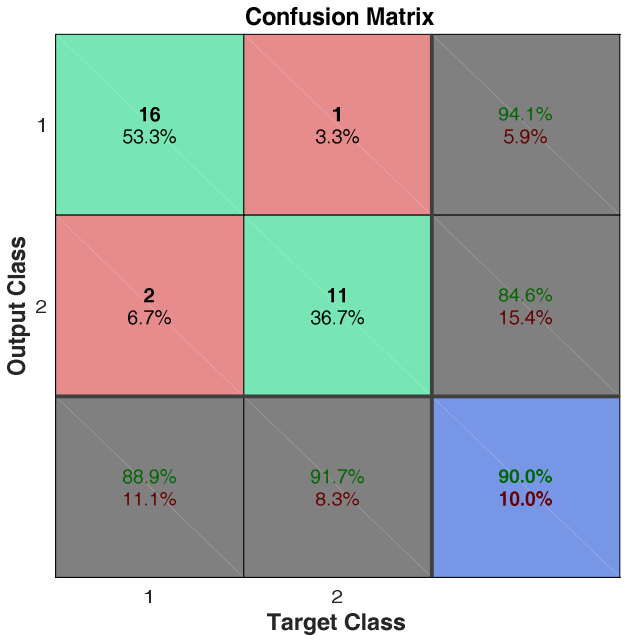


Figure 4.10 *Confusion matrix* showing the extent to which the perceptron classification is preserved by a reduction of ionosphere data set from 30 to 26 attributes using second novel approach.

number of correct predictions made by the classifier). Due to constriction of space, confusion matrices for the rest of the reductions will not be shown here, but the summary of the results can be looked up from the tables.

In the confusion matrices presented in this thesis, the first two diagonal cells of the 3 X 3 matrix represents the number and percentage of correctly classified instances and the third shows the overall correctly classified instances. For example, in Figure 4.1.10 above, 16 instances are correctly predicted as positive which corresponds to 53.3% of all 30 instances of the test data. Likewise, 11 of the negative instances are correctly classified as negative which represents 36.7% of the data.

One tuple from the negative instances is incorrectly predicted as positive, which represents 3.3% of the whole test instances. 2 positive instances are incorrectly classified as negative, this corresponds to 6.7% of the whole data. 94.1% of the 17 positive predictions are correct while 5.9% are wrong. Similarly, 84.6% of the 13 negative predictions are correct while 15.4% are wrong. Out of 18 positive cases, 88.9% are correctly predicted as positive while 11.1% are predicted as negative. 91.7% of the 12 negative instances are correctly predicted as negative and 8.3 % are classified as positive. As for the overall accuracy, 90% of the predictions are correct while 10% are wrong. Until otherwise specified, the same explanation, as above, applies for all the confusion matrices in this thesis.

As seen from tables 4.1, 4.2 and 4.3, on average, for a reduction to the least number of attributes (shown above) in the three datasets, the second and third novel approaches best preserve the perceptron classification while the new top down approach least preserves the classification of the original data sets using the perceptron. The rest of the techniques are approximately as good as

each other in preserving the classification of the original data sets using the perceptron classification.

4.2 K-Nearest Neighbor Classification Preservation

The results of the comparison amongst dimensionality reduction techniques explained in chapter three of this thesis-for K-Nearest Neighbor classification preservation of the weather, student and ionosphere datasets- are presented. To achieve these results, the training set in all the three data sets for both the original and reduced data sets is used in training the K-Nearest Neighbor classifier, each observation in the test sets of the respective training sets will be assigned a label that is most common amongst its k nearest neighbors. A confusion matrix is then used to show the accuracy of the classification for each of the three data sets reduced to *m* (the number of attributes a particular data set is reduced to, e.g. from 30 to 18, 21, 24 and 27 attributes for weather data set) attributes. The obtained results are shown in tables 4.3, 4.4 and 4.5.

The results in tables 4.3, 4.4 and 4.5 show that, on average, almost all the DR techniques yielded excellent results in preserving the classification of the original data set. Random projection is the reduction technique with the least accuracy in preserving K-Nearest Neighbor classification. This means all the other techniques performed relatively better than Random projection in the preservation of K-Nearest Neighbor classification.

Table 4.4: Comparing the reduction techniques for K-Nearest Neighbor classification preservation using the weather data set

Reduction Techniques	18 Attributes	21 Attributes	24 Attributes	27 Attributes
Variance	100%	100%	100%	100%
Novel approach 3	100%	100%	100%	100%
Novel approach 2	100%	100%	100%	100%
Novel approach 1	100%	100%	100%	100%
New bottom-up approach modified	100%	100%	100%	100%
New top-down approach modified	100%	100%	100%	100%
New bottom-up approach	100%	100%	100%	100%
New top-down approach	100%	100%	100%	100%
Principal component analysis	100%	100%	100%	100%
Direct approach	100%	100%	100%	100%
Combined approach	100%	100%	100%	100%
New random approach	100%	100%	100%	100%
Random approach	100%	100%	100%	100%
Random projection	100%	100%	100%	100%

Table 4.5: Comparing the reduction techniques for K-Nearest Neighbor classification preservation using the student data set

Reduction Techniques	12 Attributes	13 Attributes	14 Attributes
Variance	100%	100%	100%
Novel approach 3	100%	100%	100%
Novel approach 2	100%	100%	100%
Novel approach 1	100%	100%	100%
New bottom-up approach modified	100%	100%	100%
New top-down approach modified	100%	100%	100%
New bottom-up approach	100%	100%	100%
New top-down approach	100%	100%	100%
Principal component analysis	100%	100%	100%
Direct approach	100%	100%	100%
Combined approach	100%	100%	100%
New random approach	100%	100%	100%
Random approach	100%	100%	100%
Random projection	100%	100%	100%

Table 4.6: Comparing the reduction techniques for K-Nearest Neighbor classification preservation using the ionosphere data set

Reduction Techniques	26 Attributes	28 Attributes	30 Attributes	32 Attributes
Variance	100%	100%	100%	100%
Novel approach 3	100%	100%	100%	100%
Novel approach 2	100%	100%	96.7%	100%
Novel approach 1	100%	100%	100%	100%
New bottom-up approach modified	100%	100%	100%	100%
New top-down approach modified	100%	100%	100%	100%
New bottom-up approach	100%	100%	96.7%	100%
New top-down approach	96.7%	96.7%	100%	100%
Principal component analysis	100%	100%	100%	100%
Direct approach	100%	100%	100%	100%
Combined approach	100%	100%	100%	100%
Modified New random approach	96.7%	96.7%	96.7%	100%
New Random approach	100%	100%	100%	100%
Random projection	93%	93%	96.7%	96.7%

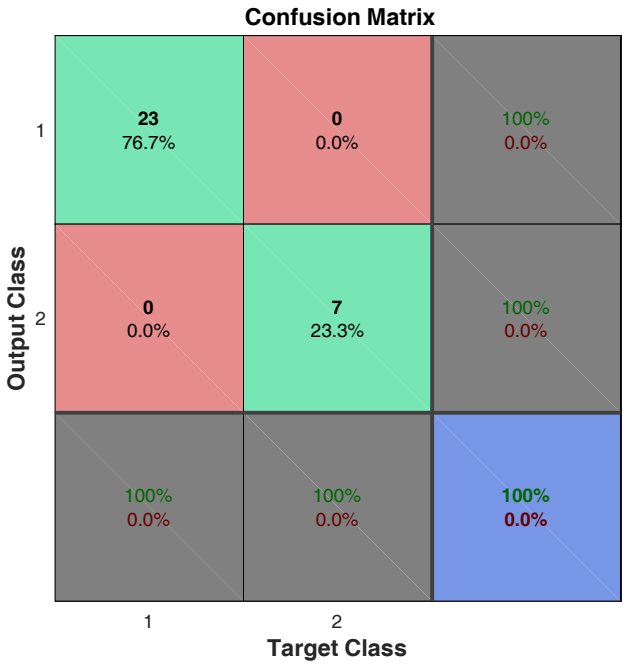


Figure 4.11 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of weather data set from 30 to 26 attributes using PCA

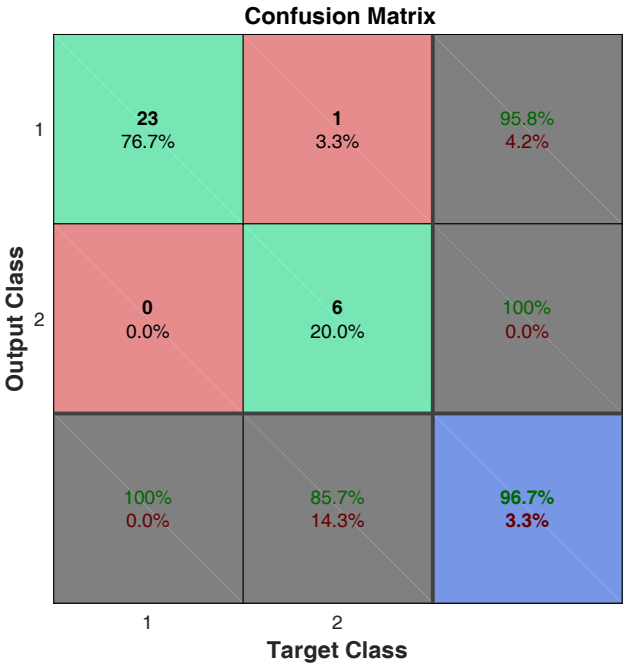


Figure 4.12 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of ionosphere data set from 34 to 30 attributes using second novel approach.

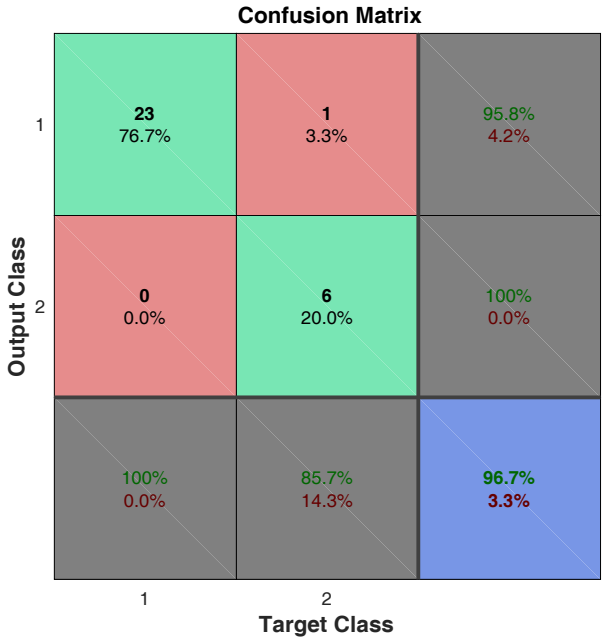


Figure 4.13 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of ionosphere data set from 34 to 26, 28 and 30 attributes using new random approach.

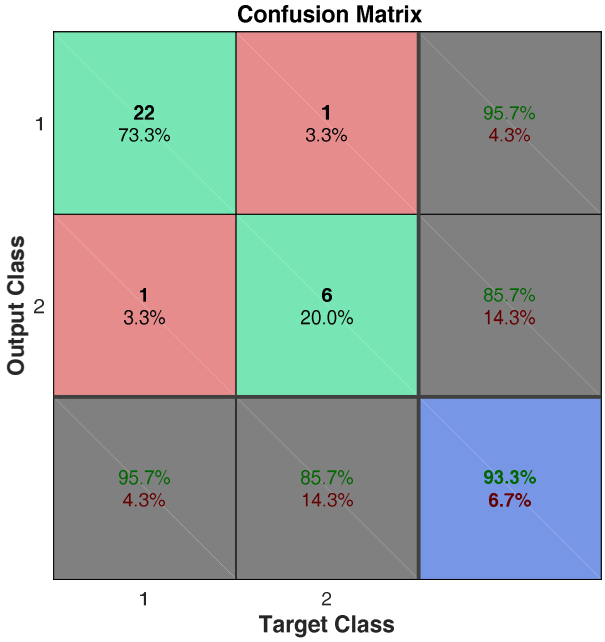


Figure 4.14 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of ionosphere data set from 34 to 26 and 28 attributes using new random projection.

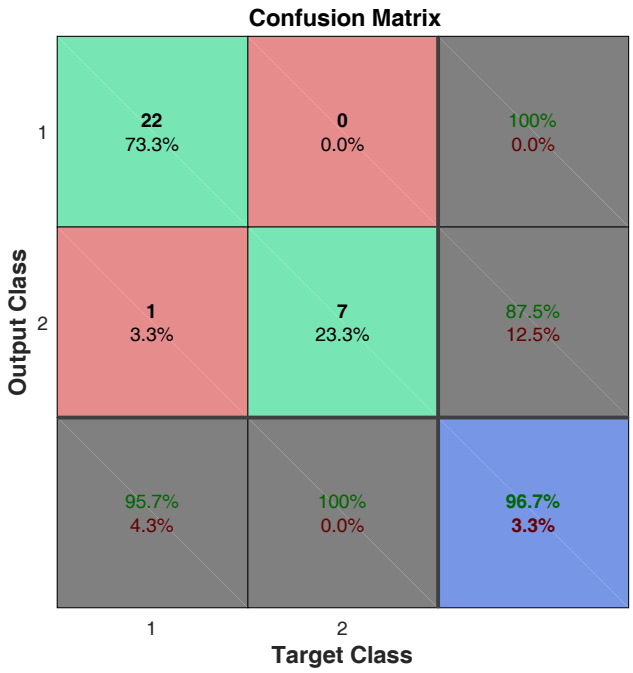


Figure 4.15 *Confusion matrix* showing the extent to which K-Nearest Neighbor classification is preserved by a reduction of ionosphere data set from 34 to 30and 32 attributes using new random projection.

On this regard, all the dimensionality reduction techniques seem to perform much better at preserving K-Nearest Neighbor classification than they do at preserving the classification of the original datasets using the perceptron. In general, the dimensionality reduction techniques implemented in this thesis prove to be very efficient in preserving the classification of both the lazy and eager learners used for this investigation.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

In this thesis, we started by pointing out the challenges faced in the extraction of useful information from available large pool of data which increases at an alarming rate. Dimensionality reduction was introduced as a method that provides a compact representation of an original high-dimensional data, thus making it a very powerful tool and also an invaluable preprocessing step in facilitating the application of many machine learning algorithms.

After that, a review was done on literature related to the subject of this thesis. As pointed out, in the review of related work, dimensionality reduction has been applied to several domains, including machine learning. The methodology used in achieving the objectives of this research was then explained in detail. This includes detailed explanation of the methods involved; fifteen dimensionality reduction techniques, two classification algorithms (the perceptron and K-Nearest Neighbors) and the confusion matrix. The results of the achieved objectives, which were presented in the fourth chapter, revealed the extent to which dimensionality reduction techniques preserve the perceptron and K-Nearest Neighbor classification.

Next, the confusion matrix was used to show the extent to which these fifteen dimensionality reduction techniques – compared against each other - preserve the perceptron and k-nearest neighbor classification.

5.2 Conclusion

The aim of this thesis as stated in chapter 1 is to investigate the extent to which dimensionality reduction techniques preserve classification. This investigation revealed that the dimensionality

reduction techniques implemented in this thesis seem to perform much better at preserving K-Nearest Neighbor classification than they do at preserving the classification of the original datasets using the perceptron. In general, the dimensionality reduction techniques prove to be very efficient in preserving the classification of both the lazy and eager learners used for this investigation.

5.3 Recommendation

It would be interesting and worth investigating the classification preservation of dimensionality reduction methods on more sophisticated classifiers like the support vector machine and decision trees.

REFERENCES

[1] N. Sharma and K. Saroha, "Study of dimension reduction methodologies in data mining," in *International Conference on Computing, Communication and Automation*, 2015, pp. 133–137.

[2] I. K. Fodor, "A survey of dimension reduction techniques," *Center for Applied Scientific Computing, Lawrence Livermore National Laboratory*, no. 1, pp. 1–18, 2002.

[3] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, 2003.

[4] A. S. Nsang, I. Diaz, and A. Ralescu, "Ensemble Clustering based on Heterogeneous Dimensionality Reduction Methods and Context-dependent Similarity Measures," *Int. J. Adv. Sci. Technol.*, vol. 64, pp. 101–118, 2014.

[5] A. S. Nsang, A. Maikori, F. Oguntoyinbo and H. Yusuf, "A New Random Approach To Dimensionality Reduction," in *Int'l Conf. on Advances in Big Data Analytics | ABDA'15 |*, 2014, vol. 60, no. 6, pp. 2114–2142.

[6] D. H. Deshmukh, T. Ghorpade, and P. Padiya, "Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset," in *Proceedings - 2015 International Conference on Communication, Information and Computing Technology, ICCICT 2015*, 2015.

[7] I. Kalamaras, "A novel approach for multimodal graph dimensionality reduction," Imperial college London, 2015.

[8] I. Kavakiotis, O. Tsave, A. Salifoglou, N. Maglaveras, I. Vlahavas, and I. Chouvarda, "Machine Learning and Data Mining Methods in Diabetes Research," *Comput. Struct. Biotechnol. J.*, vol. 15, pp. 104–116, 2017.

[9] T. M. Mitchell, *Machine Learning*, vol. 1, no. 3. 1997.

[10] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[11] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: A review of classification and combining techniques," *Artif. Intell. Rev.*, vol. 26, no. 3, pp. 159–190, 2006.

[12] M. Capó, A. Pérez, and J. A. Lozano, "An efficient approximation to the K-means clustering for massive data," *Knowledge-Based Systems*, 2016.

[13] Y. H. and W. Lam, "Lazy Learning for Classication Based on Query Projections," in *Proceedings of the 2005 SIAM International Conference on Data Mining,* 2005, pp. 227–238.

[14] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[15] W. Ertel, *Introduction to Artificial Intelligence*. 2011.

[16] S. Haykin, *Neural Networks and Learning Machines*, vol. 3. 2008.

[17] I. M. Galván, J. M. Valls, M. García, and P. Isasi, "A lazy learning approach for building classification models," *Int. J. Intell. Syst.*, vol. 26, no. 8, pp. 773–786, 2011.

[18] E. Alpaydın, *Introduction to Machine Learning*, Second Edi. 2010.

[19] N. Singh, "Malware Analysis , Clustering and Classification : A Literature Review," *IJCST Int. J. Comput. Sci. Technol.*, vol. 8491, pp. 68–72, 2015.

[20] Y. Usharani and P. Sammulal, "A novel approach for imputation of missing values for mining medical datasets," in *2015 IEEE International Conference on Computational Intelligence and Computing Research*, 2016.

[21] R. Rajagopal and V. Ranganathan, "Evaluation of effect of unsupervised dimensionality reduction techniques on automated arrhythmia classification," *Biomed. Signal Process. Control*, vol. 34, pp. 1–8, 2017.

[22] I. Martin-Diaz, D. Morinigo-Sotelo, O. Duque-Perez, and R. D. J. Romero-Troncoso, "Advances in Classifier Evaluation: Novel Insights for an Electric Data-Driven Motor Diagnosis," *IEEE Access*, vol. 4, pp. 7028–7038, 2016.

[23] K. Bache and M. Lichman, "UCI Machine Learning Repository," *University of California Irvine School of Information*, vol. 2008, no. 14/8. p. 0, 2013.

[24] A. Nsang, "Novel Approaches to Dimensionality Reduction and Applications," University of Cincinnati, 2011.

[25] A. S. Nsang, D. Edi, and C. Ahanonu, "Query-Based Dimensionality Reduction Applied To Images," in *Int'l Conf. on Advances in Big Data Analytics | ABDA'15 |*, 2015, no. 2, pp. 81–86.

[26] Y. Li, G. Wang, H. Chen, L. Shi, and L. Qin, "An Ant Colony Optimization Based Dimension Reduction Method for High-Dimensional Datasets," *J. Bionic Eng.*, vol. 10, no. 2, pp. 231–241, 2013.

[27] J. Tomášek, "Master Thesis," Charles University in Prague, 2015.

[28] R. Mahapatra, B. Majhi, and M. Rout, "Reduced Feature Based Efficient Cancer Classification Using Single Layer Neural Network," *Procedia Technol.*, vol. 6, pp. 180–187, 2012.

[29] S. Visa, B. Ramsay, A. Ralescu, and E. Van Der Knaap, "Confusion matrix-based feature selection," in *CEUR Workshop Proceedings*, 2011, vol. 710, pp. 120–127.

[30] H. Kim, P. Howland, and H. Park, "Dimension Reduction in Text Classification with Support Vector Machines," *J. ofMachine Learn. Res.*, vol. 6, pp. 37–53, 2005.

[31] I. Gkioulekas and T. Zickler, "Dimensionality reduction using the sparse linear model," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2011.

[32] J. Durand and T. Atkison, "Applying Random Projection to the Classification of Malicious Applications Using Data Mining Algorithms," in *Proceedings of the 50th*

Annual Southeast Regional Conference, 2012, pp. 286–291.

[33] L. Jena and N. K. Kamila, "Distributed Data Mining Classification Algorithms for Prediction of Chronic- Kidney-Disease," *Int. J. Emerg. Res. Manag. &Technology*, vol. 4, no. 11, pp. 110–118, 2015.

[34] N. Bagherzadi, A. O. Borcek, G. Tokdemir, N. Cagiltay, and H. H. Maras, "Analysis of Neurooncological Data to Predict Success of Operation Through Classification," *Proc. 7th ACM Int. Conf. Bioinformatics, Comput. Biol. Heal. Informatics - BCB '16*, pp. 485–486, 2016.

[35] T. R. Baitharu and S. K. Pani, "Analysis of Data Mining Techniques for Healthcare Decision Support System Using Liver Disorder Dataset," *Procedia Comput. Sci.*, vol. 85, no. Cms, pp. 862–870, 2016.

[36] A. S. Nsang, "Image Reduction Using Assorted Dimensionality Reduction Techniques," 2010.

[37] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," *Int. Conf. Knowl. Discov. Data Min.*, pp. 245–250, 2001.

[38] A. S. Nsang and A. Ralescu, "Approaches to Dimensionality Reduction to a Subset of the Original Dimensions," 2010, no. Achlioptas.

[39] S. Haykin, "Rosenblatt ' s Perceptron," *Neural Networks Learn. Mach.*, no. 1943, pp. 47–67, 2009.

[40] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[41] L.-Y. Hu, M.-W. Huang, S.-W. Ke, and C.-F. Tsai, "The distance function effect on k-nearest neighbor classification for medical datasets," *Springerplus*, vol. 5, no. 1, p. 1304, 2016.

[42] V. Kirubha and S. M. Priya, "Survey on Data Mining Algorithms in Disease Prediction," *Int. J. Comput. Trends Technol.*, vol. 38, no. 3, pp. 124–128, 2016.

[43] D. Reddy Edla, V. Gondlekar, and V. Gauns, "HK-Means: A Heuristic Approach to Initialize and Estimate the Number of Clusters in Biological Data," *Acta Phys. Pol. A*, vol. 130, no. 1, pp. 78–82, 2016.

[44] S. Singh and R. Singla, "Comparative Performance of Fault-Prone Prediction Classes with K-means Clustering and MLP," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, 2016.

[45] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, pp. 427–437, 2009.

Appendix A

Weather Dataset

```
13    15    17    18    17    13    14    16    17    18    17    14    13
      14    18    15    13    14    14    16    13    15    13    18    13
      14    13    18    17    15    -1
34    34    33    33    35    33    33    32    35    35    32    34    32
      35    35    33    32    34    35    32    32    34    35    32    33
      35    34    32    33    33    1
20    17    16    19    19    16    19    20    21    20    20    19    19
      18    20    20    19    17    20    18    17    21    21    18    18
      21    20    18    19    16    -1
14    13    15    15    15    15    16    17    17    17    16    13    18
      15    14    18    16    17    14    18    17    15    17    18    16
      18    17    13    18    18    -1
22    23    22    19    21    21    20    22    22    19    23    20    20
      22    22    22    22    22    22    22    21    24    21    19    23
      21    23    22    23    24    1
16    17    20    19    17    20    19    19    19    21    16    16    21
      18    20    20    17    20    19    19    17    20    16    20    21
      21    18    18    20    18    -1
30    30    30    30    29    29    32    32    31    31    32    29    31
      32    32    31    31    32    31    32    30    31    30    31    30
      29    32    29    29    32    1
25    29    27    27    26    24    26    29    27    28    25    24    25
      25    24    25    27    29    25    25    26    25    28    24    24
      24    28    28    29    28    1
23    22    23    21    21    23    19    21    24    23    22    23    20
      20    24    19    19    24    22    23    23    20    23    20    23
      24    23    24    21    19    1
34    32    32    34    35    33    35    33    34    33    32    34    35
      34    34    33    33    34    33    35    34    35    33    34    35
      34    35    34    32    35    1
32    31    30    29    30    32    31    29    29    29    30    30    29
      30    31    31    29    29    31    32    30    30    30    31    32
      31    30    32    32    30    1
26    29    29    27    24    27    26    25    25    25    24    27    28
      24    28    25    27    25    29    28    24    25    25    26    26
      28    27    25    27    29    1
14    16    18    19    18    14    15    17    18    19    18    15    14
      15    19    16    14    16    16    18    15    17    15    20    15
      16    15    20    19    17    -1
33    33    32    32    34    32    32    31    34    34    31    33    31
      34    34    32    31    32    33    30    30    32    33    30    31
      33    32    30    31    31    1
23    20    19    22    22    19    22    23    24    23    23    22    22
      21    23    23    22    21    24    22    21    25    25    22    22
      25    24    22    23    20    1
11    10    12    12    12    12    13    14    14    14    13    10    15
      12    11    15    13    13    10    14    13    11    13    14    12
      14    13    9     14    14    -1
23    24    23    20    22    22    21    23    23    20    24    21    21
      23    23    23    23    24    24    24    23    26    23    21    25
      23    25    24    25    26    1
```

15 16 19 18 16 19 18 18 18 20 15 15 20
17 19 19 16 18 17 17 15 18 14 18 19
19 16 16 18 16 -1
33 33 33 33 32 32 35 35 34 34 35 32 34
35 35 34 34 36 35 36 34 35 34 35 34
33 36 33 33 36 1
22 26 24 24 23 21 23 26 24 25 22 21 22
22 21 22 24 25 21 21 22 21 24 20 20
20 24 24 25 24 1
24 23 24 22 22 24 20 22 25 24 23 24 21
21 25 20 20 26 24 25 25 22 25 22 25
26 25 26 23 21 1
33 31 31 33 34 32 34 32 33 32 31 33 34
33 33 32 32 32 31 33 32 33 31 32 33
32 33 32 30 33 1
35 34 33 32 33 35 34 32 32 32 33 33 32
33 34 34 32 33 35 36 34 34 34 35 36
35 34 36 36 34 1
23 26 26 24 21 24 23 22 22 22 21 24 25
21 25 22 24 21 25 24 20 21 21 22 22
24 23 21 23 25 1
15 17 19 20 19 15 16 18 19 20 19 16 15
16 20 17 15 18 18 20 17 19 17 22 17
18 17 22 21 19 -1
32 32 31 31 33 31 31 30 33 33 30 32 30
33 33 31 30 30 31 28 28 30 31 28 29
31 30 28 29 29 1
26 23 22 25 25 22 25 26 27 26 26 25 25
24 26 26 25 25 28 26 25 29 29 26 26
29 28 26 27 24 1
12 17 15 19 18 27 10 11 11 11 10 15 12
16 18 12 13 19 16 11 15 17 19 20 18
14 18 15 19 21 -1
24 25 24 21 23 23 22 24 24 21 25 22 22
24 24 24 24 26 26 26 25 28 25 23 27
25 27 26 27 28 1
14 15 18 17 15 18 17 17 17 19 14 14 19
16 18 18 15 16 15 15 13 16 12 16 17
17 14 14 16 14 -1
36 36 36 36 35 35 38 38 37 37 38 35 37
38 38 37 37 40 39 40 38 39 38 39 38
37 40 37 37 40 1
19 23 21 21 20 18 20 23 21 22 19 18 19
19 18 19 21 21 17 17 18 17 20 16 16
16 20 20 21 20 -1
25 24 25 23 23 25 21 23 26 25 24 25 22
22 26 21 21 28 26 27 27 24 27 24 27
28 27 28 25 23 1
32 30 30 32 33 31 33 31 32 31 30 32 33
32 32 31 31 30 29 31 30 31 29 30 31
30 31 30 28 31 1
38 37 36 35 36 38 37 35 35 35 36 36 35
36 37 37 35 37 39 40 38 38 38 39 40
39 38 40 40 38 1
20 16 19 21 18 21 20 19 19 19 18 21 18
18 17 19 21 17 19 20 16 17 17 18 18
20 19 17 19 16 -1

16 18 20 21 20 16 17 19 20 21 20 17 16
17 21 18 16 20 20 22 19 21 19 24 19
20 19 24 23 21 -1
31 31 30 30 32 30 30 29 32 32 29 31 29
32 32 30 29 28 29 26 26 28 29 26 27
29 28 26 27 27 1
29 26 25 28 28 25 28 29 30 29 29 28 28
27 29 29 28 29 32 30 29 33 33 30 30
33 32 30 31 28 1
15 12 11 16 13 12 12 15 13 16 18 14 17
20 14 18 13 15 17 16 16 13 15 16 14
16 15 11 16 12 -1
25 26 25 22 24 24 23 25 25 22 26 23 23
25 25 25 25 28 28 28 27 30 27 25 29
27 29 28 29 30 1
13 14 17 16 14 17 16 16 16 18 13 13 18
15 17 17 14 14 13 13 11 14 10 14 15
15 12 12 14 12 -1
39 39 39 39 38 38 41 41 40 40 41 38 40
41 41 40 40 44 43 44 42 43 42 43 42
41 44 41 41 44 1
16 20 18 18 17 15 17 20 18 19 16 15 16
16 15 16 18 17 13 13 14 13 16 12 12
12 16 16 17 16 -1
26 25 26 24 24 26 22 24 27 26 25 26 23
23 27 22 22 30 28 29 29 26 29 26 29
30 29 30 27 25 1
31 29 29 31 32 30 32 30 31 30 29 31 32
31 31 30 30 28 27 29 28 29 27 28 29
28 29 28 26 29 1
41 40 39 38 39 41 40 38 38 38 39 39 38
39 40 40 38 41 43 44 42 42 42 43 44
43 42 44 44 42 1
19 22 22 20 17 20 19 18 18 18 17 20 21
17 21 18 20 15 19 18 14 15 15 16 16
18 17 15 17 19 -1
19 21 23 24 23 19 20 22 23 24 23 20 19
20 24 21 19 24 24 26 23 25 23 28 23
24 23 28 27 25 1
34 34 33 33 35 33 33 32 35 35 32 34 32
35 35 33 32 32 33 30 30 32 33 30 31
33 32 30 31 31 1

Student Dataset

```
65    77    69    75    69    67    79    71    79    73    69    81    73
      77    71    1
61    74    70    66    68    59    72    68    62    64    57    70    66
      58    60    1
46    49    39    45    42    45    45    41    45    46    47    47    43
      39    44    -1
81    80    71    74    79    87    86    77    82    87    93    92    83
      90    95    1
19    22    18    21    20    22    24    29    22    21    19    23    23
      20    25    -1
88    76    80    88    79    82    70    74    80    71    76    64    68
      72    63    1
69    77    74    69    76    99    100   88    81    88    93    94    82
      73    80    1
31    32    32    31    28    29    35    35    32    34    29    33    33
      35    31    -1
89    93    78    77    80    79    83    68    65    68    81    85    70
      69    72    1
26    30    28    28    33    32    30    31    34    26    36    28    29
      32    24    -1
55    64    60    50    63    57    66    62    54    67    55    64    60
      50    63    1
84    83    80    77    78    82    81    78    73    74    88    87    84
      81    82    1
28    33    28    32    34    24    32    30    33    35    29    34    32
      30    32    -1
86    75    81    87    79    92    81    87    95    87    86    75    81
      87    79    1
29    33    23    30    25    32    31    33    31    28    27    30    36
      29    34    -1
84    82    86    92    85    78    76    80    84    77    88    86    90
      96    89    1
71    70    73    81    79    81    80    83    93    91    71    70    73
      81    79    1
40    43    46    42    41    40    46    41    42    43    38    39    39
      38    39    -1
81    88    80    79    83    71    78    70    91    95    73    80    72
      95    99    1
32    33    36    35    31    30    35    30    33    29    28    33    28
      31    30    -1
84    78    80    74    80    86    80    82    78    84    88    82    84
      82    88    1
81    77    81    83    79    79    75    79    79    75    77    73    77
      75    71    1
27    28    30    29    26    27    30    31    29    30    27    32    30
      28    34    -1
78    66    90    84    75    84    72    96    92    83    82    70    94
      88    79    1
43    44    45    46    42    40    40    40    40    39    38    38    38
      36    42    -1
67    74    73    76    72    61    68    67    68    64    67    74    73
      76    72    1
```

```
96    87    80    78    82    100   97    90    90    94    94    91    84
      82    86    1
29    26    27    33    36    34    32    30    27    29    26    28    26
      29    31    -1
66    69    75    68    70    56    59    65    56    58    66    69    75
      68    70    1
19    24    22    22    20    21    26    24    26    23    23    24    26
      22    25    -1
73    75    64    76    74    75    77    66    80    78    77    79    68
      84    82    1
75    68    82    85    79    73    66    80    81    75    71    64    78
      77    71    1
```

Ionosphere Dataset

```
1     0     0.17188     -1    -1    1     0     0     0     0     -1    1
      0     0     -0.61354    -0.67708    0.80521     0.36146     0.51979
      0.14375     0     0     -1    -0.27083    -0.84792    0.9625      1
      1     -1    0.67708     0     0     0     0     -1
1     0     1     0.09771     1     0.12197     1     0.22574     0.98602
      0.09237     0.9493      0.19211     0.92992     0.24288     0.89241
      0.28343     0.85529     0.26721     0.83656     0.33129     0.83393
      0.31698     0.74829     0.39597     0.76193     0.34658     0.68452
      0.42746     0.62764     0.46031     0.56791     0.47033     0.54252
      0.50903     1
1     0     0.01667     -0.35625    0     0     0     0     0     0     0
      0     0     0     0.12292     -0.55 0.22813     0.82813     1     -
0.42292     0     0     0.08333     -1    -0.10625    -0.16667    1     -
0.76667     -1    0.18854     0     0     1     -0.27292    -1
1     0     1     0.16801     0.99352     0.16334     0.94616     0.33347
      0.91759     0.2261      0.91408     0.37107     0.8425      0.46899
      0.81011     0.49225     0.78473     0.48311     0.65091     0.56977
      0.56553     0.58071     0.55586     0.6472      0.48311     0.55236
      0.43317     0.69129     0.35684     0.76147     0.33921     0.66844
      0.22101     0.78685     1
1     0     0.63816     1     0.20833     -1    1     1     0.87719
      0.30921     -0.66886    1     -0.05921    0.58772     0.01754
      0.05044     -0.51535    -1    0.14254     -0.03289    0.32675     -
0.4386      -1    1     0.80921     -1    1     -0.0614     1     1
      0.20614     -1    1     1     -1
1     0     1     -0.41457    1     0.76131     0.8706      0.18593     1
      -0.09925    0.93844     0.4799      0.65452     -0.1608     1
      0.00879     0.97613     -0.50126    0.80025     -0.24497    0.88065
      -0.19095    1     -0.12312    0.93593     0.10678     0.9289      -
0.07249     1     -0.27387    0.4397      0.19849     0.51382     -0.05402
      1
1     0     0.84783     0.10598     1     0.3913      1     -1    0.66938
      0.08424     1     0.27038     1     0.60598     1     0.35507     1
      0.02672     0.58424     -0.43025    1     0.63496     0.8913
      0.26585     0.91033     -0.33333    1     0.15942     0.37681     -
0.01947     1     0.22464     1     0.37409     -1
1     0     1     0.28046     1     0.02477     1     0.07764     1
      0.04317     0.98762     0.33266     1     0.05489     1     0.04384
      0.9575      -0.24598    0.84371     -0.08668    1     0.0415
      0.99933     0.27376     1     -0.39056    0.96414     -0.02174
      0.86747     0.2336      0.94578     -0.22021    0.80355     -0.07329
      1
0     0     1     -1    1     -1    1     -1    1     -1    1     1     1
      1     1     -1    1     1     1     1     1     1     1     -1    1
      -1    1     -1    1     0.65625     0     0     1     -1    -1
1     0     1     0.67784     0.81309     0.82021     0.43019     1
      0.20619     0.80541     -0.43872    1     -0.79135    0.77092     -1
      0.40268     -0.39046    -0.58634    -0.97907    -0.42822    -0.73083
      -0.76339    -0.37671    -0.97491    0.41366     -1    0.41778     -
0.93296     0.25773     -1    0.9357      -0.35222    0.98816     0.03446
      1
```

```
1     0     1     1     1     -1    1     -1    1     1     1     1     1
      1     1     -1    1     1     1     1     1     1     1     1     1
      1     1     0.5   0     0     1     -1    1     -1    -1
1     0     1     0.03529     1     0.18281     1     0.26968     1
      0.25068     1     0.28778     1     0.38643     1     0.31674     1
      0.65701     1     0.53846     1     0.61267     1     0.59457
      0.89593     0.68326     0.89502     0.71374     0.85611     0.67149
      0.74389     0.85611     0.71493     0.75837     1
0     0     1     -1    1     1     -1    -1    1     -1    0     0     0
      0     -1    1     1     -1    1     -1    -0.75 1     1     -1    1
      -1    1     -1    -1    -1    0     0     1     -1    -1
1     0     0.96087     0.0862      0.9676      0.19279     0.96026
      0.27451     0.98044     0.35052     0.92867     0.46281     0.86265
      0.52517     0.8282      0.58794     0.73242     0.69065     0.69003
      0.7314      0.54473     0.6882      0.48339     0.76197     0.40615
      0.74689     0.33401     0.83796     0.24944     0.86061     0.13756
      0.86835     0.09048     0.86285     1
1     0     0.69444     0.38889     0     0     -0.32937    0.69841     0
      0     0     0     0     0     0.20635     -0.24206    0.21032
      0.19444     0.46429     0.78175     0     0     0     0     0.73413
      0.27381     0.7619      0.63492     0     0     0     0     0     0
      -1
1     0     1     0.0507      1     0.10827     1     0.19498     1
      0.28453     1     0.34826     1     0.38261     0.94575     0.42881
      0.89126     0.50391     0.75906     0.58801     0.80644     0.59962
      0.79578     0.62758     0.66643     0.63942     0.59417     0.69435
      0.49538     0.72684     0.47027     0.71689     0.33381     0.75243
      1
0     0     1     1     0     0     1     -1    1     -1    1     1     1
      1     1     -1    1     1     1     1     1     -1    -1    -1    1
      -1    1     -1    1     1     0     0     1     -1    -1
1     0     1     0.04078     1     0.11982     1     0.16159     1
      0.27921     0.98703     0.30889     0.92745     0.37639     0.91118
      0.39749     0.81939     0.46059     0.78619     0.46994     0.794
      0.56282     0.70331     0.58129     0.67077     0.59723     0.58903
      0.6099      0.53952     0.60932     0.45312     0.63636     0.40442
      0.62658     1
0     0     1     1     1     -1    1     1     1     1     1     1     1
      1     1     1     1     -1    -1    1     -1    1     -1    1     1
      -1    1     1     -1    1     -1    -1    -1    1     -1
1     0     1     0.24168     1     0.4859      1     0.72973     1     1
      1     1     1     1     1     0.77128     1     1     1     1
      0.74468     1     0.89647     1     0.64628     1     0.38255     1
      0.10819     1     -0.1737     1     -0.81383    1     1
0     0     1     1     1     -1    1     1     -1    1     0     0     1
      1     0     0     0     0     -1    1     -1    1     1     1     1
      -1    1     1     1     1     1     -1    -1    1     -1
1     0     1     -0.06604    1     0.62937     1     0.09557     1
      0.2028      1     -1    1     -0.40559    1     -0.15851    1
      0.04895     1     -0.61538    1     -0.26573    1     -1    1     -
0.58042     1     -0.81372    1     -1    1     -0.78555    1     -0.48252
      1
```

```
0     0     1     -1    1     1     1     1     1     1     1     1
      -1    1     -1    1     1     1     -1    1     1     1     1
      -1    1     1     1     -1    1     1     1     -1    -1
1     0     0.92277     0.07804     0.92679     0.16251     0.89702
      0.24618     0.84111     0.35197     0.78801     0.42196     0.70716
      0.46983     0.70796     0.56476     0.60459     0.642 0.51247
      0.64924     0.39903     0.66975     0.34232     0.68343     0.23693
      0.76146     0.18765     0.73885     0.09694     0.71038     0.02735
      0.77072     -0.04023    0.69509     1
1     0     0.68198     -0.17314    0.82332     0.21908     0.46643
      0.32862     0.25795     0.58304     1     -0.15194    0.0106
      0.44523     0.0106      0.38869     0.18681     0.41168     0.10567
      0.36353     0.04325     0.30745     -0.00083    0.24936     -0.02862
      0.19405     -0.04314    0.14481     -0.04779    0.10349     -0.04585
      0.07064     -0.04013    0.04586     -1
1     0     0.74852     -0.02811    0.6568      -0.05178    0.80621
      0.02811     0.85947     0.02515     0.63462     0.08728     0.71598
      0.0784      0.73077     0.05178     0.7855      -0.27811    0.65976
      -0.01479    0.78698     0.06953     0.34615     -0.18639    0.65385
      0.02811     0.61009     -0.06637    0.5355      -0.21154    0.59024
      -0.14053    0.56361     0.02959     1
1     0     0.39179     -0.06343    0.97464     0.04328     1     1
      0.35821     0.15299     0.54478     0.1306      0.61567     -0.8209
      0.57836     0.6791      0.66791     -0.10448    0.46642     -0.11567
      0.65574     0.14792     0.83209     0.45522     0.47015     0.16418
      0.49309     0.1463      0.32463     -0.02612    0.39118     0.13521
      0.34411     0.12755     -1
1     0     0.67547     0.04528     0.76981     -0.10566    0.77358
      0.03774     0.66038     -0.04528    0.64528     0.01132     0.66792
      -0.13962    0.72075     -0.02264    0.76981     0.08679     0.61887
      -0.07925    0.75849     -0.23774    0.73962     -0.14717    0.84906
      -0.15094    0.73886     -0.05801    0.66792     0.02264     0.86415
      0.03774     0.73208     0.00755     1
1     0     0.72727     -0.05 0.89241     0.03462     1     0.72727
      0.66364     -0.05909    0.48182     -0.16818    0.81809     0.09559
      0.56818     1     0.50455     0.21818     0.66818     0.1   1     -0.3
      0.98636     -1    0.57273     0.32727     0.56982     0.14673
      0.42273     0.08182     0.48927     0.14643     1     1     -1
1     0     0.57647     -0.01569    0.40392     0     0.38431     0.12941
      0.4   -0.05882    0.56471     0.14118     0.46667     0.08235
      0.52549     -0.0549     0.58039     0.01569     0.50196     0
      0.45882     0.06667     0.58039     0.08235     0.49804     0.00392
      0.48601     0.10039     0.46275     0.08235     0.45098     0.23529
      0.43137     0.17255     1
1     0     0.41932     0.12482     0.35  0.125 0.23182     0.27955     -
0.03636     0.44318     0.04517     0.36194     -0.19091    0.33636     -
0.1335      0.27322     0.02727     0.40455     -0.34773    0.12727     -
0.20028     0.05078     -0.18636    0.36364     -0.14003    -0.04802    -
0.09971     -0.07114    -1    -1    -0.02916    -0.07464    -0.00526    -
0.06314     -1
1     0     0.88305     -0.21996    1     0.36373     0.82403     0.19206
      0.85086     0.05901     0.90558     -0.04292    0.85193     0.25
      0.77897     0.25322     0.69206     0.5794      0.7103      0.39056
      0.73176     0.27575     1     0.34871     0.5676      0.52039
```

```
      0.69811     0.53235     0.80901     0.58584     0.43026     0.70923
      0.52361     0.54185     1
1     0     0.84557     -0.0858     -0.31745    -0.80553    -0.08961    -
0.56435     0.80648     0.04576     0.89514     -0.00763    -0.18494
      0.63966     -0.20019    -0.68065    0.85701     -0.11344    0.77979
      -0.15729    -0.06959    0.5081      -0.34128    0.80934     0.78932
      -0.03718    0.70882     -0.25288    0.77884     -0.14109    -0.21354
      -0.7817     -0.18494    -0.59867    -1
1     0     0.7087      -0.24783    0.64348     0.04348     0.45217
      0.38261     0.65217     0.18261     0.5   0.26957     0.57826     -
0.23043     0.50435     0.37826     0.38696     -0.42609    0.36087     -
0.26087     0.26957     0.11739     0.53246     -0.03845    0.31304     -
0.12174     0.4993      -0.04264    0.48348     -0.04448    0.64348     -
0.25217     0.50435     0.14783     1
1     0     -0.5418     0.14861     -0.33746    0.73375     0.52012     -
0.13932     0.31889     -0.06811    0.20743     -0.1517     0.47368
      0.08978     0.56347     -0.1548     0.16409     0.45201     0.33746
      0.03406     0.50464     0.07121     -0.63777    -0.6161     1
      0.65635     0.41348     -0.40116    -0.1517     0.11146     0.02399
      0.5582      0.52632     -0.08978    -1
1     0     0.29202     0.13582     0.45331     0.16808     0.51783     -
0.00509     0.52632     0.20883     0.52462     -0.16638    0.47368     -
0.04754     0.55518     0.03905     0.81664     -0.22411    0.42445     -
0.04244     0.34975     0.06621     0.28183     -0.20883    0.51731     -
0.03176     0.50369     -0.03351    0.34635     0.09847     0.70798     -
0.01868     0.39559     -0.03226    1
1     0     0.79157     0.16851     0     0     0.56541     0.06874
      0.39468     1     0.38359     0.99557     -0.02439    0.53215
      0.23725     0.1286      -0.02661    0.95122     -0.50998    0.84922
      -0.102      0.38803     -0.42572    0.23725     -0.91574    0.8071
      -0.34146    0.88248     -1    0.69401     -1    0.1286      0     0
      -1
1     0     0.90116     0.16607     0.79299     0.37379     0.7299
      0.50515     0.59784     0.72997     0.44303     0.81152     0.24412
      0.87493     0.06438     0.85038     -0.12611    0.87396     -0.28739
      0.79617     -0.46635    0.65924     -0.57135    0.53805     -0.68159
      0.39951     -0.71844    0.25835     -0.72369    0.11218     -0.71475
      -0.05525    -0.67699    -0.19904    1
1     0     0.97714     0.19049     0.82683     0.46259     0.71771
      0.58732     0.47968     0.84278     0.31409     0.92643     0.10289
      0.93945     -0.13254    0.8429      -0.3202     0.91624     -0.52145
      0.79525     -0.68274    0.49508     -0.77408    0.33537     -0.85376
      0.17849     -0.83314    -0.01358    -0.82366    -0.19321    -0.67289
      -0.33662    -0.59943    -0.497      1
1     0     -1    -1    0     0     0.50814     -0.78502    0.60586
      0.32899     -1    -0.41368    0     0     0     0     1     -0.2671
      0.36482     -0.63518    0.97068     -1    -1    -1    1     -0.59609
      -1    -1    -1    -1    1     -1    0     0     -1
1     0     0.74084     0.04974     0.79074     0.02543     0.78575
      0.03793     0.6623      0.09948     0.67801     0.31152     0.75934
      0.07348     0.74695     0.08442     0.70681     -0.07853    0.63613
      0     0.70021     0.11355     0.68183     0.12185     0.67016
      0.15445     0.64158     0.13608     0.65707     0.17539     0.59759
      0.14697     0.57455     0.15114     1
```

```
1     0     0.01975     0.00705     0.0409      -0.00846    0.02116
      0.01128     0.01128     0.04372     0.00282     0.00141     0.01975
      -0.03103    -0.01975    0.06065     -0.0409     0.0268      -0.02398
      -0.00423    0.04372     -0.02539    0.01834     0     0     -0.01269
      0.01834     -0.01128    0.00564     -0.01551    -0.01693    -0.02398
      0.00705     0     -1
1     0     0.85736     0.00075     0.81927     -0.05676    0.77521     -
0.04182     0.84317     0.09037     0.86258     0.11949     0.88051     -
0.06124     0.78342     0.0351      0.83719     -0.06796    0.8357      -
0.1419      0.88125     0.01195     0.90515     0.0224      0.79686     -
0.01942     0.82383     -0.03678    0.88125     -0.06423    0.73936     -
0.01942     0.79089     -0.09186    1
1     0     1     -1    1     1     -1    1     1     -1    1     -1    -1
      -1    -1    1     1     1     1     1     -1    1     1     -1    1
      -1    1     1     1     1     -1    1     -1    1     -1
1     0     0.85209     0.39252     0.38887     0.76432     0.08858
      0.98903     -0.42625    0.88744     -0.76229    0.4998      -0.93092
      0.10768     -0.859      -0.31044    -0.6603     -0.55262    -0.1926
      -0.86063    0.28444     -0.80496    0.64649     -0.3523     0.77814
      -0.23324    0.71698     0.21343     0.3783      0.5831      0.19667
      0.66315     -0.11215    0.64933     1
1     0     1     1     1     0.5125      0.625 -1    1     1     0.025
      0.03125     1     1     0     0     1     -1    1     1     1     1
      0.3125      1     1     1     1     1     1     1     -0.94375    1
      0     0     -1
1     0     1     0.54902     0.62745     1     0.01961     1     -0.4902
      0.92157     -0.82353    0.58824     -1    0.11765     -0.96078    -
0.33333     -0.64706    -0.68627    -0.23529    -0.86275    0.35294     -1
      0.7451      -0.72549    0.92157     -0.21569    0.92874     0.21876
      0.72549     0.56863     0.23529     0.90196     -0.11765    0.90196
      1
1     0     0     0     -1    -1    -1    1     0     0     -1    1     1
      1     1     -1    0     0     0     0     -1    -1    -1    1     1
      0.4375      1     -1    0     0     -1    -1    -1    1     -1
1     0     0.44444     0.44444     0.53695     0.90763     -0.22222    1
      -0.33333    0.88889     -1    0.33333     -1    -0.11111    -1    -
0.22222     -0.66667    -0.77778    0.55556     -1    -0.22222    -0.77778
      0.77778     -0.22222    0.33333     0     0.9212      0.45019
      0.57454     0.84353     0.22222     1     -0.55556    1     1
0     0     0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0     0     0     0
      0     1     0     0     0     0     0     0     0     -1
1     0     1     0     1     0     0.5   0.5   0.75  0     0.91201
      0.12094     0.89067     0.1421      0.86922     0.16228     0.75  0.25
      0.75  0.5   0.75  0     1     -0.25 0.5   0.5   0.73944     0.26388
      0.75  0.25  0.69635     0.29074     0.67493     0.30293     1
0     0     -1    1     1     1     0     0     1     -1    1     -1    1
      -1    -1    -1    0     0     -1    -1    0     0     0     0     -1
      -1    1     -1    1     1     -1    -1    0     0     -1
1     0     1     0     1     0     0.66667     0.11111     1     -0.11111
      0.88889     -0.11111    1     -0.22222    0.77778     0     0.77778
      0     1     -0.11111    0.77778     -0.11111    0.66667     -0.11111
      0.66667     0     0.90347     -0.05352    1     0.11111     0.88889
      -0.11111    1     0     1
```

```
0     0     0     0     0     0     0     0     0     0     0     0     -1
      -1    0     0     1     0.75  0     0     0     0     -1    1     0
      0     1     -1    -1    -1    1     1     0     0     -1
1     0     1     0.45455     1     -0.45455    1     0.09091     1     -
0.09091     1     0     1     -0.27273    1     -0.18182    1     0.09091
      1     0     1     -0.36364    1     0.09091     1     -0.09091    1
      -0.04914    1     0.45455     1     -0.27273    1     -0.18182    1
1     0     0.62121     -0.63636    0     0     0     0     0.3447
      0.28788     0.42803     0.39394     -0.07576    0.51894     0.36364
      0.31439     -0.53788    0.32955     0.12121     -0.14773    0.01894
      -0.53409    -0.57576    0.17803     0.29167     -0.27273    0.25758
      -0.57576    0.43182     0.24242     0.18182     -0.02273    0.17045
      -0.41667    -1
1     0     1     0.11765     1     0.23529     1     0.41176     1
      0.05882     1     0.23529     1     0.11765     1     0.47059     1
      -0.05882    1     -0.11765    1     0.35294     1     0.41176     1
      -0.11765    1     0.20225     1     0.05882     1     0.35294     1
      0.23529     1
1     0     0     0     -1    -0.62766    1     0.51064     0.07979     -
0.23404     -1    -0.3617     0.12766     -0.59043    1     -1    0     0
      0.82979     -0.07979    -0.25 1     0.17021     -0.70745    0     0
      -0.19149    -0.46809    -0.2234     -0.48936    0.74468     0.90426
      -0.67553    0.45745     -1
1     0     0.91667     0.29167     0.83333     -0.16667    0.70833     0.25
      0.875 -0.08333    0.91667     0.04167     0.83333     0.125 0.70833
      0     0.875 0.04167     1     0.08333     0.66667     -0.08333    0.75
      0.16667     0.83333     -0.125      0.83796     0.05503     1
      0.20833     0.70833     0     0.70833     0.04167     1
1     0     0.1859      -0.16667    0     0     0     0     0     0     0
      0     0.11538     -0.19071    0     0     0     0     0     0     0
      0     -0.05128    -0.06571    0.07853     0.08974     0.17308     -
0.10897     0.125 0.09615     0.02564     -0.04808    0.16827     0.19551
      -1
1     0     0.99539     -0.05889    0.85243     0.02306     0.83398     -
0.37708     1     0.0376      0.85243     -0.17755    0.59755     -0.44945
      0.60536     -0.38223    0.84356     -0.38542    0.58212     -0.32192
      0.56971     -0.29674    0.36946     -0.47357    0.56811     -0.51171
      0.41078     -0.46168    0.21266     -0.3409     0.42267     -0.54487
      0.18641     -0.453      1
1     0     1     -0.03365    1     0.00485     1     -0.12062    0.88965
      0.01198     0.73082     0.05346     0.85443     0.00827     0.54591
      0.00299     0.83775     -0.13644    0.75535     -0.0854     0.70887
      -0.27502    0.43385     -0.12062    0.57528     -0.4022     0.58984
      -0.22145    0.431 -0.17365    0.60436     -0.2418     0.56045     -
0.38238     1
1     0     1     -0.02401    0.9414      0.06531     0.92106     -0.23255
      0.77152     -0.16399    0.52798     -0.20275    0.56409     -0.00712
      0.34395     -0.27457    0.5294      -0.2178     0.45107     -0.17813
      0.05982     -0.35575    0.02309     -0.52879    0.03286     -0.65158
      0.1329      -0.53206    0.02431     -0.62197    -0.05707    -0.59573
      -0.04608    -0.65697    1
1     0     0.02337     -0.00592    -0.09924    -0.11949    -0.00763    -
0.11824     0.14706     0.06637     0.03786     -0.06302    0     0     -
0.04572     -0.1554     -0.00343    -0.10196    -0.11575    -0.05414
```

```
      0.01838     0.03669     0.01519     0.00888     0.03513     -0.01535
      -0.0324     0.09223     -0.07859    0.00732     0     0     -0.00039
      0.12011     -1
1     0     0.97588     -0.10602    0.94601     -0.208      0.92806     -
0.2835      0.85996     -0.27342    0.79766     -0.47929    0.78225     -
0.50764     0.74628     -0.61436    0.57945     -0.68086    0.37852     -
0.73641     0.36324     -0.76562    0.31898     -0.79753    0.22792     -
0.81634     0.13659     -0.8251     0.04606     -0.82395    -0.04262    -
0.81318     -0.13832    -0.80975    1
0     0     0     0     0     0     1     -1    0     0     -1    -1    0
      0     0     0     1     1     -1    -1    0     0     0     0     1
      1     1     1     0     0     1     1     0     0     -1
1     0     0.96355     -0.07198    1     -0.14333    1     -0.21313    1
      -0.36174    0.9257      -0.43569    0.9451      -0.40668    0.90392
      -0.46381    0.98305     -0.35257    0.84537     -0.6602     0.75346
      -0.60589    0.69637     -0.64225    0.85106     -0.6544     0.57577
      -0.69712    0.25435     -0.63919    0.45114     -0.72779    0.38895
      -0.7342     1
1     0     -0.01864    -0.08459    0     0     0     0     0.1147      -
0.2681      -0.45663    -0.38172    0     0     -0.33656    0.38602     -
0.37133     0.15018     0.63728     0.22115     0     0     0     0     -
0.14803     -0.01326    0.20645     -0.02294    0     0     0.16595
      0.24086     -0.08208    0.38065     -1
1     0     1     0.06655     1     -0.18388    1     -0.2732     1     -
0.43107     1     -0.41349    0.96232     -0.51874    0.90711     -0.59017
      0.8923      -0.66474    0.69876     -0.70997    0.70645     -0.7632
      0.63081     -0.80544    0.55867     -0.89128    0.47211     -0.865
      0.40303     -0.83675    0.30996     -0.89093    0.22995     -0.89158
      1
1     0     1     -0.5421     1     -1    1     -1    1     0.36217     1
      -0.41119    1     1     1     -1    1     -0.29354    1     -0.93599
      1     1     1     1     1     -0.40888    1     -0.62745    1     -1
      1     -1    1     -1    -1
1     0     1     -0.16316    1     -0.10169    0.99999     -0.15197    1
      -0.19277    0.94055     -0.35151    0.95735     -0.29785    0.93719
      -0.34412    0.94486     -0.28106    0.90137     -0.43383    0.86043
      -0.47308    0.82987     -0.5122     0.8408      -0.47137    0.76224
      -0.5837     0.65723     -0.68794    0.68714     -0.64537    0.64727
      -0.67226    1
1     0     1     0.0738      1     0.0342      1     -0.05563    1
      0.08764     1     0.19651     1     0.20328     1     0.12785     1
      0.10561     1     0.27087     1     0.44758     1     0.4175      1
      0.20033     1     0.36743     0.95603     0.48641     1     0.32492
      1     0.46712     1
1     0     0.99645     0.06468     1     -0.01236    0.97811     0.02498
      0.96112     0.02312     0.99274     0.07808     0.89323     0.10346
      0.94212     0.05269     0.88809     0.1112      0.86104     0.08631
      0.81633     0.1183      0.83668     0.14442     0.81329     0.13412
      0.79476     0.13638     0.7911      0.15379     0.77122     0.1593
      0.70941     0.12015     1
0     0     0     0     -1    -1    1     1     -1    1     -1    1     1
      -1    1     1     -1    -1    -1    1     1     -1    -1    1     -1
      1     1     -1    -1    1     -1    -1    1     -1    -1
```

```
1     0     1     -0.00612    1     -0.09834    1     -0.07649    1     -
0.10605     1     -0.11073    1     -0.39489    1     -0.15616    0.92124
      -0.31884    0.86473     -0.34534    0.91693     -0.44072    0.9606
      -0.46866    0.81874     -0.40372    0.82681     -0.42231    0.75784
      -0.38231    0.80448     -0.40575    0.74354     -0.45039    1
0     0     1     1     0     0     0     0     -1    -1    0     0     0
      0     -1    -1    -1    -1    -1    1     -1    1     0     0     0
      0     1     -1    -1    1     -1    1     -1    1     -1
1     0     0.96071     0.07088     1     0.04296     1     0.09313
      0.90169     -0.05144    0.89263     0.0258      0.8325      -0.06142
      0.87534     0.09831     0.76544     0.0028      0.75206     -0.05295
      0.65961     -0.07905    0.64158     -0.05929    0.55677     -0.07705
      0.58051     -0.02205    0.49664     -0.01251    0.5131      -0.00015
      0.52099     -0.00182    1
0     0     -1    1     0     0     0     0     -1    1     1     1     0
      0     0     0     1     -1    -1    1     1     1     0     0     -1
      -1    1     -1    1     1     -1    1     0     0     -1
1     0     1     -0.06182    1     0.02942     1     -0.05131    1     -
0.01707     1     -0.11726    0.84493     -0.05202    0.93392     -0.06598
      0.6917      -0.07379    0.65731     -0.20367    0.9491      -0.31558
      0.80852     -0.31654    0.84932     -0.34838    0.72529     -0.29174
      0.73094     -0.38576    0.54356     -0.26284    0.64207     -0.39487
      1
0     0     1     -1    1     1     1     -1    1     1     1     -1    1
      -1    1     -1    1     1     1     1     1     1     1     -1    1
      1     1     1     1     1     1     1     1     -1    -1
0     0     -1    -1    1     1     1     -1    -1    1     1     -1    -1
      -1    0     0     1     1     -1    -1    1     -1    1     -1    1
      1     1     -1    1     -1    -1    1     1     -1    -1
1     0     1     -0.14236    1     -0.16256    1     -0.23656    1     -
0.07514     1     -0.2501     1     -0.26161    1     -0.21975    1     -
0.38606     1     -0.46162    1     -0.35519    1     -0.59661    1     -
0.47643     0.9882      -0.49687    1     -0.7582     1     -0.75761    1
      -0.84437    1
1     0     0.88208     -0.14639    0.93408     -0.11057    0.921 -0.1645
      0.88307     -0.17036    0.88462     -0.31809    0.85269     -0.31463
      0.82116     -0.35924    0.80681     -0.33632    0.75243     -0.47022
      0.70555     -0.47153    0.6615      -0.50085    0.61297     -0.48086
      0.56804     -0.54629    0.50179     -0.59854    0.47075     -0.57377
      0.42189     -0.58086    1
1     0     1     -0.15899    0.72314     0.27686     0.83443     -0.58388
      1     -0.28207    1     -0.49863    0.79962     -0.12527    0.76837
      0.14638     1     0.39337     1     0.2659      0.96354     -0.01891
      0.92599     -0.91338    1     0.14803     1     -0.11582    1     -
0.11129     1     0.53372     1     -0.57758    1
1     0     0.66161     -1    1     1     1     -0.67321    0.80893     -
0.40446     1     -1    1     -0.89375    1     0.73393     0.17589
      0.70982     1     0.78036     1     0.85268     1     -1    1
      0.85357     1     -0.08571    0.95982     -0.3625     1     0.65268
      1     0.34732     -1
1     0     1     0.00433     1     -0.01209    1     -0.0296     1     -
0.07014     0.97839     -0.06256    1     -0.06544    0.97261     -0.07917
      0.92561     -0.13665    0.94184     -0.14327    0.99589     -0.14248
      0.94815     -0.13565    0.89469     -0.20851    0.89067     -0.17909
```

```
      0.85644     -0.18552    0.83777     -0.20101    0.83867     -0.20766
      1
0     0     1     1     1     -1    0     0     0     0     -1    -1    0
      0     0     0     -1    1     1     1     -1    1     -1    1     1
      -1    1     1     -1    1     1     1     0     0     -1
1     0     0.91241     0.04347     0.94191     0.0228      0.94705
      0.05345     0.93582     0.01321     0.91911     0.06348     0.92766
      0.12067     0.92048     0.06211     0.88899     0.12722     0.83744
      0.14439     0.80983     0.11849     0.77041     0.14222     0.75755
      0.11299     0.7355      0.13282     0.66387     0.153 0.70925
      0.10754     0.65258     0.11447     1
1     0     1     0.06538     1     0.20746     1     0.26281     0.93051
      0.32213     0.86773     0.39039     0.75474     0.50082     0.79555
      0.52321     0.65954     0.60756     0.57619     0.62999     0.47807
      0.67135     0.40553     0.6884      0.34384     0.72082     0.27712
      0.72386     0.19296     0.70682     0.11372     0.72688     0.0699
      0.71444     1
1     0     -1    -1    1     1     1     -0.14375    0     0     -1    1
      1     1     0.17917     -1    -1    -1    0.0875      -1    1     -1
      -1    1     -1    -1    1     -1    -1    -1    1     1     0     0
      -1
0     0     1     1     1     -1    1     1     1     1     1     1     1
      -1    1     1     1     -1    1     -1    1     1     1     1     1
      -1    1     1     1     1     1     1     1     1     -1
1     0     -0.64286    -1    1     0.82857     1     -1    1     -0.23393
      1     0.96161     1     -0.37679    1     -1    1     0.13839     1
      -1    1     -0.03393    -0.84286    1     0.5375      0.85714     1
      1     1     -1    1     -1    1     -1    -1
1     0     0.99025     -0.05785    0.99793     -0.13009    0.98663     -
0.1943      0.99374     -0.25843    0.92738     -0.3013     0.92651     -
0.37965     0.89812     -0.43796    0.84922     -0.52064    0.87433     -
0.57075     0.79016     -0.59839    0.74725     -0.64615    0.68282     -
0.68479     0.65247     -0.73174    0.6101      -0.75353    0.54752     -
0.80278     0.49195     -0.83245    1
1     0     1     -0.0373     1     -0.07383    0.99601     -0.11039
      0.99838     -0.09931    0.98941     -0.13814    0.96674     -0.21695
      0.95288     -0.25099    0.91236     -0.344      0.90581     -0.32152
      0.89991     -0.34691    0.87874     -0.37643    0.86213     -0.4299
      0.83172     -0.43122    0.81433     -0.42593    0.77919     -0.47977
      0.75115     -0.50152    1
1     0     0.98166     0.00874     0.98103     -0.03818    0.97565     -
0.05699     0.95947     -0.06971    0.99004     -0.04507    0.94713     -
0.11102     0.93369     -0.1279     0.94217     -0.11583    0.79682     -
0.192 0.88274     -0.17387    0.86257     -0.18739    0.88487     -0.19689
      0.81813     -0.21136    0.78546     -0.23864    0.76911     -0.23095
      0.74323     -0.23902    1
1     0     1     1     1     1     0.9101      1     -0.2697     1     -
0.83152     1     -1    1     -1    0.72526     -1    -0.57779    -1    -
0.42052     -1    -1    -0.52838    -1    0.90014     -1    1     -1    1
      -1    1     -0.34686    1     0.34845     1
1     0     -0.67935    -1    -1    1     1     0.63317     0.03515     -1
      -1    -1    1     1     0.88683     -1    -1    1     0.8384      1
      1     -1    -1    -1    -0.18856    1     1     -1    -1    -1    -1
      1     1     0.33611     -1
```

```
1     0     0.95659     0.08143     0.97487     -0.05667    0.97165     -
0.08484     0.96097     -0.06561    0.94717     0.01279     0.95436     -
0.16795     0.94612     -0.19497    0.9963      -0.32268    0.90343     -
0.35902     0.91428     -0.27316    0.9014      -0.29807    0.99899     -
0.40747     0.87244     -0.34586    0.92059     -0.30619    0.83951     -
0.39061     0.82166     -0.41173    1
0     0     1     -1    1     1     1     -1    1     1     1     -1    1
      1     1     -1    1     -1    1     1     1     1     1     -1    1
      -1    1     -1    1     1     1     -1    1     1     -1
1     0     1     0.05812     0.94525     0.07418     0.99952     0.13231
      1     -0.01911    0.94846     0.07033     0.95713     0.14644
      0.94862     0.11224     0.90896     0.20119     0.96741     0.16265
      0.99695     0.14258     0.90784     0.1641      0.91667     0.22431
      0.88423     0.23571     0.88568     0.22511     0.78324     0.29576
      0.83574     0.31166     1
```

Appendix B

MATLAB Codes

Variance approach

```
M3 = load('ionosphere.txt')
 V = var(M3)

[r, n] = size(M3)
k = 32;
l = [];
l2 = [];
t = 0;
while t < k
    max = 0;

    for i = 1 : n
        x = V(i);
        if (x > max) && not(ismember(i,l2))
            max = x;
            p = i;
        end
    end
    l = [l, max]
    l2 = [l2, p]
    t = t + 1
end
l2
l
ls = sort(l2)
DR = M3(:,ls)
```

Singular Value Decomposition

```
q = 4;
D = load('D.txt')
[U, S, V] = svd(D);
DR = D'* U(:,1:q)
```

Random Projection

```
D = load('ionosphere.txt')
R1 = round(1000*rand(34,32));
for j = 1 : 34,
   for k = 1 : 32,
      if (R1(j, k) >= 0) && (R1(j, k) < 666)
         R(j, k) = 0;
      elseif (R1(j, k) >= 666) && (R1(j, k) < 833)
         R(j, k) = -1;
      else
         R(j, k) = 1;
```

```
      end;
    end;
end;
DR = D * R
```

New Random Approach

```
D = load('ionosphere.txt')
%R1 = round(30*rand(1,18))
x = []
y = 0
while y < 32
    r = round(34*rand(1))
    s = ismember (r,x)
if (s == 0)

    x = [x r]
    y = y + 1
end
end
l = sort(x)
DR = D (:, l)
```

PCA

```
q = 32;
D = load('ionosphere.txt')
[U, S, V] = svd(D)
DR = D * V(:,1:q)
```

Modified New Top-Down

```
D = load('ionosphere.txt');
D = D(:,[1:8]);
 [r,c] = size(D);
AML = [];
p = 7;
t = c;
Lx = [1:t];
while t > p,

   t1 = t - 1;
   LC = Combinations(Lx, t1);

   for i = 1 : t,
      L = LC(i,:);
      Di  = (D(:, L));
      AMi = dist_preserve(D, Di)
      AML = [AML, AMi];
   end;
    M = max(AML)
   i = 1;
   while AML(i) < M
      i = i + 1;
```

```
   end;
   L = LC(i,:);
   t = t1;
   AML = [];
   Lx = L
end;
L
L = sort(L)
 D = D(:, L)
```

New Top-Down

```
D = load('ionosphere.txt');

D = D(:,[26:30]);

[r,c] = size(D);

D1 = kmeans1(D);

AML = [];
p = 4;
t = c;
Lx = [1:t];
while t > p,

   t1 = t - 1;
   LC = Combinations(Lx, t1);

   for i = 1 : t,
      L = LC(i,:);
      Di  = kmeans1(D(:, L));
      AMi = testkmeans1(D1, Di)
      AML = [AML, AMi];
   end;
   M = max(AML)
   i = 1;
   while AML(i) < M
      i = i + 1;
   end;
     L = LC(i,:);
     t = t1;
   AML = [];
   Lx = L
end;

L
L = sort(L)
D = D(:, L)
```

Third Novel Approach

```
D = load('ionosphere.txt')
%D = D(:,1:20)
```

```
k = 26;
[r,c] = size(D);
D1 = kmeans1(D)
AML = [];
for i = 1 : c,
     Di  = kmeans1(D(:,i))
      AMi = testkmeans1(D1, Di)
      AML = [AML, AMi];
end;
AML
l = selectbestk(AML, k)

Dr = D(:,l)
```

Second Novel Approach

```
D1 = load('ionosphere.txt')
D = D1(:,1:5)
k = 4;
[r,c] = size(D);

dp = []
for i = 1:c,
    Di = D(:,i);
    adpi = dist_preserve(D,Di);
    dp = [dp, adpi];
end;

dp

l = selectbestk(dp, k)

Dr = D(:,l)
```

First Novel Approach

```
D = load('ionosphere.txt')
k = 32;
[r,c] = size(D)
gmid = [];

for i = 1 : c;
   Di = D(:,i);
   gmidi = computegG(D, Di);
   gmid = [gmid, gmidi]
end;
gmid
l = selectbestk(gmid, k)

Dr = D(:,l)
```

Modified New Random Approach

```
D = load('testing.txt')
x = []
y = 0

while y < 4
    z = ceil(6*rand(1))
    s = ismember (z,x)
        if (s == 0)
             x = [x z]
             y = y + 1
        end
end
l = sort(x)
DR = D (:, l)
```

Direct Approach

```
drnewx2;

[r,c] = size(M3x);

i = 1; found = 0;

while (i <= r) && not(found)
    if M3x(i,k3) ~= Md
        i = i + 1;
    else
        found = true;
    end
end

l = M3x(i,1:k)
DR = D(:,l)
```

Combined Approach

```
drnewx2;

[r,c] = size(M2x);

i = 1; found = 0;

while (i <= r) && not(found)
    if M2x(i,k3) ~= Mc
        i = i + 1;
    else
        found = true;
    end
end

l = M2x(i,1:k)

DR = D(:,l)
```

The Perceptron

```
tstart=tic;
D = load('DRrp.txt')

[r,c] = size(D)
D1 = D(:,1:c-1)
w = ones(1,c-1)
changes = 1

while changes == 1
    changes = 0
    for i = 1:r
        if D(i,c) == 1
            x = D1(i,:)
            s = dot(w,x)
            if s<=0
                w = w + x
                changes = 1
            end
        end
    end
    for i = 1:r
        if D(i,c) == -1
            x = D1(i,:)
            s = dot(w,x)
            if s>0
                w = w - x
                changes = 1
            end
        end
    end
end
w
tElapsed=toc(tstart)
```

tclassify

```
w = [-6,1,11.0591900000000,9.57016000000001,-
0.325250000000002,10.2327600000000,-8.25832000000000,-
0.715139999999995,15.2152000000000,-4.12394000000001,-1.27723000000000,-
2.13797000000000,-
2.28207000000000,0.598960000000001,1.55706000000000,2.17638000000000,2.976310
00000000,-2.74741000000000,-3.56819000000000,-
12.0506900000000,10.6959900000000,9.12442000000001,-
6.86902000000001,10.5148299999999,-22.7577300000000,-
4.55862000000000,13.1937300000000,-2.35142000000000,17.0259400000000,-
0.208099999999990,-12.2587500000000,3.24763000000000]
Res = []
D = load('drnbup32.txt')
%D = D(:,1:15)
%R = Dx(:,16)
for i = 71 : 100
    x = D(i, :);
    s = dot(w, x);
    if s > 0
```

```
        t = 1
    else
        t = -1
    end;
    Res = [Res, t];
end
```

K-Nearest Neighbor

```
D1 = load('ionosphereknn.txt')
%D1 = D1(:,[1,2,4,7,8,9,11]);
U = load('ionosphere.txt')
D2 = U(71:100,:)
%D2 = D2(:,[1,2,4,7,8,9]);

D3 = [];
k = 5;
cl = 2;
 [r1, c1] = size(D1);
 [r2, c2] = size(D2);
R = zeros(1,cl);
r3 = r2;
for i = 1 : r2,
    L = [];
    X = D2(1,:);
    for j = 1 : r1,
        c = c1 - 1;
        Y = D1(j, 1 : c);
        S = sum((X - Y).^2);
        L = [L; [S j]];
    end;
    L = sort3(L, 1)

    for  q = 1 : k,
         z = L(q, 2);
         m = D1(z, c1);
       R(m)= R(m) + 1;
    end;
    M = max(R);
    for p = 1 : cl,
        if R(p) == M
            class = p;
        end;

    end;

    D3 = [D3; [X,class]];
    D2 = D2(2:r3, :);
    r3 = r3 - 1;
    R = zeros(1,cl);
end
D1
D2
```