Comparative study of state-of-the-art machine learning models for analytics-driven embedded systems

Master of Science Thesis

University of Turku
Department of Future Technologies
Faculty of Science and Technology

2019

Sushri Sunita Purohit

Reviewers:

Ph.D. (Tech.) Tomi Westerlund
Prof. Tapio Pahikkala

UNIVERSITY OF TURKU
Department of Future Technologies

Sushri Sunita Purohit: Comparative study of state-of-the-art machine learning models for analytics-driven embedded systems

Master of Science Thesis, 63 p., 5 app. p.
Faculty of Science and Technology
March 2019

Analytics-driven embedded systems are gaining foothold faster than ever in the current digital era. The innovation of Internet of Things(IoT) has generated an entire ecosystem of devices, communicating and exchanging data automatically in an interconnected global network. The ability to efficiently process and utilize the enormous amount of data being generated from an ensemble of embedded devices like RFID tags, sensors etc., enables engineers to build smart real-world systems. Analytics-driven embedded system explores and processes the data in-situ or remotely to identify a pattern in the behavior of the system and in turn can be used to automate actions and embark decision making capability to a device. Designing an intelligent data processing model is paramount for reaping the benefits of data analytics, because a poorly designed analytics infrastructure would degrade the system's performance and effectiveness. There are many different aspects of this data that make it a more complex and challenging analytics task and hence a suitable candidate for big data. Big data is mainly characterized by its high volume, hugely varied data types and high speed of data receipt; all these properties mandate the choice of correct data mining techniques to be used for designing the analytics model. Datasets with images like face recognition, satellite images would perform better with deep learning algorithms, time-series datasets like sensor data from wearable devices would give better results with clustering and supervised learning models. A regression model would suit best for a multivariate dataset like appliances energy prediction data, forest fire data etc. Each machine learning task has a varied range of algorithms which can be used in combination to create an intelligent data analysis model.

In this study, a comprehensive comparative analysis was conducted using different datasets freely available on online machine learning repository, to analyze the performance of state-of-art machine learning algorithms. WEKA data mining toolkit was used to evaluate C4.5, Naïve Bayes, Random Forest, kNN, SVM and Multilayer Perceptron for classification models. Linear regression, Gradient Boosting Machine(GBM), Multilayer Perceptron, kNN, Random Forest and Support Vector Machines (SVM) were applied to dataset fit for regression machine learning. Datasets were trained and analyzed in different experimental setups and a qualitative comparative analysis was performed with k-fold Cross Validation(CV) and paired t-test in Weka experimenter.

Keywords: Embedded system analytics, IoT, Data mining, Machine learning, WEKA

Acknowledgements

I would like to thank my thesis advisors Prof. Tapio Pahikkala and Prof. Tomi Westerlund for the constant support , guidance and their valuable comments on this thesis. I would also like to thank my work colleagues for supporting me and being patient with me while I took days off from work to complete this thesis work. Finally I would like to thank my husband Bineet Panda and my family for being a constant source of motivation and encouragement. This accomplishment would not have been possible without them. Thank you.

At Vantaa 20.3.2019
Sushri Sunita Purohit

Table of Contents

List of figures

List of tables

List of acronyms

RFID	Radio Frequency Identification
IoT	Internet of things
XML	Extensible Markup Language
HDF	Hierarchical Data Format
CAN	Controller Area Network
WSN	Wireless Sensor Networks
TBDCS	Tree-Based Data Collection Scheme
FNS	Forwarding Nodes Set
MDL	Minimum Description Length
WEKA	Waikato Environment for Knowledge Analysis
SVM	Support Vector Machines
kNN	k-Nearest Neighbor
CV	Cross Validation
IG	Information gain
ID3	Iterative Dichotomiser 3
RF	Random Forest
GBM	Gradient Boosting Machines

ANN	Artificial Neural Networks
TP	True Positive
TN	True Negative
FN	False Negative
FP	False Positive
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
WSDN	Wireless Distributed Network
HAR	Human Activity Recognition
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
ARFF	Attribute-Relation File Format
CLI	Command Line Interface
RBF	Radial Basis Function
DELVE	Data for Evaluating Learning in Valid Experiments
CFS	Correlation-based Feature Selection

Chapter 1

1.1 Introduction

"*Information is the oil of the 21st century, and analytics is the combustion engine*"

Peter Sondergaard, Gartner

Integration of embedded systems into the field of electronics design has been increasingly ubiquitous, ranging from modern everyday appliances like mobile phones, Radio Frequency Identification (RFID) tags in home appliances, modems, remote controls, watches etc. to complex system development like automobiles, space researches, power plants etc. Internet of things (IoT) has also emerged as an evolving technology which has found its place in embedded electronics domain. With the rapid growth and advancement in IoT and embedded electronics, a network of interconnected devices has been created, which has revolutionized the concept of networking and data communication but it comes with a price, the challenge of handling the massive amount of data these interconnected applications generate. With increasing demand for energy efficient, reliable, scalable and faster applications, the need for turning data into useful information and knowledge has attracted a great deal of attention in the research community, which has led to the emergence of analytics-driven system design.

Designers combine the data mining techniques into the embedded system design to make the application context-aware and be able to predict the system's behavior. The implementation of analytics differ depending on the system usage, in some cases, the analytics is performed in the cloud to improve the embedded-systems performance while in some it runs directly in the embedded system. The initial step in the direction of analytics design is the selection of the appropriate data mining technique to ensure system robustness, reliability and efficient cost management in the architecture, design, and maintenance.

In this thesis, a qualitative comparative analysis of state-of-art data mining algorithms using datasets generated by embedded devices is presented. The objective of this study is to identify the optimal data mining models specific to the dataset.

1.2 Organization of thesis

The rest of the thesis has been organized as follows. Chapter 2 focuses on the theoretical foundations of analytics-driven embedded system design. The first part of the chapter covers the generalized workflow of the analytics design for embedded systems, here different steps and the tasks performed during the process is explained. The second part of the chapter covers the basic concept and principle of the state-of-art machine learning algorithms. In chapter 3, dataset chosen for the study and the toolkit used are described and an overview of the experimental setup is illustrated. Chapter 4 covers experimentation in detail and initial result of the analysis. Finally, in chapter 5 the t-test analysis results are scrutinized and concluding remarks are presented.

Chapter 2

2.1. Workflow in analytics-driven system design

Typically, the process of data mining starts with problem definition which sets the path for data modeling. The key aspects of a problem definition are clarity of the business requirement and a cost/benefit estimate [9]. Knowledge of business requirement and the way data can be used to achieve the desired goal is the stepping stone in the process of analytics. The next significant step is the collection of data which determine the rest of the step to be followed in the data modeling. A pictorial representation of the sequence of steps designers follow to build an embedded system analytics model to accomplish the expected outcome is shown in Figure 2.1.

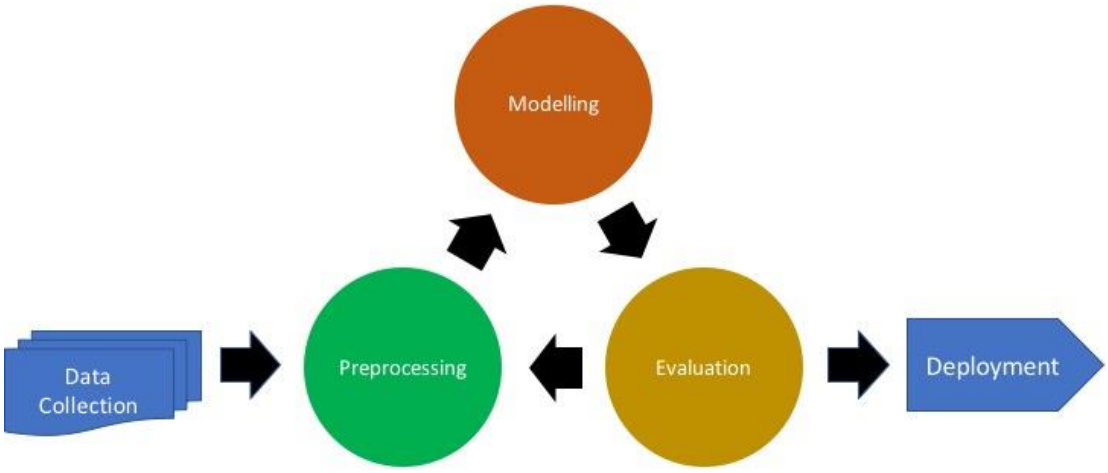


Figure 2.1: Analytics-driven system design workflow

2.1.1. Data collection

Applying a systematic approach to collecting the raw data is foundation step in the workflow. In embedded computing domain, data is being collected from a varied range of sources and in distinct forms. For smart devices embedded with electronics, software, sensors, actuators, each event detection generate data and can be of various formats such as text, spreadsheet, image, audio, video, geospatial, web, Extensible Markup Language (XML) and Hierarchical Data Format (HDF) for scientific data, and Controller Area Network (CAN) for automotive data. Data from multiple sources must be integrated and stored so that it is accessible for training the model. In case the IoT devices, multiple devices are part of a Wireless Sensor

Networks (WSN) creating a massive volume of data. Data collection schemes in devices connected through WSN or any embedded devices that operate in-situ must fulfill following objectives: a) minimal energy consumption, b) minimized latency and c) optimized CPU usage [3].

Different data collection methodologies have been proposed keeping the mentioned goals in mind. Data aggregation is a basic operation in WSN since it is typically restricted in hardware infrastructure and communication recourses. Tree-based technique and cluster-based technique are the two most commonly used data collection methodologies adopted in embedded systems analytics workflow. As proposed in [4], Tree-Based Data Collection Scheme(TBDCS) is a distributed data collection method which establishes a tree structure with intermediate nodes termed as Forwarding Nodes Set(FNS). These intermediate nodes act like data aggregators to transmit data from sensor nodes back along the tree. Through different simulations and scrutiny, it has been inferred that TBDCS significantly reduces transmission latency and network congestion.

Cluster-based data collection method has been claimed to be effective in systems which are limited in energy consumption and network bandwidth [5]. In this technique, a cluster head is designated among a group of sensor nodes and similarly multiple disjoints sets are formed with a group of sensor nodes. Sensors in the cluster send information to the cluster head, The cluster head suppresses the local redundancies and communicates the compressed data to the main system. In this process, the cost of sending redundant data is reduced thereby preserving energy and minimizing the scalability constraint.

Another challenge of data collection is the structure of data that determine various characteristics of the dataset such as multivariate, sequential, time-series, image signals etc. In addition to that, each dataset can contain a heterogeneous list of attributes. All these aspects of data steer the choice of machine learning technique to be used for analysis.

2.1.2. Data pre-processing

The bitter truth of data science is the data collected from real-world applications are not machine learning ready. Real data is low in quality in terms of completeness, accuracy, and consistency, hence unreliable to be used to design an intelligent analytics model. The data collected needs to be pre-processed before applying any machine learning algorithms. Pre-

processing the data has become the most essential and expensive step in the data mining workflow. Due to growing demand for smart systems with multiple data generating nodes, designing a reliable and high performing predictive model has become paramount. Pre-processing is part of the iterative data mining workflow and are optimized by multiple runs of the workflow and are greatly determined by the statistics generated by the algorithms used to train the data. One must undergo different phases of data pre-processing to achieve a predictive data model.

The first stage of preparing the data is to understand the features of the raw data to identify outliers, noise etc. Descriptive data summarization reveals the raw data characteristics. Measurement of central tendency and dispersion of data are standard techniques used to describe the anomalies in data. Mean, median, mode, and midrange are properties that define the central tendency of a dataset and measures of quartiles, interquartile range and variance determine how disperse the data is [7]. During this process, the main features of the data can be identified which are essential for proceeding to the next stages [6]

Another phase of preparing the data is feature selection. Raw data contain magnitude of attributes and it is crucial to filter out those attributes which do not have a relevant contribution to the machine learning process. Discretization is one way of transforming the data to suit some of the specific classification and clustering algorithms. In this process, a large number of data values (numeric attributes) are converted into a smaller number of discrete values. These methods are used for attribute reduction by grouping the continuous attributes into a range of values. Applying discretization to data enhances the predictive accuracy and boost the performance as well. Discretization algorithms can be categorized as boolean reasoning, equal frequency binning, entropy-based discretization [7]. Depending on the complexity of the raw data, additional data transformation is needed to optimize the training process. Projection is one of the technique used to project training data such as spatial data in lower dimensional spaces, but still preserves the inherent relationships in the data. Principal component analysis another way of transforming data into a set of values of linearly uncorrelated variables called principal components. Section 2.2 covers the details of some of the popular pre-processing algorithms.

2.1.3. Data modeling

After the data has passed the initial pre-processing stage, mining model is built using the transformed data and machine learning algorithms. This model forms the base to extract patterns. Discovery of patterns is mainly determined by the selection of training data, type of algorithms used and how the algorithm is configured[8]. Different dataset require specific machine learning tasks, for example, deep learning algorithms should be used in case of complicated dataset like face recognition, time-series datasets like sensor data from wearable devices need clustering and classification learning models, multivariate dataset like appliances energy prediction data, forest fire data etc. might work best with regression learning tasks.

Classification

Classification machine learning is used to predict discrete data points or class labels for the data instance based on the prediction model called classifier. A classifier is constructed by training on the dataset through a series of machine learning tasks. The simplest classification type is of a binary class labeled data model. In binary classification the observed instance can only be categorized into two class labels. Classification algorithms find relationship between the value the attributes which are identified as predictors and a discrete output value such as color, type or true/false, using the training set. The model is designed based on these relationships and then applied to the real time dataset to predict the class label.

Regression

Regression data modeling determine relationships between the value the attributes which are identified as predictors and a continuous output value. A continuous output variable is a real-value, such as an integer or floating point value such as size, amount etc. There are several machine learning algorithms which can be used for both regression and classification such as decision trees, support vector machine and artificial neural networks but there some specific algorithms which are only meant for regression like linear regression and addictive regression.

Clustering

Clustering is used when the class label is unknown or not certain. The aim is to segregate dataset instances with similar traits into groups and categorize them into clusters by dividing the data instances in the dataset into a number of groups such that data points within a group are more similar to other data points in the same group than those in other groups.

A mining model is generated by performing a series of adjustments to the algorithms and data creating different results which are then evaluated and compared to select the optimal analytics setup. During this process, both the mining structure and mining model are updated after each adjustment. Dataset structure mainly consists of data source definition, list of instances and list of features. A mining model architecture consists of metadata, data mining results in form of patterns and a data binding structure to the original data. The metadata gives the specifics of the model such as the name of the model, mining structure used for the model, machine learning algorithms used to analyze the data. Each model is characterized by two properties. Algorithm property defines the algorithm that is used to create the model. It can be set during the data analysis phase and can be changed later but the model must be reprocessed to generate the accurate pattern. Another property is the usage which specifies how each attribute is used by the model [8].

Each machine learning task has a varied range of algorithms which can be used in combination to create an intelligent data analysis model. Section 2.2 covers the concepts of some of the state-of-art algorithms.

2.1.4. Data evaluation

Data evaluation is the key step to ensure the credibility of a model and is a cumbersome process. Data evaluation begins by setting up a systematic evaluation method to explore how different models structure the data and do a comparative analysis. Selection of a discrete test data set is pre-requisite for trustworthy evaluation since performance on the training set is not a good indicator of future performance. The reason is that the model has been trained from the very same training data and an estimate of performance based on that data will be optimistic and not realistic.

Over the years, different evaluation methods have been proposed for different category of data mining models. Cross-validation, hold out and random subsampling and bootstrap are the most commonly used methods for classification and regression models. Evaluation schemes assess the performance on multiple metrics such as sensitivity and specificity, precision, kappa statistics, mean absolute error, root mean squared error, relative absolute error and root relative squared error. Cross-validation is the preferred method of choice of data scientists. For clustering schemes, Minimum Description Length(MDL) is considered for evaluation. To

find the best data mining algorithm to design the data model, a comparative analysis is needed to predict the true performance of different algorithms. The t-test is commonly used to perform such experimentation. Details of the evaluation schemes used in this study have been covered in section 2.2. However MDL is beyond the scope of this study and hence not covered.

2.1.5. Model deployment

The last step in the data mining process is to integrate the analytics model into a commercial environment. The model can then be used to perform multiple tasks such as real-time object detection, tracing patterns from new signals, making predictions to direct business decisions, creating queries to retrieve new patterns, rules, and behavior.

2.2. Conceptual overview of data mining algorithms

2.2.1. Data transforming algorithms

Feature selection

Feature selection is a form of data reduction where irrelevant, or redundant attributes can be detected and removed. Algorithms are categorized into a scheme-independent selection (filter method) and scheme-specific selection (Wrapper method) [10]. In WEKA interface, the process of attribute selection is split into two parts: (a) attribute evaluator- the process of assessing the selected attribute subset (b) search method- the process of searching a possible subset.

2.2.1.1. Ranker

This algorithm works on the principle of information gain attribute ranking. It is one of the simplest attribute selection method which ranks attributes by their individual evaluations and mostly used in decision tree classifiers like C4.5 classifier. Information gain(IG) is based on entropy metrics. The entropy measure is a measure impurity and can be calculated as $H(X)$ (Equation 2.1).

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2(p(x_i)) \quad (2.1)$$

where p(x) is the marginal probability density function of a discrete random variable X with N outcomes. IG is measured by the amount of information gained by splitting the dataset using the chosen attribute whereas entropy of the class reflects the attribute contribution in gaining clear information about a class. Attributes are ranked based on IG value, where attributes with high scores are selected since they can be used for better prediction [10] [11].

2.2.1.2. Correlation-based Feature Selection (CFS)

Correlation-based Feature Selection method evaluates a subset of attributes instead of assessing individual attributes. Efficacy of individual features is considered for predicting the class along with the degree of inter-correlation among them. High scores are assigned to subsets containing attributes that are highly correlated with the class and poorly inter-correlated with each other. Scores are measured by heuristic merits formulated by the Equation 2.2 (Ghiselli 1964) [12].

$$r_{\theta S} = \frac{k\overline{r_{\theta}}_i}{\sqrt{k + k(k-1)\overline{r_{ij}}}} \quad (2.2)$$

where $r_{\theta s}$ is the score of the subset, $r_{\theta i}$ is the average correlation between the k variable and θ, and r_{ij} is the average of attribute subset intra-correlation.

2.2.1.3. Greedy stepwise

Greedy stepwise method selects the attributes by performing a forward or backward search through the attribute subspace which can be random. During the process of search the greedy stepwise method creates a ranked list of attributes and the search process stops when the addition or deletion of any remaining attributes results in a decrease in evaluation. Greedy stepwise feature search is used in conjunction with CFS in WEKA.

Discretization

As mentioned in section 2.1, discretization is mainly needed for datasets with continuous numeric attributes and can be achieved by sorting all the continuous values of the attribute

and splitting continuous values into a predetermined number of equal intervals (unsupervised discretization) or using the number of classes as the discretization parameter (supervised discretization). Unsupervised discretization methods are only suitable for clustering type of training where the class is non-existent or uncertain [7].

2.2.1.4. Boolean reasoning

Boolean reasoning builds on the Boole-Schröder algebra of logic, which is based on Boolean equations with predicates which are true or false. It is a straightforward implementation that filters a small subset of attribute values that do not preserve the discernibility. The remaining subset is a minimal set of cuts preserves the discernibility inherent in the dataset. The algorithm operates by first creating a Boolean function from the set of candidate cuts, and then computing a prime implicate of the Boolean function [7].

2.2.1.5. Entropy-based discretization

It is a form of supervised discretization which based on the MDL principle as described in [11]. Several entropies-based algorithms have been proposed which work for multiple domains. Some algorithms work on the principle of recursively partitioning the value set of each attribute so that the local measure of entropy is optimized and some are based on maximizing the entropy over discretization space [13].

Feature extraction

Also known as dimensionality reduction, feature extraction is a process of transforming the high dimensional dataset to a reduced or compressed representation of the original data before starting the modeling process. During the process of feature extraction, the original data can be either be transformed without losing any information (lossless data reduction) or the transformed data approximates the original data (lossy data reduction) [1]. In realistic data mining process, lossy data reduction is the typical outcome of as dimensionality reduction. Several algorithms have been invented for this purpose out which PCA and Random Projections are the preferred ones.

2.2.1.6. Principal Component Analysis

Principle Component Analysis(PCA) is a mathematical procedure which is based on projection principles. In this method, an orthogonal restructuring of a high dimensional data set to a set of values of linearly uncorrelated variables called principal components. PCA was developed and named by Harold Hotelling in 1930 which was based on the algorithm derived by Karl Pearson as an analog of the principal axis theorem in 1901[14]. PCA is done by deriving a covariance matrix of a data set consists of covariance values between all the different dimensions. Covariance indicates the relationship between two dimensions and can be calculated as

$$C(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{n-1} \quad (2.3)$$

where C is the covariance and X and Y are the two dimensions of a dataset which has n number of instances and $\overline{X}, \overline{Y}$ are the mean of the dimension X and Y. For an N-dimensional data set, covariance matrix will have $\frac{N!}{(N-2)!*2}$ a number of covariance values. Then the eigenvectors and eigenvalues of the covariance matrix are calculated and eigenvectors are normalized so that the length is always 1 The eigenvectors are then ranked according to the eigenvalues. The eigenvectors which have high scores are selected as principal components. The transformed data is derived from the original dataset by using the set of selected principal components called a feature vector, formulated in Equation 2.4.

$$D_{final} = FV_{trasposed} \times MD_{org}transposed \quad (2.4)$$

where FV is the feature vector matrix which row transposed and $\mathrm{MD_{org}}\mathrm{transposed}$ is the transposed matrix of mean adjusted values of the original dataset i.e. the data items are in each column, with each row holding a separate dimension [15].

2.2.1.7. Random Projection

Another technique of feature extraction is Random Projection(RP) where the original data is projected onto a predefined lower dimensional subspace using a random matrix whose columns have unit lengths. In RP, the original d-dimensional data is projected to a k-dimensional (k << d) subspace through the origin, using a random (k × n) matrix R whose columns have unit lengths. RP can be formulated in the Equation 2.5.

$$D_{k\times N}^{RP} = R_{k\times d} \times D_{d\times N} \quad (2.5)$$

where $D_{d\times N}$ is the original set of N n-dimensional instances, $D_{k\times N}^{RP}$ is the projection of the data onto a lower p-dimensional subspace. The key idea of random mapping arises from the Johnson-Lindenstrauss lemma [16] which states that distance relationships are preserved quite well on an average. Random projection implementation can be done in different ways. One is based on Gaussian distribution [17] and other implements a sparse random matrix as proposed by Achlioptas in [18].

2.2.2. Data mining algorithms

Machine learning algorithms are typically categorized into supervised and unsupervised algorithms based on the training dataset characteristics. A supervised method of learning can be applied to the dataset where the output (the desired result) is one of the attributes in the training dataset. Classification and regression algorithms are used for those cases. Classification models are based on predicting discrete class attribute value or a probability for a class attribute value whereas regression modeling is used to predict a continuous quantity which can be a real value or discrete integer variable. There are few algorithms which can be used for both classification and regressing modeling such as neural networks, decision trees but algorithms like linear regression and additive regression can be used only for regression type. Unsupervised algorithms are used in those cases where the output attribute is unknown or non-existent in the training dataset. Clustering is the most common technique used in unsupervised algorithms.

2.2.2.1. C4.5

C4.5 is an algorithm used for classification by generating decision trees. It was developed by Ross Quinlan as an extension to the Iterative Dichotomiser 3 (ID3) algorithm [2]. C4.5 is the most popular algorithm used for classification type data modeling. The basic principle of C4.5 involves building decision trees on the training data using the information gain entropy principle for attribute selection as explained in section 2.2.1.1.The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller partitioned data. Usually, a fully expanded decision tree reveals irrelevant structure so pruning method is applied. Pruning can be either forward pruning which is done

during the decision tree creation, or it can be backward pruning which is done after the decision tree has been built. C4.5 uses a backward pruning method by default. In some cases pruning causes a fall in the accuracy of the model prediction, thereby correct estimation of error rates is significant. C4.5 uses a default confidence level of 25% to calculate the error rate e in Equation 2.6 [2].

$$e = \frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}} \quad (2.6)$$

where N is the number of samples and f is the observed error rate and z is the number of standard deviation which is calculated using the confidence level.

2.2.2.2. Naïve Bayes

Naïve Bayes machine learning algorithm is a type of statistical classifier and is based on Bayes' theorem which works on the principle of conditional probability. In a practical scenario, datasets have many attributes therefore, use of simple Bayes' theorem would be computationally expensive, so Naïve Bayes makes an assumption of class conditional independence which means there are no dependency relationships among the attributes. Naïve Bayes classifier predicts that the data instance X belongs to a class if and only if,

$$P(C_i|X) > P(C_j|X) \ for\ 1 \leq j \leq m, i \neq j \quad (2.7)$$

where $P(C_i|X)$ is the probability of X belonging to class C_i , assuming that the training dataset has m classes. $P(C_i|X)$ is calculated as

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (2.8)$$

where $P(X)$ is the prior probability of X which is a constant value, $P(C_i)$ is the prior probability of C_i independent of its attribute values. $P(X|C_i)$ is the posterior probability of X conditioned on C_i and is calculated as

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \quad (2.9)$$

where x_k refers to the value of attribute A_k for instance X. In case of a continuous attribute value, $P(X|C_i)$ is calculated using the Gaussian distribution of the attribute value [1].

2.2.2.3. Random Forest

Random Forest(RF) is an ensemble machine learning method best suited for classification and regression tasks. An ensemble machine learning is a combination of multiple classifiers to get a better predictive efficiency. There are different types of ensemble methods such as error-correcting output coding, bagging, boosting and randomization[27]. RF algorithm was first developed by Leo Breiman [26] by combining his concept of bagging [28] and random subspace method introduced by Tin Kam Ho [29]. Leo Breiman defines RF as "A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta_k), k = 1, \dots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree cast a unit vote for the most popular class at input x" [26]. As described by Leo Breiman in [26] the error rate of random forest classification is dependent on correlation between any two trees in the forest. The error rate is directly proportional to correlation. RF is claimed to be an efficient machine learning method when a dataset is large and there is a high risk of overfitting due to missing data[26].

2.2.2.4. k-nearest neighbor(kNN)

k-nearest neighbor classifier is one of the basic classification technique used when the estimation of the parametric probabilities is not an easy process. It belongs to the family of instance-based machine learning methods. kNN was first proposed by Fix & Hodges in 1951. In kNN a lazy learning approach where the actual processing is done when a test instance is applied to the classification model. Assuming that the training dataset is described by *n* attributes, a k-nearest-neighbor classifier searches for patterns between the test data instance and the k training data instances based on the closeness in distance metrics. The distance metric, usually Euclidean distance between a test sample $X_1 = (x_{11}, x_{12}, x_{13} \dots x_{1n})$ and the given training samples $X_2 = (x_{21}, x_{22}, x_{23} \dots x_{1n})$ is taken into consideration to determine the closeness and can be calculated as Equation 2.10.

$$Dist(X_1X_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2} \quad (2.10)$$

kNN in its original form perform poorly in terms of accuracy due to its inherent property of assigning equal weight to the attributes, so the application of data transformation methods prior to classification modeling is imperative. kNN classifier also lacks in speed since it requires an iterative process to spot the K training instance. Transforming the training data into sorted search trees and employing parallel execution reduces the comparison time[1].

2.2.2.5 Linear Regression

Linear regression method is typically used for modeling dataset with numeric attributes and numeric class prediction. Linear regression models assert response variable y_i as a linear function of x_i is the weighted attribute value. y_i for i^{th} instance can be calculated as

$$y_i = w_0 + \sum_{i=1}^{k} w_j x_j \quad (2.11)$$

where w_0 , w_j are the weights which are used as regression coefficients and are computed by the least square method to find the best fitting values which minimize the difference between the actual class and the predicted class y_i. The overall minimization can be calculated by taking the sum of squares of differences for all instances and the goal is to have a minimum value for better prediction accuracy. Linear regression models are limited to datasets which exhibit linear dependency and only support regression type problems [1][2].

2.2.2.6 Gradient Boosting Machines (GBM)

Gradient boosting works on the notion of a week classifier can be enhanced to give better results. Friedman introduced the Gradient Boosting Machines to conceptualize this idea[41]. Elements involved in gradient boosting are loss function such as least square that is optimized during the learning process, a base learner such as decision tree ,which makes the predictions and an addictive model to which the base learner is sequentially fitted in order to minimize

the loss function. Friedman in [41] analyses that accuracy and the execution speed of gradient boosting can be enhanced by adding randomization to the sampling procedure. For a given training sample D $\{y_i, x_i\}^N$ where X is the set of attributes, the goal is to find a function $F^*(x)$ that maps x to y such that over the joint distribution of all values, the expected value of some specified loss function $\psi(y, f(x))$ is minimized as shown in Equation 2.12.

$$F^*(x) = arg \min_{F(x)} E_{y,x}\ \psi(y, f(x)) \qquad (2.12)$$

Boosting constructs the addictive model by approximating $F^*(x)$ as shown in Equation 2.13.

$$F(x) = \sum_{m=0}^{M} \beta_m\, h(x; a_m) \qquad (2.13)$$

where $h(x; a_m)$ is the base learner with parameters $a = \{a_1, a_2, \ldots a_m\}$ and β_m is the expansion coefficient and M is the number of iterations.

The generic gradient boosting is greedy algorithm this prone to overfitting . To minimize the overfitting some parametric modification such as tree depth, learning rate and number of random samples can be done in the algorithm. It is a common practice to have the tree depth between 4-8 levels and learning rate between 0.1 to 0.3.

2.2.2.7. Support Vector Machines (SVM)

Support Vector Machines is one of the most popular machine learning algorithm which is widely used for both classification and regression modeling tasks. The reason for SVM's high acclamation is its capability to provide a robust and efficient algorithm which can handle both linear and nonlinear data. The standard algorithm that is widely used today was proposed by Corinna Cortes and Vapnik in 1993 but its origin dates back to 1963 [32]. The basic principle of SVM is to construct a set of hyperplanes called support vectors and then a linear model is built on the nonlinear hyperplanes. In case of a two-class learning task, the goal is to identify the best classification function to separate data instances in the training dataset. The separation function corresponds to a hyperplane separating dataset into two classes. In SVM, the best separating hyperplanes can be identified by maximizing the margins between the classes. Margin can be deduced geometrically as the shortest distance between the nearest

data points called support vectors, to the hyperplane. The maximum margin hyperplanes can be defined by Equation 2.14[2]

$$x = b + \sum_{\substack{i \text{ is the} \\ support\ vector}} \alpha_i y_i \, (a_i \cdot a) \tag{2.14}$$

where y_i is the class label, $a_i \cdot a$ is the dot product of training data vector a_i and test data vector a, α_i and b are coefficients determined during the training process using constrained quadratic optimization. In case of the nonlinear dataset, the hyperplane function can be derived by extending the dot product $a_i \cdot a$ to a kernel functional mapping $\Phi a_i \cdot \Phi a$ where Φ is the function that projects the data into a transformed higher dimension [2]. Several kernel functions have been proposed in recent years which optimizes the SVM classification. To handle multiclass classification, the coefficient α_i is generalized. Many variations of SVM have been proposed to handle different machine learning tasks. LIVSVM is an open source library which was developed by Chang, Chih-Chung; Lin, Chih-Jen in 2000[33], which includes support vector classification, regression and one-class SVM.

2.2.2.8. Artificial Neural Networks

Artificial Neural Networks (ANN) are complex data modeling systems which are inspired by the core concept of biological neurons. An ANN is composed of connected input/output processing unit each with an associated weight. The individual units are called nodes or neurons and are based on the basic computation model proposed by McCulloch and Pitts as shown in Figure 2.2.

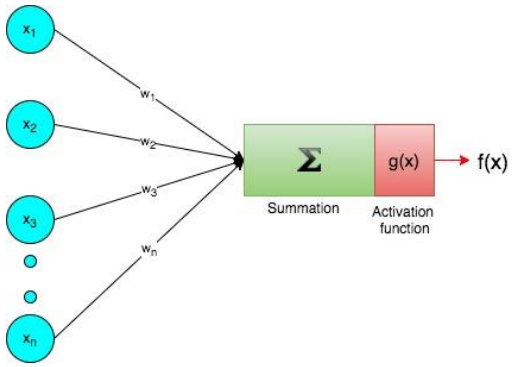


Figure 2.2: Artificial neuron computational model

The output can be mathematically computed in Equation 2.15 as the weighted sum of its n input signals, $x_{j=\{1,2,\ldots n\}}$ and it generated the output as 1 if the weighted sum is above a certain threshold u, else output is 0 [32].

$$f(x) = \theta\left(\sum_{j=1}^{n} w_j\, x_j \; - u\right) \tag{2.15}$$

where $\theta(.)$ is the activation or transfer function, w_j is the weight of the input. The purpose of activation function is to non-linearize the neural network and generalize the neurons. The basic architecture of ANN consists of layers of interconnected neurons and connections with associated weights as shown in Figure 2.3. Performance is improved over time by iteratively updating the weights in the network.

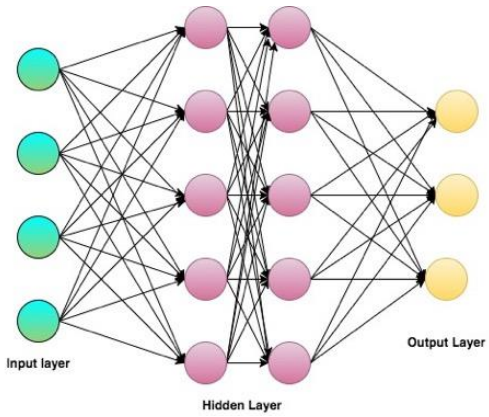


Figure 2.3: ANN architecture

Depending on the architecture, ANN can be categorized into feedforward or feedback(recurrent) networks. The process of learning in ANN involves updating the architecture and the connection weights. Feedforward networks are associated with supervised learning and feedback networks are associated with unsupervised learning. In feedforward network, there is an input layer, one or multiple hidden layers and an output layer and lines connecting the neurons have an associated weight with it as shown in the figure. The inputs to the network correspond to the attributes of training data instance.

Multilayer Perceptron

A multilayer perceptron is a class of feed-forward neural network and utilizes backpropagation algorithm for learning. During the training phase, backpropagation learns by iteratively processing the dataset instance and comparing the network output to the class

attribute value and subsequently adjusting the weight to reduce the mean squared error between the predicted value and actual value by using a gradient-descent method. The algorithm is called backpropagation because the adjustment is made in backward direction i.e. from the output layer down to hidden layers. The squared error in the j^{th} node of the output layer for a can be determined as

$$E = \frac{1}{2}\sum_{j}(y_j - f(x_j))^2 \qquad (2.16)$$

Here *y* is the class label of the instance and *f(x)* is value produced by the output layer. With gradient descent, the connection weight of j^{th} and i^{th} neurons is adjusted by

$$\Delta w_{ji} = -\eta \frac{dE}{dg(x_j)}(f(x_i)) \qquad (2.17)$$

where $f(x_i)$ is the output of the previous neuron, η is the learning rate and. The derivative can be calculated as

$$-\frac{dE}{dg(x_j)} = \emptyset'(g(x_j))\sum_{k} -\frac{dE}{dg(x_k)} w_{kj} \qquad (2.18)$$

with $\emptyset'$ being the derivative of activation function and can be seen that the derivate depends on the depends on the change in weights of the k^{th} node in output layer and thus the change of weight in the hidden layer is the back propagated according to the derivative of the activation function [34].

Due to its capability to handle non-linearity and versatile nature, multilayer perceptron has its roots in varied machine learning tasks such as image recognition, speech recognition, regression modeling etc.

Self-Organizing Maps(SOM)

Self-Organizing Maps also known as Kohonen's SOM, is a class of feedforward artificial neural network which uses unsupervised mode of machine learning and is based on the principle of topographic map formation. SOM was first introduced by Prof. Kohonen in

1984[35]. The principal objective of SOM is to define a high dimensional dataset in one or two-dimensional data space while preserving the topographic relationships of data. SOM has a single computational layer consisting of a grid of neurons arranged in rows and columns and an input layer consisting of an input vector. Each neuron in the computational layer is connected to all the source nodes in the input layer as shown in below Figure 2.4.

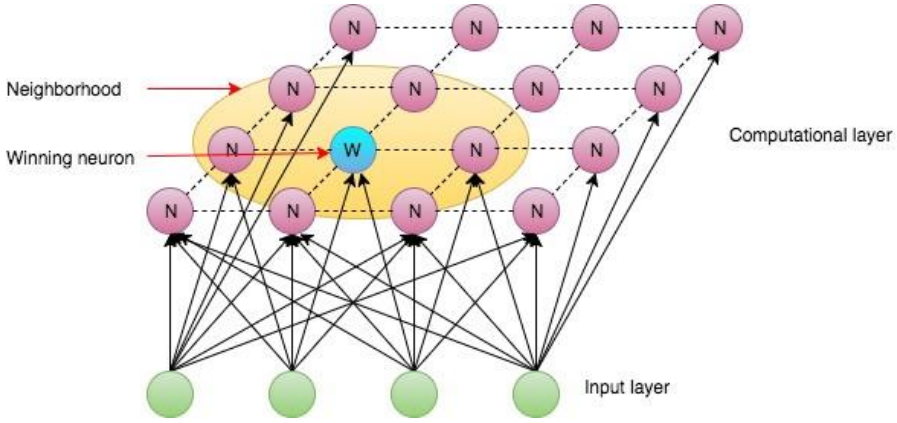


Figure 2.4: Self Organizing maps architecture

The process of generating the self-organizing maps begins by initiation the connection weights with a random value. A sample of the input vector is chosen from the training dataset. If the input vector has d dimension then it is represented as $x_i : i \in \{1,2 \ldots . d\}$ and the computational layer is constructed by N neurons mapped in *(X, Y)* dimension. Each input unit is connected to the neurons by connection weight $w_{ji} : j \in \{1,2, \ldots N\}$. Next step is to find the winning neuron, the one which has the weight vector closest to the input vector i.e. minimum value of discriminant function $f(x)$. It is computed as

$$f_j(x) = \sum_{i=1}^{d} (x_i - w_{ji})^2 \qquad (2.19)$$

where $f(x)$ calculated by taking the square Euclidean distance between the input unit and neuron. Once the winning neuron has been identified, a topological neighborhood is defined by considering the lateral distance between the neurons in the grid as formulated as Equation 2.20.

$$T_{jW(X)} = exp(-\frac{S_{jW(X)}^2}{2\sigma^2})$$

(2.20)

where $S_{jW(X)}$ is the lateral distance between the neuron *j* and winning neuron *W(X),* σ is the epoch which denotes the width of the neighborhood and programmed to exponentially decrease over time. Once the neighborhood is determined, the weight neurons in the neighborhood are updated by Δw_{ji}.

$$\Delta w_{ji} = \eta(t) T_{jW(X)}(t)(x_i - w_{ji}) \quad (2.21)$$

where *t* is the epoch dependent on learning rate $\eta(t) = \eta_0 \exp(t/\tau_\eta)$. The update causes the weight vectors of the winning neurons and the neighboring neurons to move towards the input vectors and this process is iterated for all training data instances to achieve a topographical convergence generating self-organizing feature maps which discretely represent the input data in a lower dimensional output space [35].SOM is also considered as a non-linear generalization of PCA due to its capability to cluster non-linear data instances.

2.2.2.9. K-mean

K-mean is a classic clustering technique based on iterative partitioning method. The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique of vector quantization in signal processing [31].The k-mean algorithm creates an initial partition of k ($k \leq n$) clusters from a dataset of n instances where each cluster is represented by the mean value of the instances in the cluster. In the second step, an iterative relocation technique is applied to the data observed by comparing the least squared Euclidean distance between the objects and the centroid and relocating the centroid. This process is repeated unit the squared error criterion converses. It can be defined as

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} |p - m_i|^2 \quad (2.22)$$

where E is the sum of squared error of all instances in the dataset, p is a point in the multidimensional space in the cluster c_i , m_i is the mean of the cluster c_i [1]. K-mean algorithm determines *k* partitions that minimize the square-error function and its efficiency is limited only to datasets which are clearly divergent. Additional data transformation is needful for fitting it to the real world dataset clustering.

2.2.3. Model Evaluation techniques

Designing an analytic model for the embedded system requires a thorough evaluation of the model in terms of cost and performance. As mentioned in section 2.1 , cross-validation and t-test are commonly used evaluation techniques.

2.2.3.1. Cross-validation

Cross-validation is the most commonly used method to evaluate predictive models by partitioning the original sample dataset into a training set to train the model, and a test set to evaluate it. It is mainly used to estimate the performance of the model through different metrics. The basic form of cross-validation is the k-fold cross validation where the original sample is randomly partitioned into k equal sized subsamples. k − 1 subsamples are used as training dataset and 1 is used as test dataset to perform the prediction of the model trained using the training dataset. Cross-validation process is repeated K times with of the data sub-samples used exactly once for creating the test data and an average of K then k results from the folds gives a single estimation [20]. The estimation is measured in terms of prediction accuracy as proposed in Equation 2.23 [19].

$$acc_{cv} = \frac{1}{n} \sum_{v_i y_i \in D} \delta(I(D \setminus D_i, v_i), y_i) \qquad (2.23)$$

where D is the sample dataset which is randomly split into K folds. The model is trained on $D \setminus D_t$ subsample where $t \in \{1,2 \dots k\}$. D_i is the test data sample with instance $x_i = (v_i y_i)$. The overall cross-validation is average of $(\frac{m}{m/k})$ possibilities for m/k instances out of m instances. To reduce the cost, folds are stratified before running the cross-validation, so that each class in original dataset is consistently represented in both training and test dataset. A 10-fold cross validation is the preferred method in the data science community and hence used in this study as well.

2.2.3.2. T-test

Usually, in machine learning, the toughest is to decide which mining model would be optimal for the system the analytics is designed for. Most of the embedded systems generate a huge

amount of data from different devices but the dataset used for training is usually just a sample of that big data space. Cross-validation technique evaluates individual models by predicting the true performance from an error rate of a given test dataset but does not give any assertive outcome that same result will be produced with another sample dataset. Typically, the t-test is used to do a comparative analysis of different machine learning algorithms with different samples of data. Statistically, the t-test is a method of comparing means of two different data samples.

The concept of t-test and t-distribution was developed by William Sealy Gosset under the name of Student's t-test. The t distribution is a family of curves which is derived from the degree of freedom. The degree of freedom is calculated as the number of distinct estimates on individual samples minus one. The t-distribution curve approaches the bell shape of the standard normal distribution with the increase in the degree of freedom[21][22]. A paired t-test is commonly used for t-test analysis. In paired t-test , a pair of observation for each sample is collected and a mean difference between the two sets of observation is computed. Paired t-test uses null hypothesis and a two-tailed alternative hypothesis. The null hypothesis assumes that the true mean difference between the paired samples is zero and the alternative hypothesis assumes that the true mean difference between the paired samples is not equal to zero. In a paired sample t-test, the observations are defined as the differences between two sets of values, and each assumption refers to these differences, not the original data values. The process of paired t- can be outlined in 4 steps as described below.

1. Compute sample mean
2. Compute sample Standard Deviation(SD)
3. Calculate t statistic (t) using the Equation 2.24.

$$t = \frac{\overline{d}}{\sqrt{\sigma_{d^2}/k}} \qquad (2.24)$$

   where $\overline{d}$ is the difference between means of two different sample, σ_{d^2} is the variance of two samples and k is the number of instances.

4. Calculate probability p by observing the test statistic under the null hypothesis. This value is obtained by comparing t to a t-distribution with (n − 1) degrees of freedom as formulated in Equation 2.25.

$$p = 2 \cdot Pr(T > |t|) \quad (two - tailed) \tag{2.25}$$

WEKA Experimenter provides the interface to perform paired t -test and is used for this study in Chapter 5.

2.2.4. Performance metrics

There are several performance measures that can determine the quality of a data-mining model from different aspects. Most common measured aspects are accuracy and the squared error of the predicting algorithm. Metrics used in this thesis are described in this section.

2.2.4.1. Confusion Matrix

A confusion matrix also called as error matrix[42] is type of contingency table with actual and predicted values. Terms used to describe a binary class confusion matrix are True Positive (TP), True Negative (TN), False Negative (FN), and False Positive (FP). TP is the number of correctly identified instances, TN is the correctly rejected instances and FP is the incorrectly identified instances and FN is the incorrectly rejected instance. For a multiclass dataset with $N = \{C_1, C_2 \ldots C_N\}$ classes the confusion matrix would be of N x N matrix with left axis showing the predicted class and the top axis is the actual class. For instance class C_1 has TP which is all C_1 instances that are classified as C_1, TN is calculated as all non-C_1instances that are not classified as C_1, FP is all non-C_1 instances that are classified as C_1, and FN is all C_1 instances that are not classified as C_1.

2.2.4.2. Sensitivity, Specificity And Accuracy

Sensitivity is also known as the true positive rate or recall, which measures the rate of positive instances that are correctly identified and can be calculated using the Equation 2.26.

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.26}$$

Specificity is the measure of the proportion of true negatives that are correctly identified. It can calculated using Equation 2.27.

$$Specificity = \frac{TN}{TP + FN} \quad (2.27)$$

Accuracy is a measure of all correct assessments which is calculated the percentage of correctly classified instances as shown in Equation 2.28.

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \ 100\% \quad (2.28)$$

2.2.4.3. Kappa statistic

Kappa statistic is a single value statistical metric which compares observed accuracy of the classifier with the expected accuracy. The Kappa score is a normalized value and can be calculated by using confusion matrix observations. Formulas in equation are used to calculate the Kappa statistic. Observed accuracy is sum of true positive and true negatives, divided by the total number of instances and the expected accuracy is can be defined as sum of product of reference probability and the actual probability of each class. The Kappa score can be calculated using the Equation 2.29.

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

$$Expected\ Accuracy = \left(\frac{(TN + FP) \times (TN + FN) + (FN + TP) \times (FP + TP)}{Total\ number\ of\ intances \times Total\ number\ of\ intances}\right) \quad (2.29)$$

$$Kappa\ Statistic = \frac{Observed\ Accuracy - Expected\ Accuracy}{1 - Expected\ Accuracy}$$

2.2.4.4. Precision

Precision is the positive predicated measure and is calculated ratio of true positives to the number of all relevant observations including true positives and false positives(Equation 2.30).

$$Precision = \frac{TP}{TP + FP} \tag{2.30}$$

2.2.4.5. F-Measure

F- measure is a weighted harmonic mean of the precision and sensitivity (also known as recall). It is metric used to measure the accuracy of the test. F-score can be calculated using Equation 2.31.

$$F = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} \tag{2.31}$$

2.2.4.6. ROC area

Receiver Operating Characteristic(ROC) curve is formed by plotting true positive rate in a function of false positive rate for different threshold points. True positive rate is the sensitivity represented in y-axis and the false positive rate is calculated as 1-specificity and plotted in x-axis. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular threshold as shown in Figure . In a test when ROC curve passes through the upper left corner then it indicates 100% sensitivity, 100% specificity. Hence a classifier with a ROC curve closer to upper left corner has a higher overall accuracy [43]. WEKA cross validation test measures the classifier's accuracy by the area under the ROC curve, also called AUC.

AUC is measured by using Equation 2.32.

$$AUC = \int_0^1 ROC(t)dt \tag{2.32}$$

where $ROC(t)$ is the sensitivity and t = 1-specificity. A value of 1 is considered optimal and a value of 0.5 or less is considered worthless. A sample ROC curve can be seen in Figure 2.5

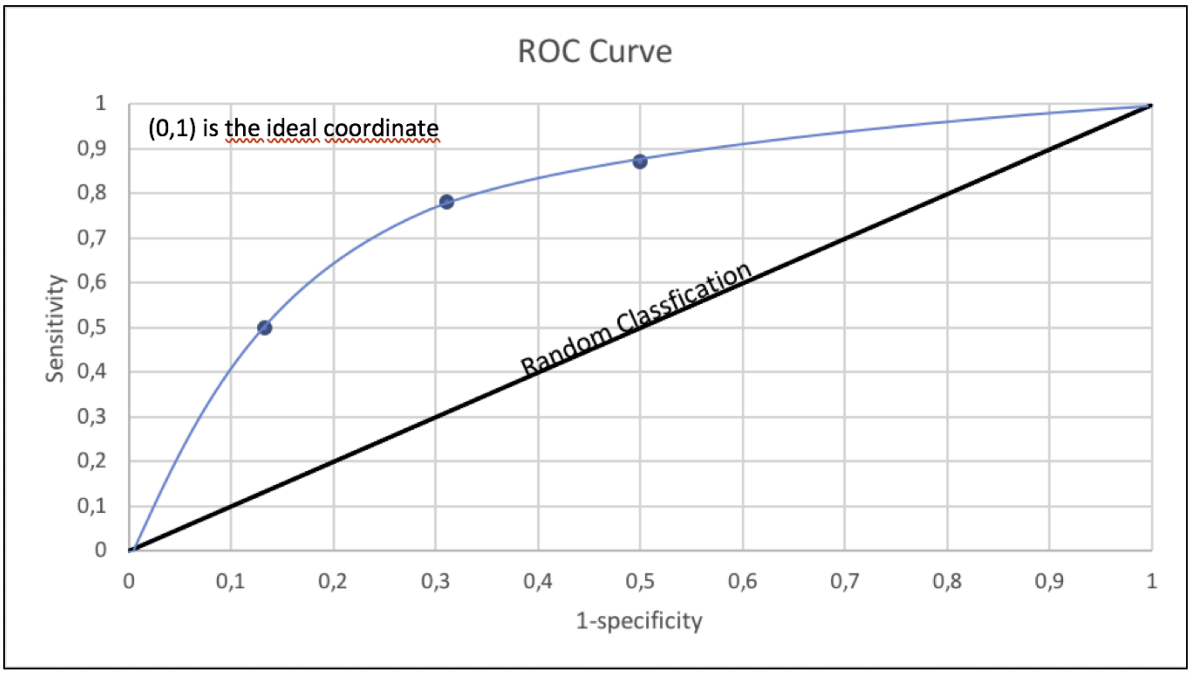


Figure 2.5:ROC Curve

2.2.4.7. Correlation coefficient

In case of predicting continuous values , correlation coefficient measures how well the predictions are correlated or change with the actual output value. It gives values between -1 and 1.A value of 0 means there is no relation and a value of 1 is a perfectly correlated set of predictions. A negative value indicates an inverse linear relation.

2.2.4.8. Mean Absolute Error(MAE)

Mean absolute error is the measure of difference between the continuous prediction and the actual data point in the data instances. The MAE is calculated as the average of absolute error per instance for all the instances in the dataset. Absolute error is the difference between the measured value and actual value(Equation 2.33).

$$MAE = \frac{\sum_{i=1}^{N} |Predicted_i - Actual_i|}{N} \quad (2.33)$$

2.2.4.9. Root mean squared error(RMSE)

Root Mean Square Error (RMSE) is the standard deviation of the prediction errors which are a measure of how far from the regression line data points are. In other words, RMSE indicate the distribution of data around the line of best fit. RMSE metric can be calculated by Equation 2.34.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}} \quad (2.34)$$

2.3 Related studies

Considering the fact that data mining domain is flooded with algorithms and procedures, the quest for finding the perfect method has been perpetual. Several research works and case studies have led to a selection of many state-of-art algorithms which are common in use across multiple domains.

Karen Zita Haigh et al. presented a case study on the use of machine learning techniques in embedded electronics and argued the challenges faced with the usage of traditional approaches [24]. They also proposed an optimization of Super vector machine algorithm which was implemented on general purpose processors of two communications networks. Qing Chen Zhang et al. argues about the data mining challenges in IoT systems and propose two enhancement to high-order c-means algorithms for clustering big dataset and exhibit the performance improvement in terms of high compression rate without compromising with the accuracy[25]. Mark A. Hall and Geoffrey Holmes [10] benchmarked some of the attribute selection methods for supervised machine learning algorithms C4.5 and naive Bayes. They concluded that attribute selection is an effective pre-processing method to improve the mining model, along with that they also opinionated that a single method cannot be claimed as the best approach but the outcome is dependent on the model and dataset characteristics. Rafet Duriqi et al. analyses classification algorithms on three different datasets using WEKA. They picked Naive Bayes, Random Forest, and K * algorithm to perform the study and concluded that the feature count and data characteristics are influential in the performance of classifier [23].

Chapter 3

3.1 Datasets selected for the analysis

Datasets have been selected based on the type of data mining tasks required to solve the objective of data analytics. Different type of datasets are selected to perform a comprehensive comparison of the algorithms. Due to growing interest in data analysis research , many public domain dataset repositories have been created with active contribution from various research disciplines. For this study as well datasets have been gathered from different public domain libraries [36][38].

Dataset 1: Human Activity Recognition(HAR)

The first dataset is multiclass Human Activity Recognition(HAR) dataset. With the explosive growth of smartphone technologies , the use of embedded sensors to monitor human activity is not a far-fetched notion and the data generated has it relevance in healthcare , security surveillances and human-machine interaction but due to its complex nature , building an analytics model is challenging and therefore is one of the most sought after research subject in data mining community. The dataset used in this study was created by Davide Anguita et. al using waist mounted android smartphone[37]. Data was acquired from smartphone embedded accelerometers and gyroscopes, targeting the recognition of six different human activities: standing, sitting, laying, down, walking, walking downstairs and upstairs. Experiments were conducted with a group of 30 volunteers within the age boundary of 19-48 years. The embedded sensors captured the 3-axial linear and acceleration and 3-axial angular velocity at a constant rate of 50Hz and the data was cleaned by noise filter application. The resulting dataset was a 561 feature vector with time and frequency domain variables. Table 3.1 gives an overview of the dataset characteristics. The class distribution can be seen in Figure 3.1.

Characteristics	Description
Dataset Name	HAR
Dataset Type	Multivariate ,Time-series
Number of Attributes	561
Attribute Type	Numeric

Number of Instances	10299
Data mining tasks	Classification , Clustering
Class labels	1. Walking 2. Walking upstairs 3. Walking downstairs 4. Sitting 5. Standing 6. Laying, down

Table 3.1: HAR dataset characteristics

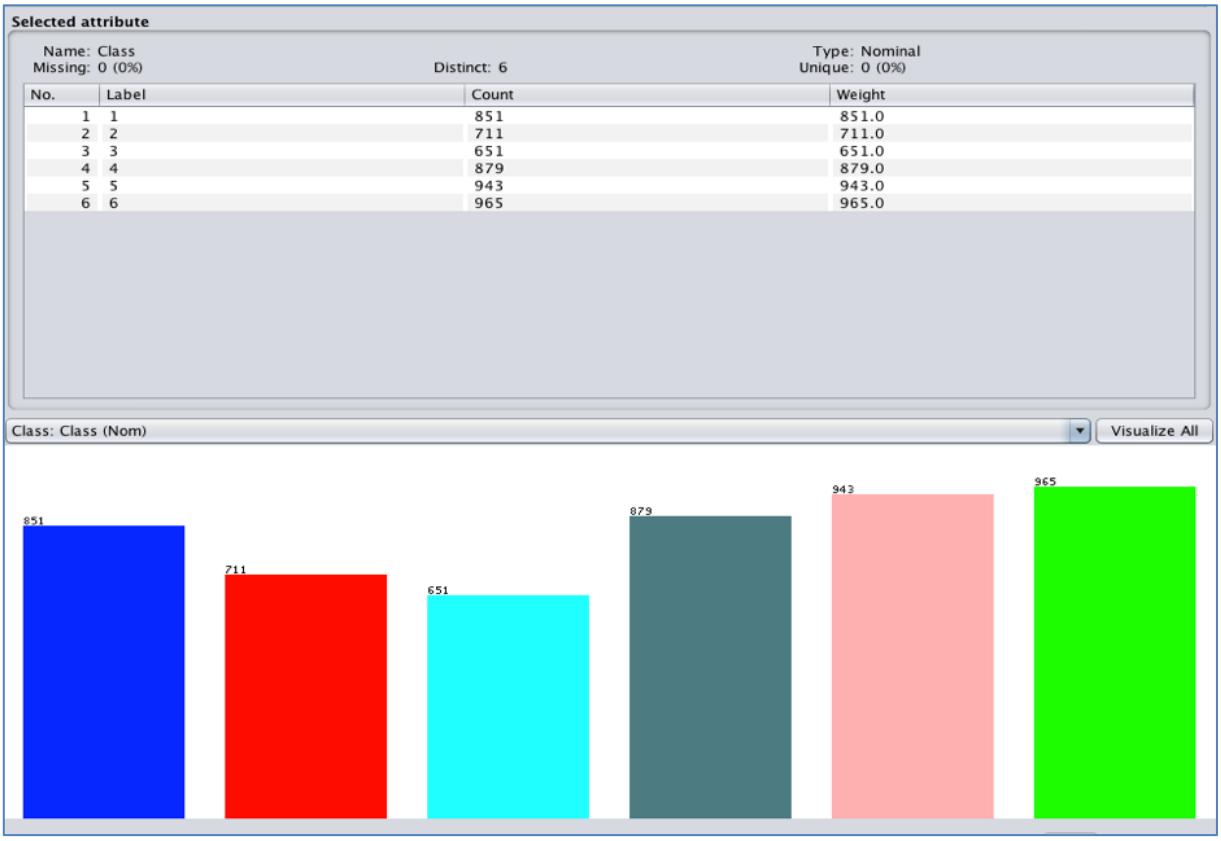


Figure 3.1: Class distribution of HAR Dataset

Dataset 2: Vehicle sensing

The second dataset is 2- class vehicle sensing dataset generated from multiple micro sensing devices integrated in to a Wireless Distributed Network (WSDN). The dataset was created by Marco F. Duarte et. al as part of SensIT situational experimentation organized by DARPA/IXOs SensIT (Sensor Information Technology) program at Marine Corps Air GroundCom bat Center in Twenty-nine Palms, CA, USA with an objective of detecting location of vehicle and its type[39]. Seventy-five WINS NG 2.0 nodes were deployed in the

region with each sensor node equipped with acoustic seismic infrared sensors. The dataset consist of time series data represented as feature vectors and with 2 class attribute (1 ,-1) . The detailed data characteristics is specified in Table 3.2. The class distribution can be seen in Figure 3.2.

Characteristics	Description
Dataset Name	vehicle
Dataset Type	Multivariate ,Time-series
Number of Attributes	101
Attribute Type	Numeric
Number of Instances	14561
Data mining tasks	Classification
Class labels	1, -1

Table 3.2: Vehicle sensing dataset characteristics

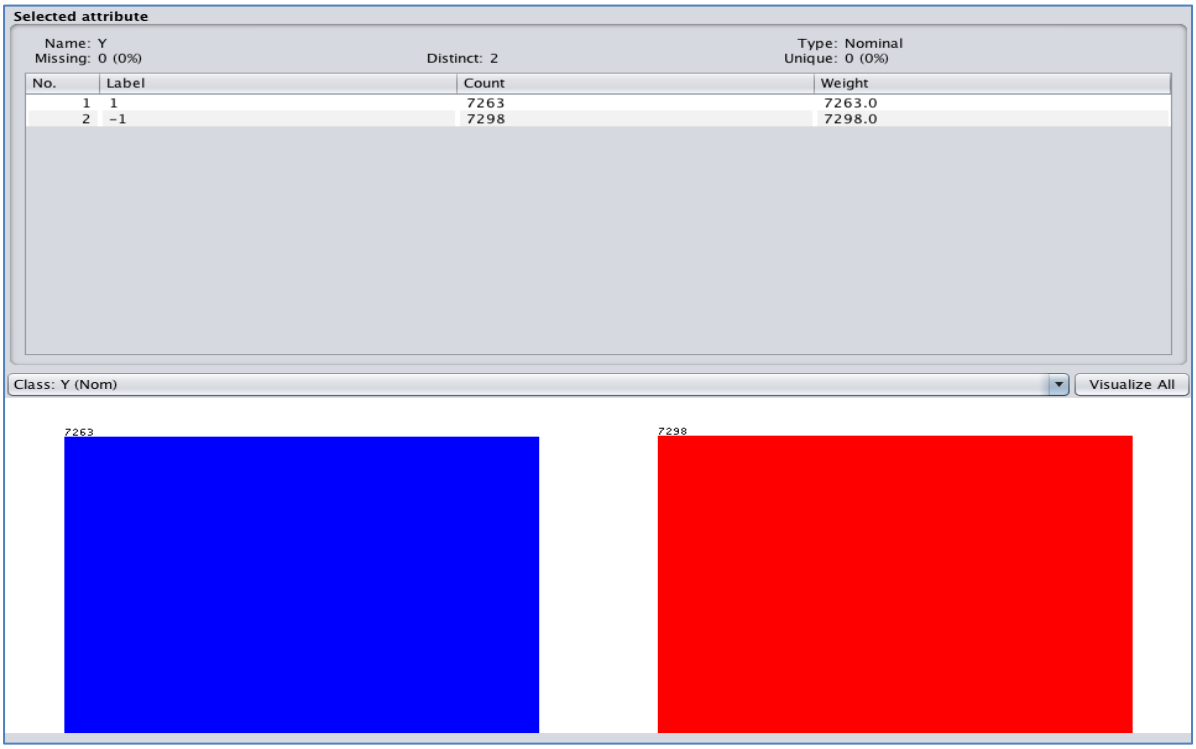


Figure 3.2: Class distribution of Vehicle Dataset

Dataset 3: Appliances energy prediction

Appliances energy prediction dataset was chosen to perform regression analysis. This dataset was contributed by Luis M et al. during their study for designing data driven predictive

models for energy usage of home appliances[40]. The main purpose of their study was to identify the relationships between the different predictive attributes such as weather, room temperature, humidity, date and time etc. and the household energy consumption. ZigBee wireless network was used to monitor the house temperature and humidity and . Energy metering was done using M-BUS energy counters. Energy counters measured the consumption every 10 min. Then the information was stored and transmitted every 12h over an internet-connected energy monitoring system. The data was logged every 10 min for the appliances. Weather data was collected from nearest airport weather station and merged together with the experimental data sets using the date and time column. The dataset was also injected with two random attributes for regression models testing. Table 3.3 provides the characteristics of the dataset and class distribution plot can be seen in Figure 3.3.

Characteristics	**Description**
Dataset Name	energydata_complete
Dataset Type	Multivariate
Number of Attributes	29
Attribute Type	Real
Number of Instances	19518
Data mining tasks	Regression

Table 3.3: Appliances energy prediction dataset characteristics

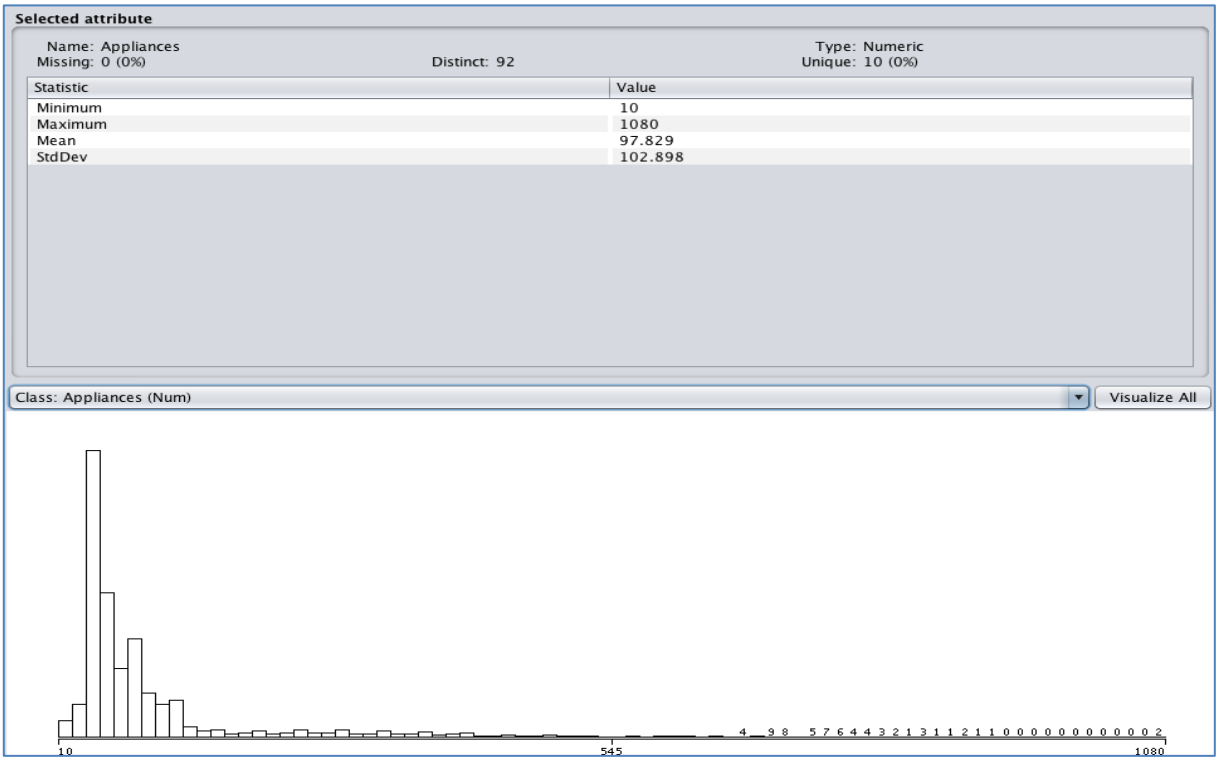


Figure 3.3: Class distribution of Appliances energy prediction dataset

Dataset 4: Puma 560 robot arm

The next dataset is part of the family of dataset which were synthetically generated from a realistic simulation of the dynamics of a Unimation Puma 560 robot arm and has been acquired from DELVE repository[46]. Data mining tasks involve regression analysis of angular acceleration of one of the robot arm's links. The dataset is defined by attributes like angular positions, velocities and torques of the robot arm. The attributes are nonlinear and the output also contains moderate amount of noise. Data characteristics are tabularized in Table 3.3 and class distribution plot is depicted in Figure 3.4.

Characteristics	Description
Dataset Name	**puma8NH**
Dataset Type	Multivariate
Number of Attributes	9
Attribute Type	Real
Number of Instances	8192
Data mining tasks	Regression

Table 3.4: Puma 560 robot arm dataset characteristics

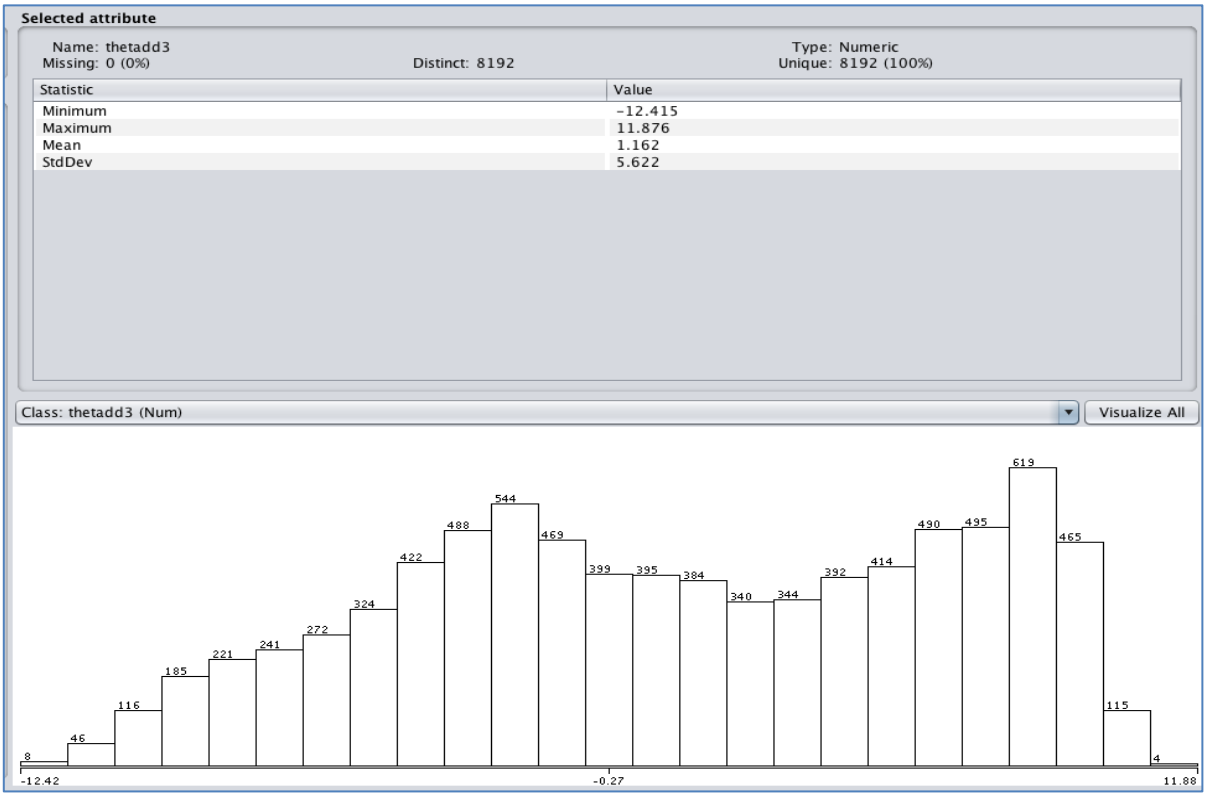


Figure 3.4: Class distribution of Puma 560 robot arm dataset

3.2 Waikato Environment for Knowledge Analysis(WEKA)

The framework used for data analysis is Waikato Environment for Knowledge Analysis(WEKA) , version 3.8.2. WEKA is an open source , easy to use machine learning and predictive modeling tool licensed under the GNU General Public License, that was developed by the University of Waikato in New Zealand and is written in JAVA [2]. Later in 2006 WEKA was bought by Pentaho and integrated as part of its Business Intelligence(BI) suite[45]. WEKA contains a collection of visualization tools and algorithms for all category of machine learning tasks such as classification, regression, clustering , evaluation and data transformation. It also provides a set of graphical user interfaces for easy data analytics model generation. The new WEKA suite include a Workbench which is a combined GUI for all the interfaces(Explorer, Experimenter, simple CLI). The Experimenter GUI is useful for performance evaluation of different algorithms for a given dataset which gives an in-depth statistics for a better decision making process. Analysis and evaluation of the models are done on WEKA Workbench and WEKA Knowledge flow. WEKA Experimenter is used to perform comparative analysis.

WEKA stores the dataset in Attribute-Relation File Format (ARFF) file. An ARFF file has header section which includes the relation name annotated by @RELATION and the attribute

names and type description annotated by @ATTRIBUTE . Weka supports types datatypes for attributes- numeric, nominal, string and date. The next section consist of the data annotated by @DATA. Each instance is represented in a single line and filled with "?" for missing values. A sample arff file is shown in Figure 3.5.

```
@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}
@attribute temperature {hot, mild, cool}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
```

Figure 3.5: Sample Arff file

3.3 Test Environment setup

3.3.1 Hardware/software specifications

The hardware/software specifications of the test environment are specified in the Table 3.5

Processor	Intel Core i5
Processor Speed	2,7 GHz
Number of Processors	1
Total Number of Cores	2
Memory	8 GB
OS	macOS High Sierra

Table 3.5 Hardware/software configuration

3.3.2 WEKA test bench

The testbench constructed in WEKA's knowledge flow can be seen in Figure 3.2 and 3.3. Figure 3.2 is the test model for classification analysis and Figure 3.3 is for regression algorithm validation. The test model is run for each dataset by loading the arff file in the ArffLoader, the class attribute is assigned in the classAssigner process and in case of

classification problem , the ClassValuePicker is used to select a specific class value to evaluate the classification and generate the ROC curve. CrossValidationFoldMaker is used to test the model with a 10-fold cross validation.

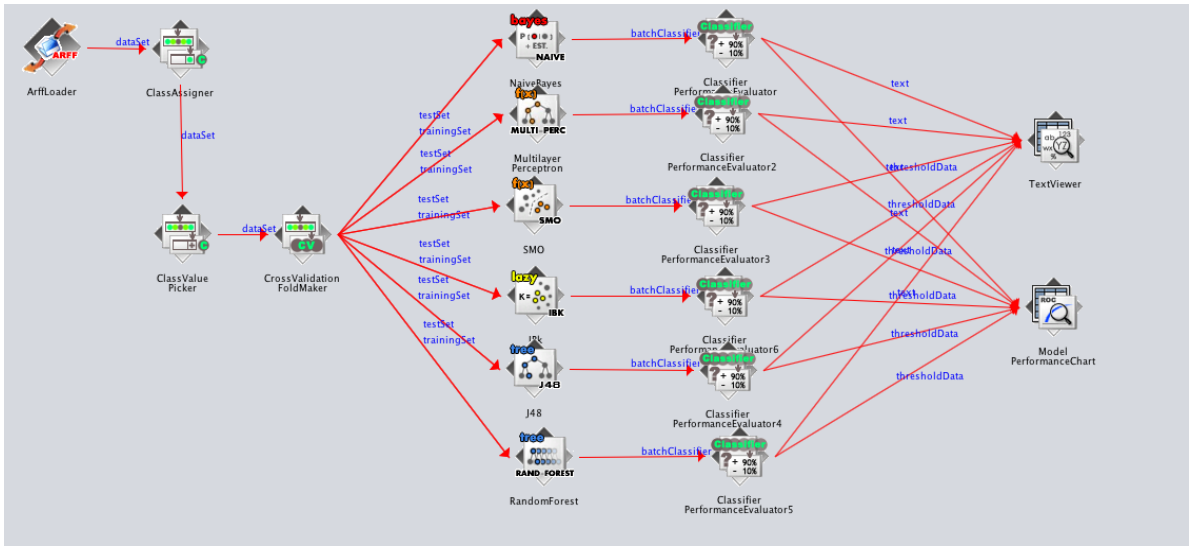


Figure 3.6 WEKA Classification testbench

The ModelPerformanceChart and TextViewer is used to visualize the evaluation results. TextViewer gives the details report of cross validation analysis and ModelPerformanceChart generate the ROC curve for class label selected in the ClassValuePicker process.

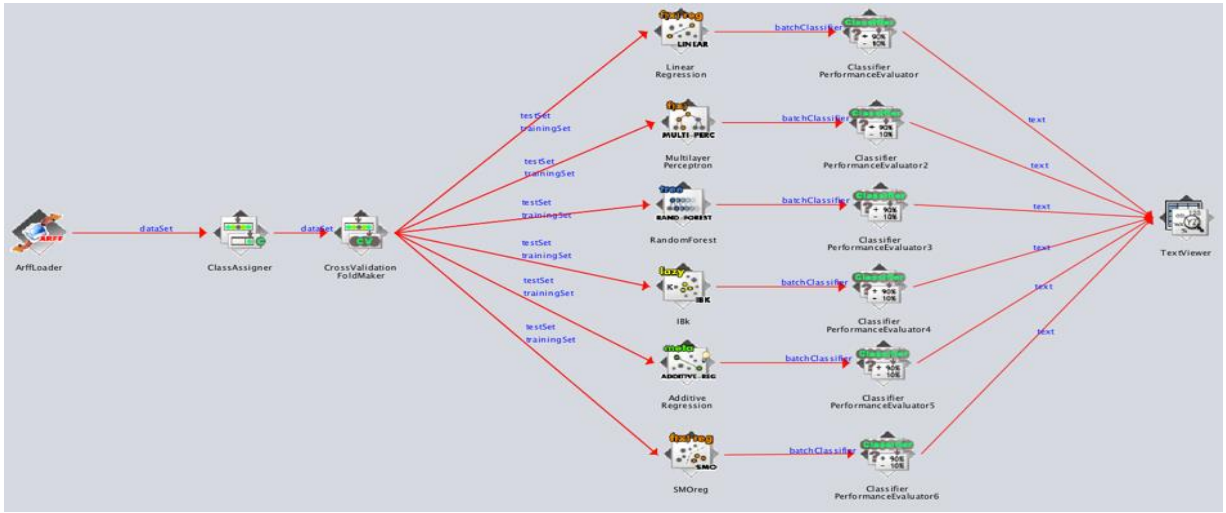


Figure 3.7 WEKA regression testbench

Chapter 4

In this chapter machine learning algorithms are validated and results are analyzed comprehensively for each dataset selected for this study. A dataset specific comparative analysis of machine learning algorithms are projected in 2D column charts for different experimental setup.

4.1 Performance analysis method

For each dataset, analysis is done in two different setups. The first set of validation was performed using the raw dataset with no data transformation. The process of experimentation for raw dataset is explained in section 4.1.1. The second set of validation is performed using dataset that has been transformed using Random Projection, CFS with greedy stepwise and PCA with Ranker method , the process of which is described in section 4.1.2.

4.1.1 Evaluation method for raw dataset

For raw dataset , a 10 fold cross validation is selected as “Test options” during the execution of the classifier in WEKA workbench. The complete dataset is used for the evaluation process during which the dataset is subsampled into 10 subsets of which a single sub sample is retained as the validation data for testing and the remaining 9 subsamples are used for training the classifier. The evaluation is repeated 10 times with each subsample used exactly once for validation. The overall performance is calculated by averaging the individual results generated by the 10 folds, to produce an overall estimation of the efficiency of the classifiers ,as illustrated in section 4.2 and section 4.3.

4.1.2 Evaluation method for transformed dataset

A slightly different approach is applied when performing evaluation with transformed dataset. Three different preprocessing methods are applied to the raw dataset to generate three transformed datasets with reduced set of attributes. Data transformation is performed using WEKA workbench “Select Attribute” feature. Inputs to the select attribute process are, attribute evaluator, search method, and attribute selection mode. The attribute evaluator is used for evaluating each attribute in the original dataset in context of the class attribute . The search method algorithm navigate through the dataset to form different combination of attributes which performs best with attribute evaluator. With Greedy stepwise search method

,CFS is used as an attribute evaluator and with Ranker search method, PCA is used as an attribute evaluator. 10 fold cross validation with seed 1 is chosen as the attribute selection mode. Cross validation is used to indicate the stability of the attribute selection by giving a statistic analysis of how many folds a given attribute appeared in the best subset found by the search method. The seed is a component of the randomness. Random projection is performed in a different method using WEKA workbench preprocessing feature which uses unsupervised attribute filtering technique for transforming the attributes.

For the performance analysis of the classifiers, each transformed dataset is used as the training dataset and original dataset is used as the test dataset by choosing the Test options in WEKA. The performance of different classifiers are illustrated in section 4.2 and section 4.3.

4.2. Performance analysis results of classification algorithms

For classification problems class label specific accuracy is depicted in a 2D AUC score chart which maps each class AUC scores calculated by 10 fold cross validation of machine learning algorithms. All the metrics results for classification analysis are depicted in percentage to conveniently project in comparative analysis charts. Regression schemes were evaluated with correlation coefficient, RMSE and MEA.

4.2.1. Experiment 1- Human Activity Recognition(HAR)

Classification specific evaluation metrics results were compared by running the WEKA test bench shown in Figure 3.2 for HAR dataset with different test setup. In the first test run with the original data, it was discovered that due to high dimensional nature of the dataset (562 attributes), the most expensive and inefficient algorithm was multilayer perceptron in terms of time taken as it look longer that 90 minutes to produce the results , hence excluded in the reports. Satisfactory results were seen C4.5, SVM, kNN and Random Forest with a accuracy and sensitivity(Recall) of more than 90% and low FP rate. Naïve Bayes had the least accuracy and a high FP rate. Detailed results can be seen in Figure 4.1.

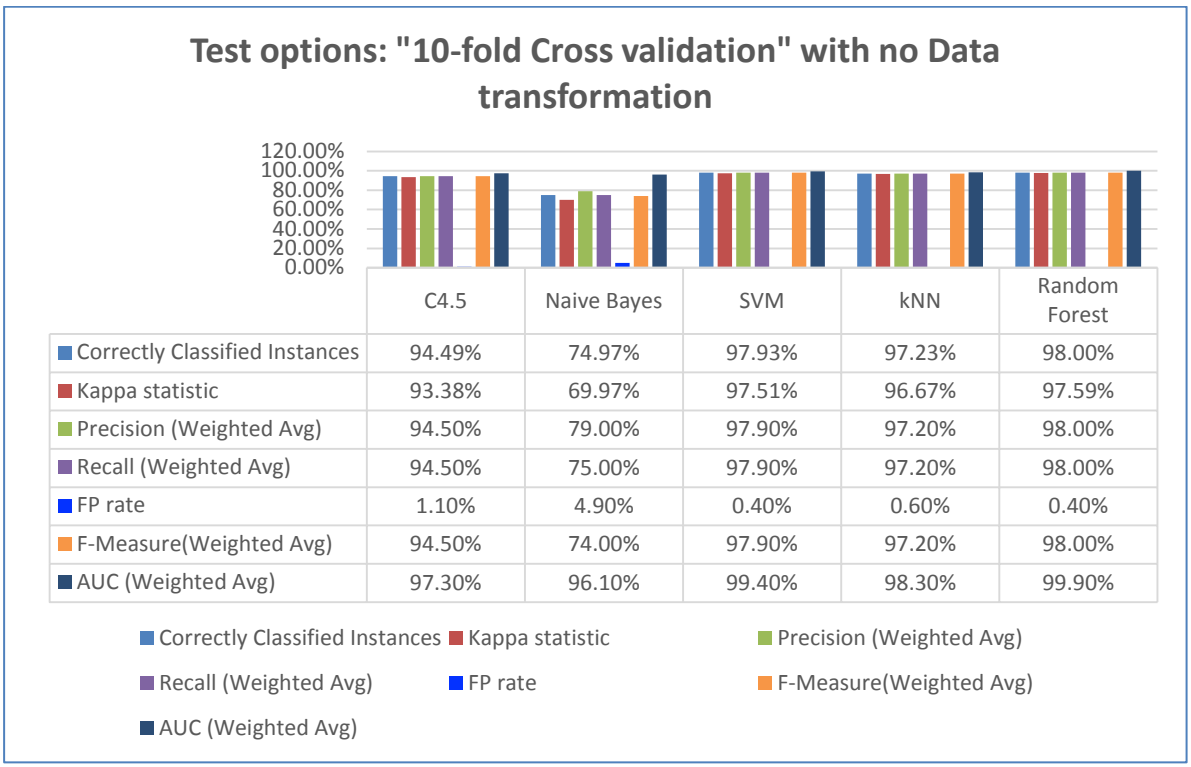


Figure 4.1: Performance analysis report on HAR dataset with no data transformation

The AUC Scores for each class were mapped in a 2D line chart for different models in figure 4.2. It is observed that Random forest AUC scores are the most consistent and optimal ones.

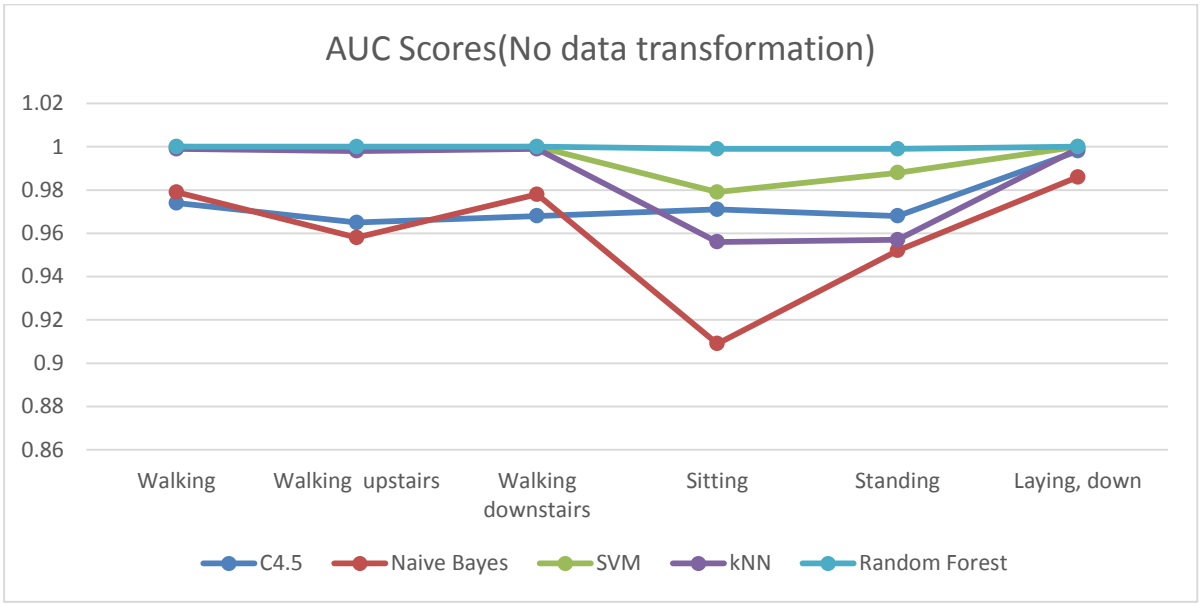
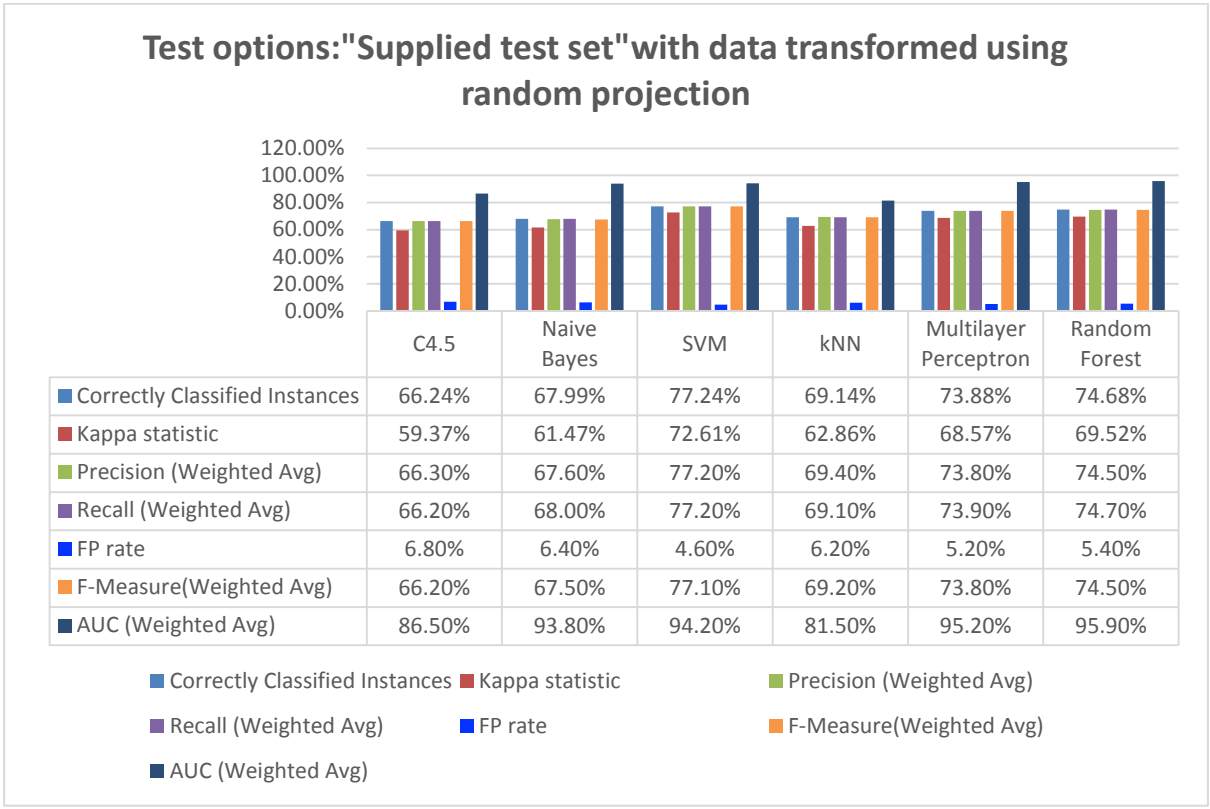


Figure 4.2 AUC score 2D -line chart for HAR dataset with no data transformation

With the application of random projection which reduced the dataset to 11 attributes to the original dataset , a significant degradation in the performance can be seen in all the evaluation metrics for all the models but results were obtained for multilayer perceptron with an accuracy of 73.88% (Figure 4.3). The AUC scores can be seen in Figure 4.4



	C4.5	Naive Bayes	SVM	kNN	Multilayer Perceptron	Random Forest
Correctly Classified Instances	66.24%	67.99%	77.24%	69.14%	73.88%	74.68%
Kappa statistic	59.37%	61.47%	72.61%	62.86%	68.57%	69.52%
Precision (Weighted Avg)	66.30%	67.60%	77.20%	69.40%	73.80%	74.50%
Recall (Weighted Avg)	66.20%	68.00%	77.20%	69.10%	73.90%	74.70%
FP rate	6.80%	6.40%	4.60%	6.20%	5.20%	5.40%
F-Measure(Weighted Avg)	66.20%	67.50%	77.10%	69.20%	73.80%	74.50%
AUC (Weighted Avg)	86.50%	93.80%	94.20%	81.50%	95.20%	95.90%

Figure 4.3:1 Performance analysis report on HAR dataset transformed with Random Projection

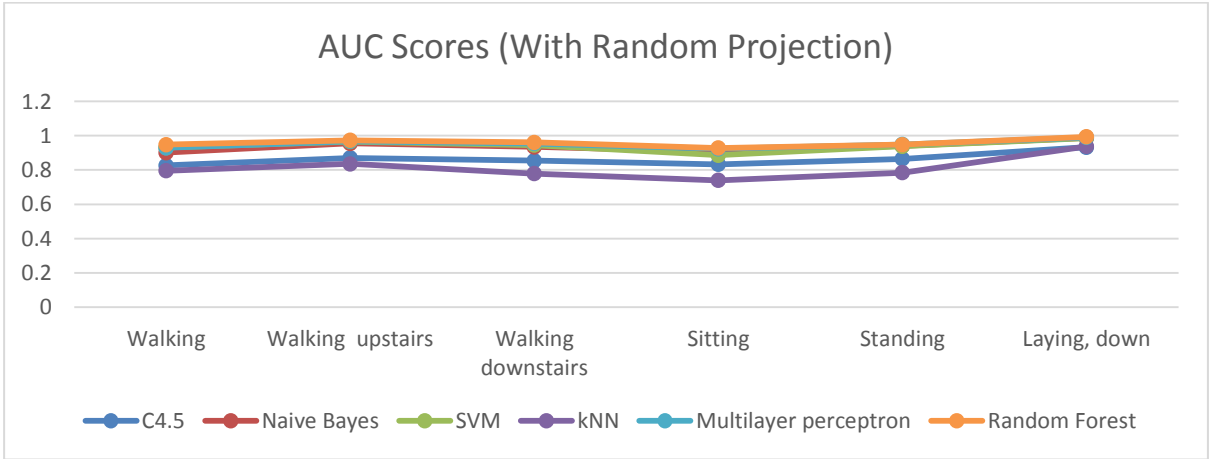
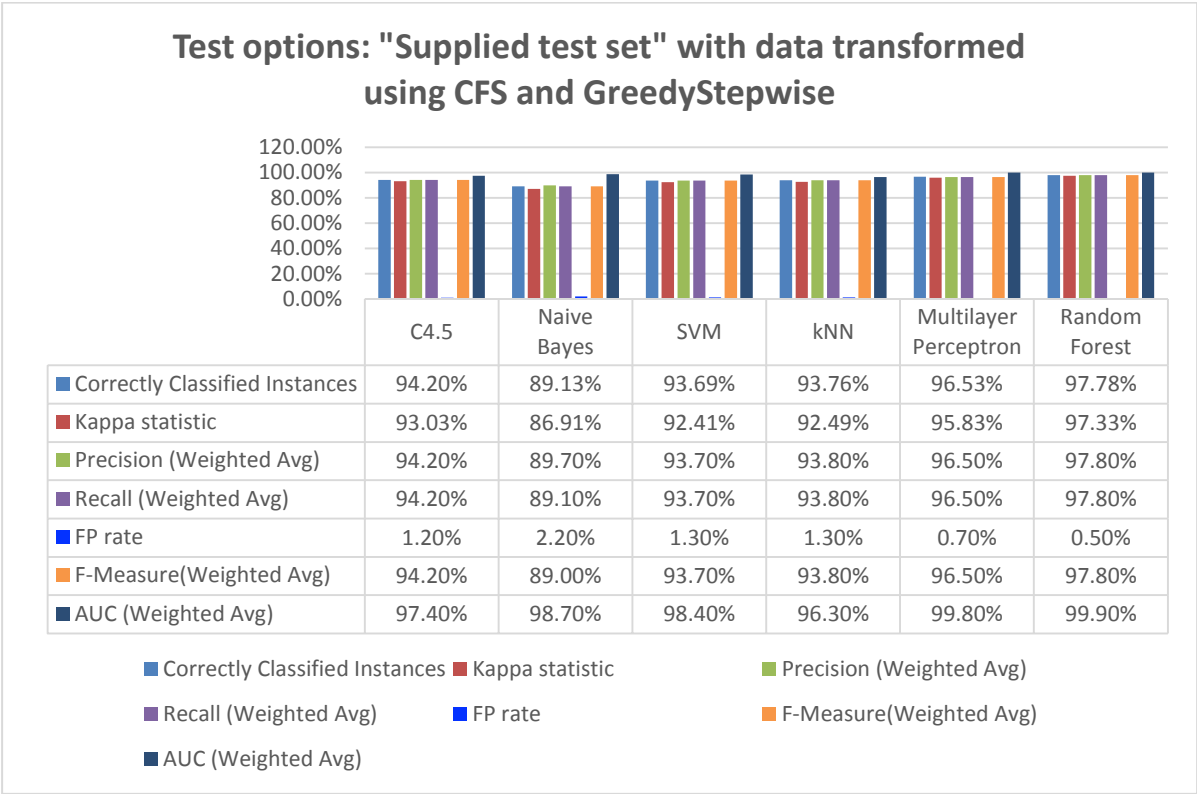


Figure 4.4 AUC score 2D -line chart for HAR dataset transformed with Random Projection

In the next experimentation the original data set was transformed using CFS and GreedyStepwise attribute selection method. The data dimension was reduced to 58 attributes. Satisfactory performance results were seen when transformed data was used as training set and original dataset is supplied as test set. Contrary to the evaluation results seen in original data models , all data models generated using CFS and GreedyStepwise transformed data displayed similar results compared to models generated with original dataset as seen in Figure 4.1. Naive Bayes and Multilayer Perceptron's performance had a significant improvement with an accuracy of more than 85%. Detail evaluation results can been seen in Figure 4.5.



	C4.5	Naive Bayes	SVM	kNN	Multilayer Perceptron	Random Forest
Correctly Classified Instances	94.20%	89.13%	93.69%	93.76%	96.53%	97.78%
Kappa statistic	93.03%	86.91%	92.41%	92.49%	95.83%	97.33%
Precision (Weighted Avg)	94.20%	89.70%	93.70%	93.80%	96.50%	97.80%
Recall (Weighted Avg)	94.20%	89.10%	93.70%	93.80%	96.50%	97.80%
FP rate	1.20%	2.20%	1.30%	1.30%	0.70%	0.50%
F-Measure(Weighted Avg)	94.20%	89.00%	93.70%	93.80%	96.50%	97.80%
AUC (Weighted Avg)	97.40%	98.70%	98.40%	96.30%	99.80%	99.90%

Figure 4.5: Performance analysis report on HAR dataset transformed with CFS and GreedyStepwise

The AUC score chart in Figure 4.6 projects the AUC scores for classes when models were built on CFS and greedy stepwise transformed data and it can be seen that class Laying down and walking have a high AUC score across all machine learning algorithm which a better likelihood of correct prediction.

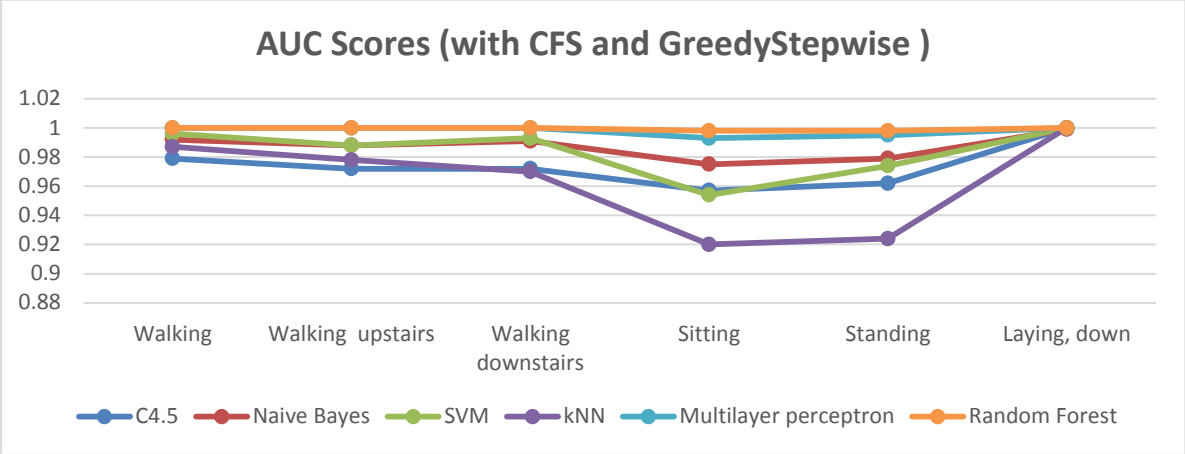
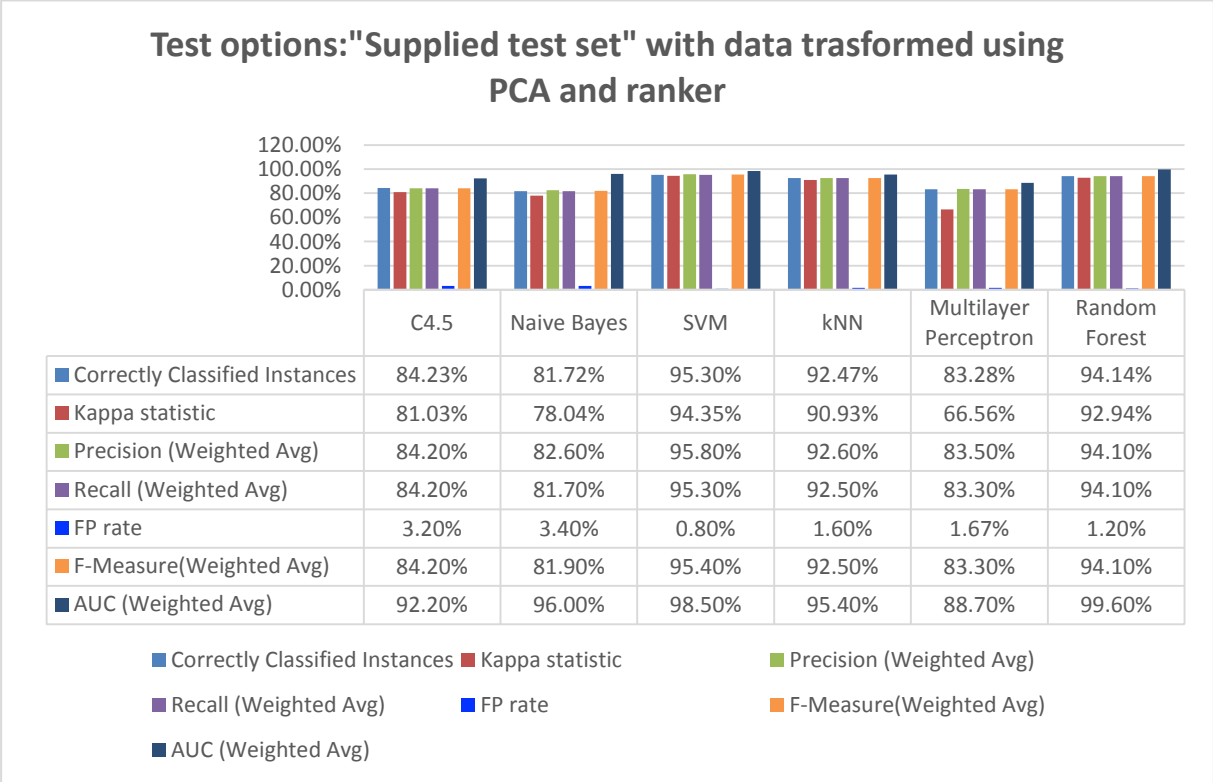


Figure 4.6: AUC score 2D -line chart for HAR dataset transformed with CFS and GreedyStepwise

In next test setup the data models were generated with a PCA and ranker method preprocessed dataset. The dataset was transformed to 105 attribute . Evaluation results did not show any overall improvement in performance compared to test setup with original dataset and with CFS and GreedyStepwise transformed data. Model build for Multilayer perceptron was faster and validation results were satisfactory(Accuracy of 83.28%) compared to original dataset. Details of the analysis can be seen in Figure 4.7.



	C4.5	Naive Bayes	SVM	kNN	Multilayer Perceptron	Random Forest
Correctly Classified Instances	84.23%	81.72%	95.30%	92.47%	83.28%	94.14%
Kappa statistic	81.03%	78.04%	94.35%	90.93%	66.56%	92.94%
Precision (Weighted Avg)	84.20%	82.60%	95.80%	92.60%	83.50%	94.10%
Recall (Weighted Avg)	84.20%	81.70%	95.30%	92.50%	83.30%	94.10%
FP rate	3.20%	3.40%	0.80%	1.60%	1.67%	1.20%
F-Measure(Weighted Avg)	84.20%	81.90%	95.40%	92.50%	83.30%	94.10%
AUC (Weighted Avg)	92.20%	96.00%	98.50%	95.40%	88.70%	99.60%

Figure 4.7: Performance analysis report on HAR dataset transformed with PCA and Ranker

The AUC scores also does not indicate any improvement and projects an inconsistent pattern in prediction of classes as can be seen in Figure 4.8.

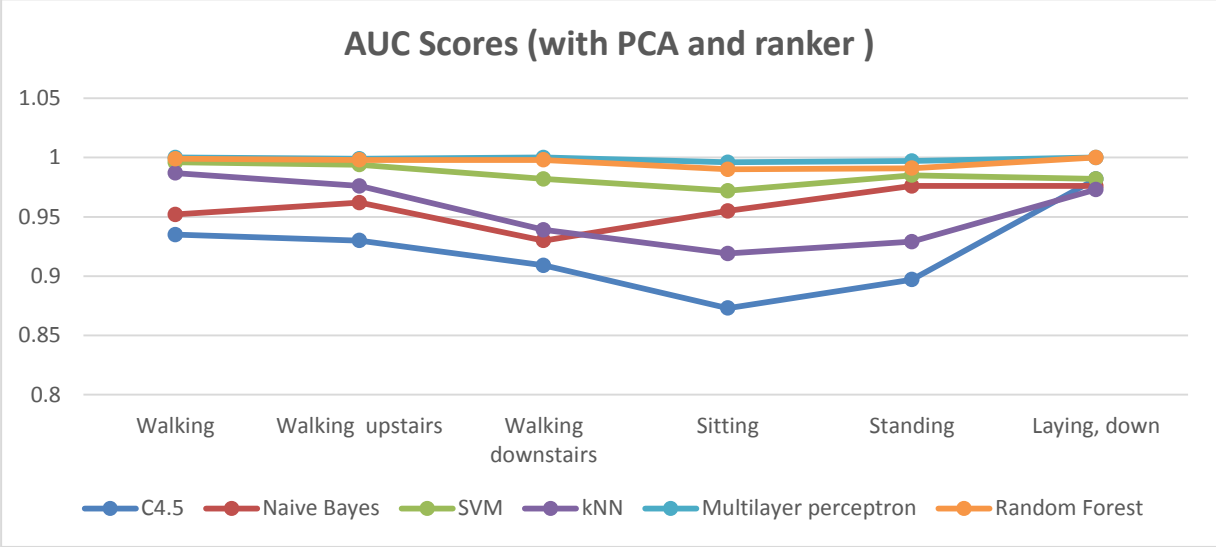


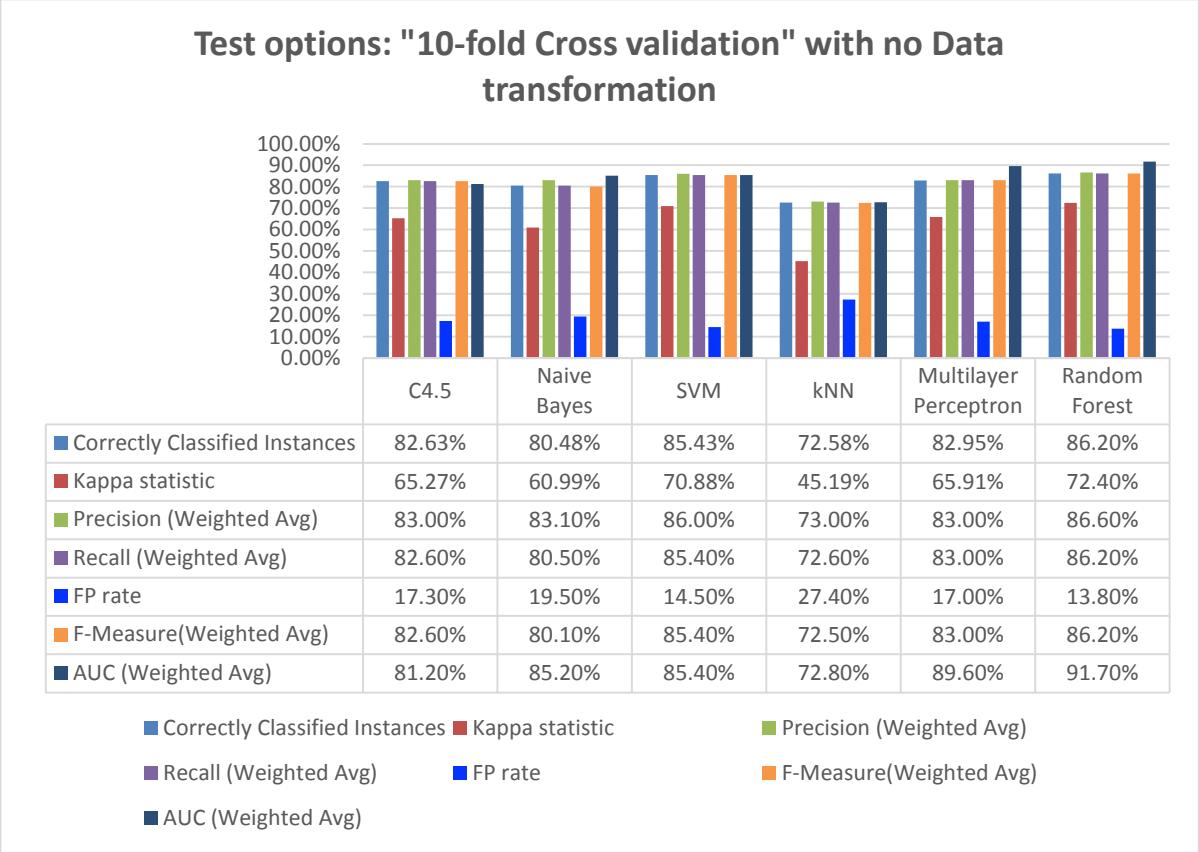
Figure 4.8: AUC score 2D -line chart for HAR dataset transformed with PCA and Ranker

From the above experimentation it can be observed that Random Forest and SVM models stand out in terms of high accuracy with original dataset but SVM shows inconsistency with the AUC scores as it has a low score for class labels sitting and standing , which means prediction of these class labels were not efficient . On the other hand Random Forest had a consistent line of AUC scores as projected in Figure 4.2. Considering the fact that storage of the data is one of the biggest challenges of data analytics in recent times, model constructed using a preprocessed data (with CFS and GreedyStepwise) with Random Forest and SVM algorithms seem to be good candidates for HAR analytics design. Further analysis in section 5.1 will consolidate this initial findings.

4.2.2. Experiment 2-Vehicle sensing

The next experiment was conducted on dataset 2 which is a 2 class vehicle sensing dataset sing test bench shown in figure 3.3. The test setup is similar to the first experiment discussed in section 4.1. The experiment was conduction with four test setup , first one with the original dataset and the next ones with transformed data using different preprocessing techniques. The models are evaluated with 10-fold cross validation. Figure 4.9 gives the details statistics of the analysis done on the original dataset with no data transformation. Random Forest has the best accuracy of 86.20% and weighted average AUC of 91.70% relative to other algorithms.

Random forest also has a low FP of 13,80% which give the rate of incorrectly sensing a vehicle. SVM also got good results with a accuracy of 85.43% and FP rate of 14.50 %.



	C4.5	Naive Bayes	SVM	kNN	Multilayer Perceptron	Random Forest
Correctly Classified Instances	82.63%	80.48%	85.43%	72.58%	82.95%	86.20%
Kappa statistic	65.27%	60.99%	70.88%	45.19%	65.91%	72.40%
Precision (Weighted Avg)	83.00%	83.10%	86.00%	73.00%	83.00%	86.60%
Recall (Weighted Avg)	82.60%	80.50%	85.40%	72.60%	83.00%	86.20%
FP rate	17.30%	19.50%	14.50%	27.40%	17.00%	13.80%
F-Measure(Weighted Avg)	82.60%	80.10%	85.40%	72.50%	83.00%	86.20%
AUC (Weighted Avg)	81.20%	85.20%	85.40%	72.80%	89.60%	91.70%

Figure 4.9: Performance analysis report on Vehicle dataset with no data transformation

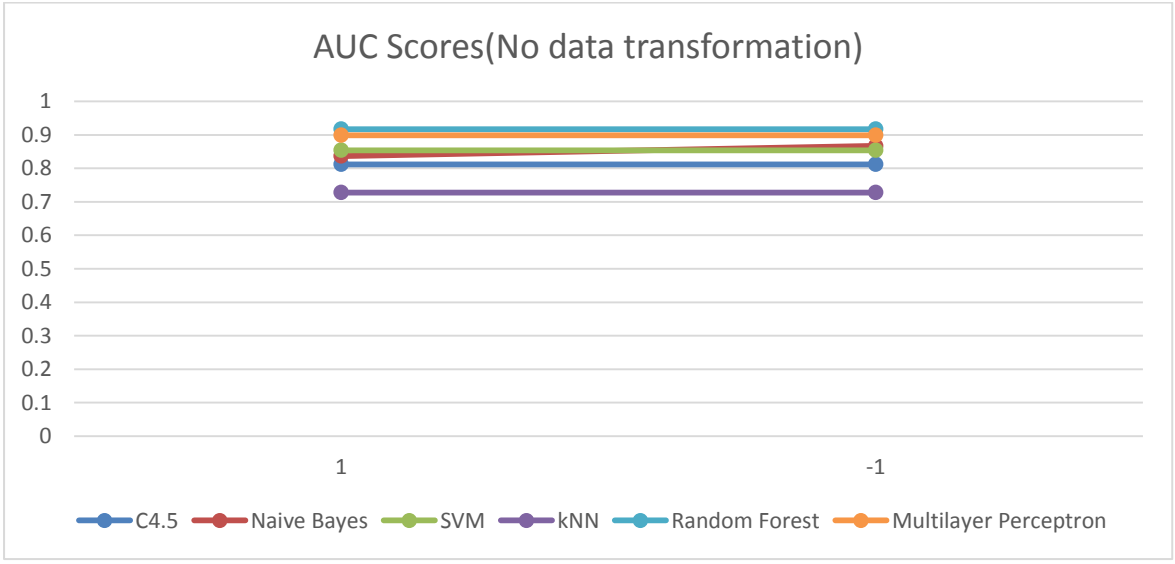
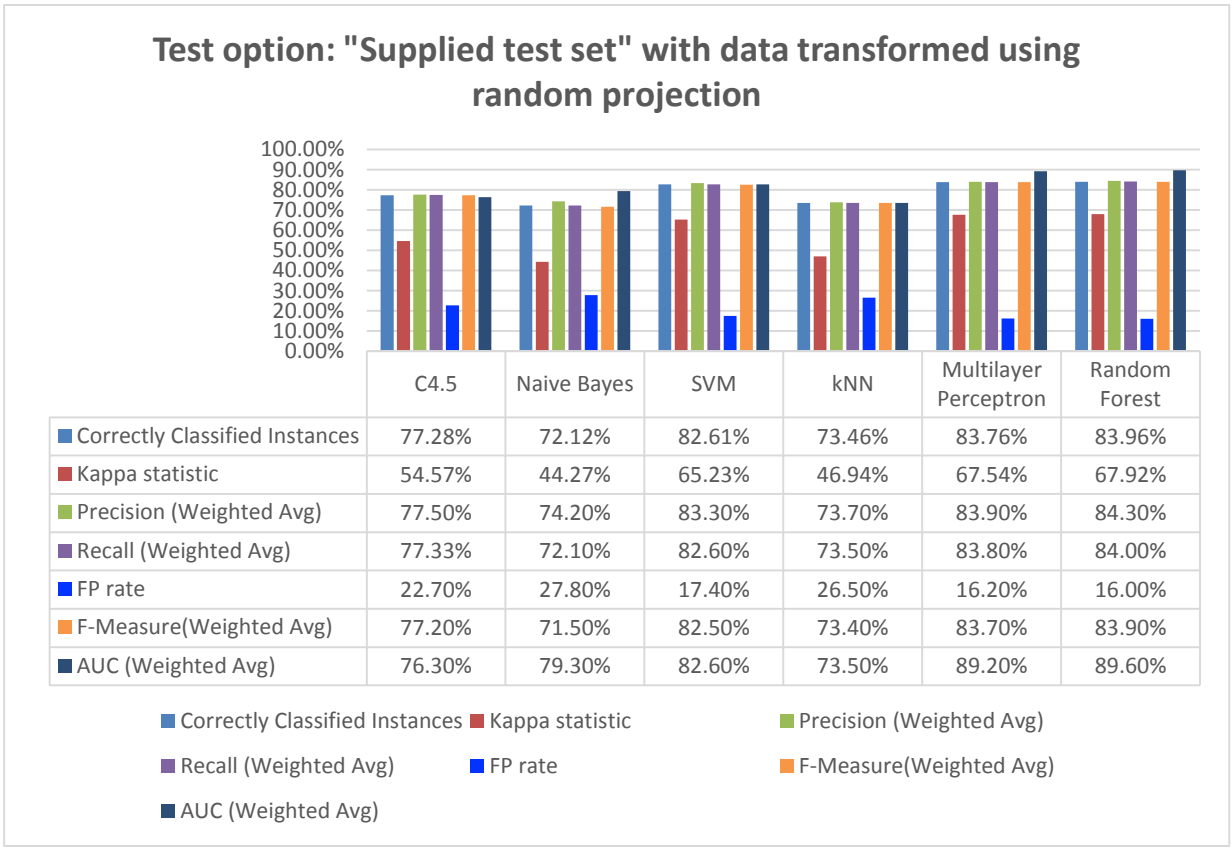


Figure 4.10: AUC score 2D -line chart for Vehicle dataset with no data transformation

With the application of random projection which reduced the dataset to 50 attributes. The overall performance of all the algorithms dropped slightly indicating towards an inefficient model design as seen in Figure 4.11. A drop in the AUC score for both the class labels can be seen in figure 4.12.



	C4.5	Naive Bayes	SVM	kNN	Multilayer Perceptron	Random Forest
Correctly Classified Instances	77.28%	72.12%	82.61%	73.46%	83.76%	83.96%
Kappa statistic	54.57%	44.27%	65.23%	46.94%	67.54%	67.92%
Precision (Weighted Avg)	77.50%	74.20%	83.30%	73.70%	83.90%	84.30%
Recall (Weighted Avg)	77.33%	72.10%	82.60%	73.50%	83.80%	84.00%
FP rate	22.70%	27.80%	17.40%	26.50%	16.20%	16.00%
F-Measure(Weighted Avg)	77.20%	71.50%	82.50%	73.40%	83.70%	83.90%
AUC (Weighted Avg)	76.30%	79.30%	82.60%	73.50%	89.20%	89.60%

Figure 4.11: Performance analysis report on Vehicle dataset transformed with Random Projection

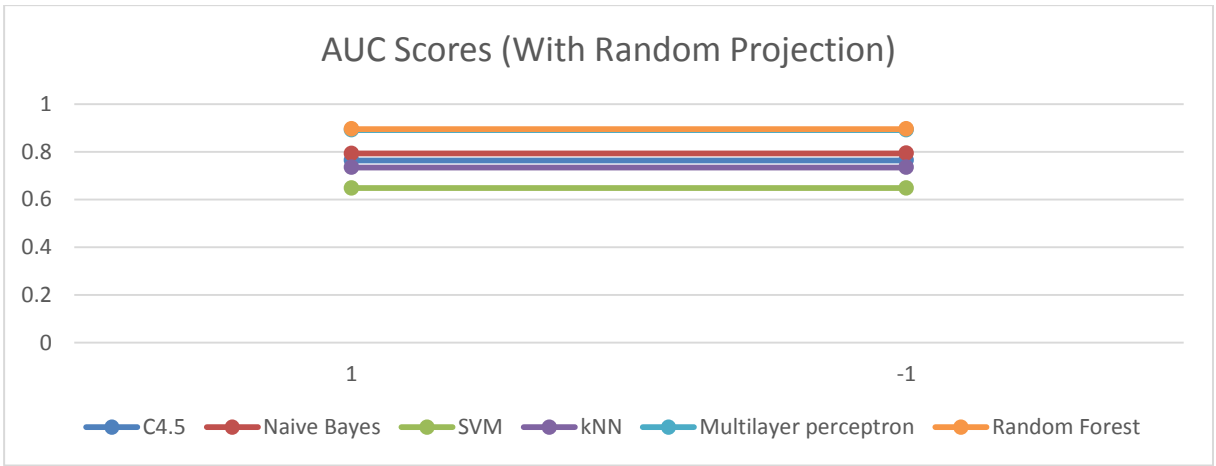
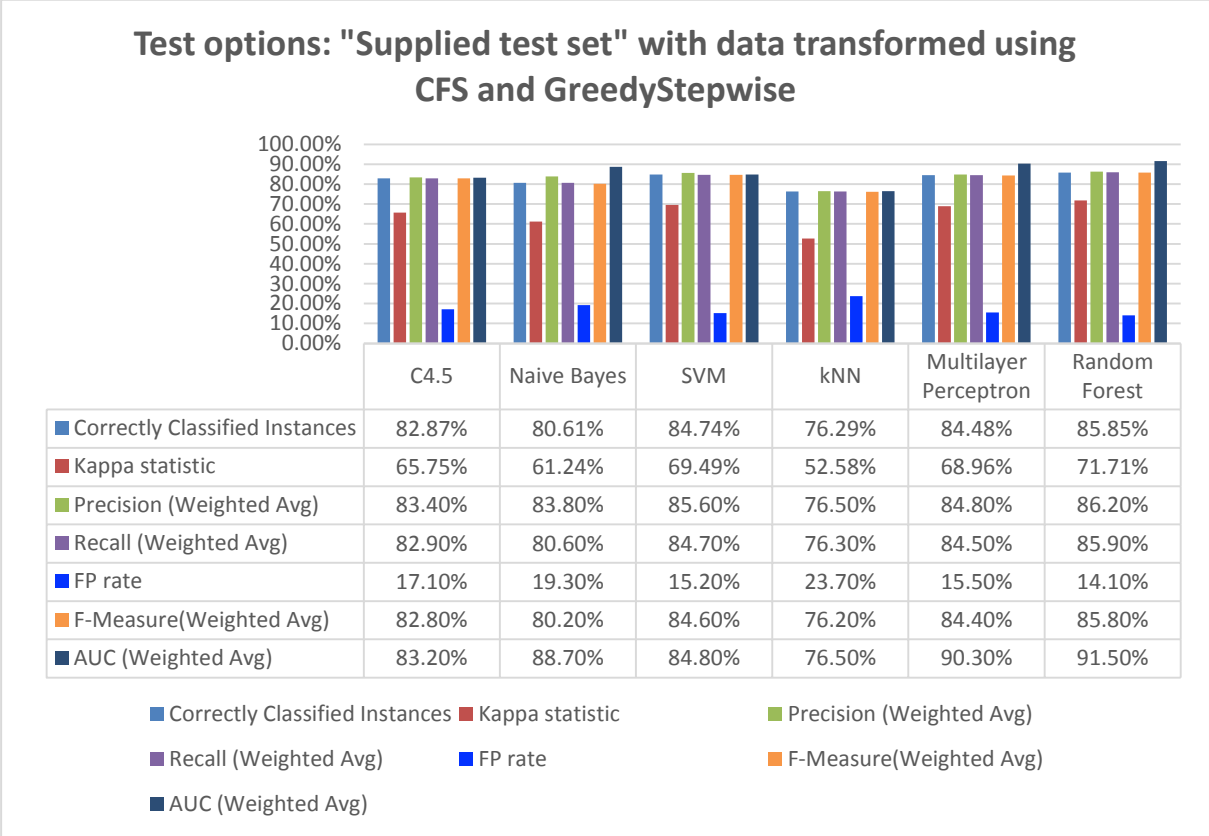


Figure 4.12: AUC score 2D -line chart for Vehicle data transformed with Random Projection

Application of CFS and GreedyStepwise data transformation which reduced the dataset to 34 attributes didn't show better results in performance except for multilayer perceptron which has an minor improvement in accuracy (84,48%) compared to model built with original dataset as seen in Figure 4.13.



	C4.5	Naive Bayes	SVM	kNN	Multilayer Perceptron	Random Forest
Correctly Classified Instances	82.87%	80.61%	84.74%	76.29%	84.48%	85.85%
Kappa statistic	65.75%	61.24%	69.49%	52.58%	68.96%	71.71%
Precision (Weighted Avg)	83.40%	83.80%	85.60%	76.50%	84.80%	86.20%
Recall (Weighted Avg)	82.90%	80.60%	84.70%	76.30%	84.50%	85.90%
FP rate	17.10%	19.30%	15.20%	23.70%	15.50%	14.10%
F-Measure(Weighted Avg)	82.80%	80.20%	84.60%	76.20%	84.40%	85.80%
AUC (Weighted Avg)	83.20%	88.70%	84.80%	76.50%	90.30%	91.50%

Figure 4.13: Performance analysis report on Vehicle dataset transformed with CFS and GreedyStepwise

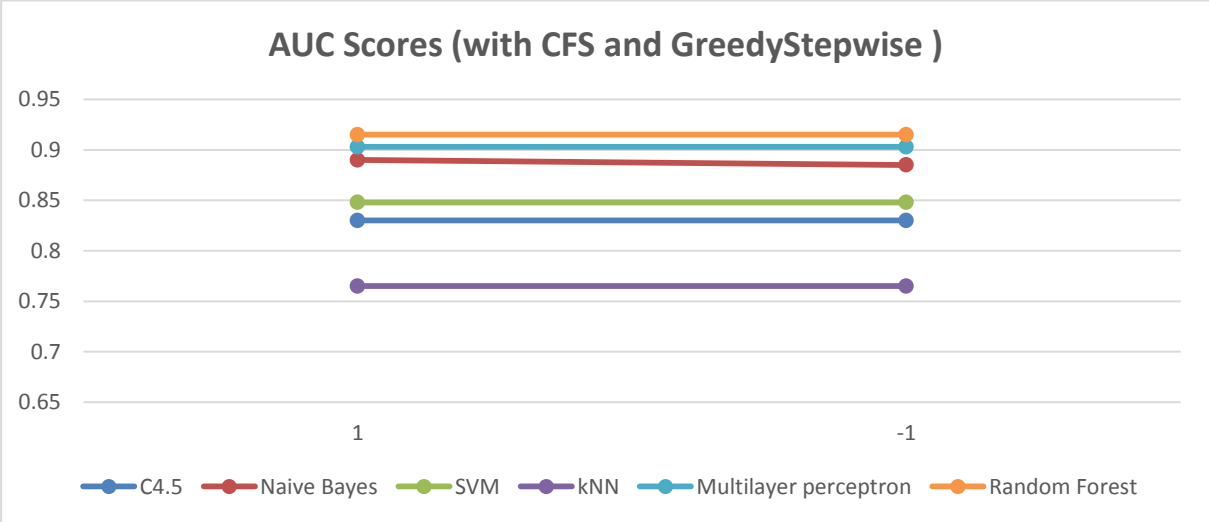
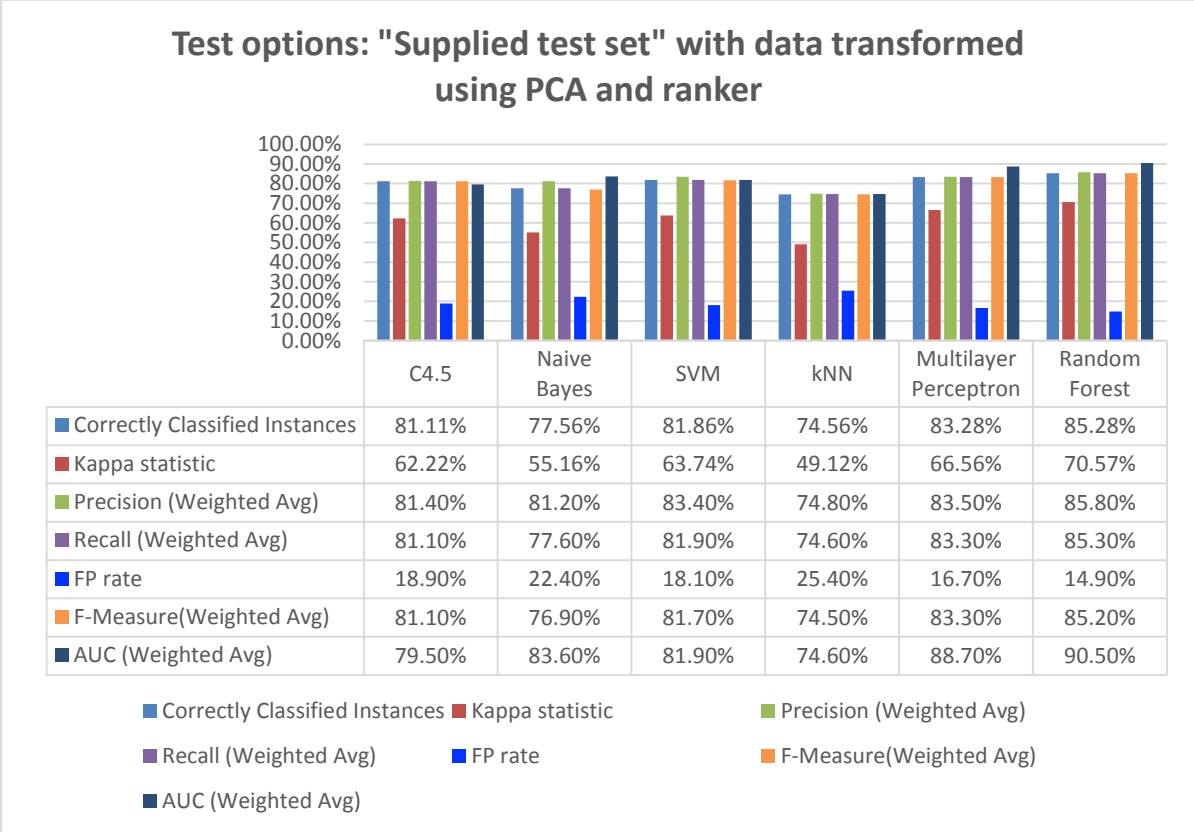


Figure 4.14: AUC score 2D -line chart for Vehicle data transformed with CFS and GreedyStepwise

As seen in figure 4.15 , 10-fold Cross validation performed on data transformed with PCA and ranker preprocessing techniques also did not give better results compared to output from models created on data with no transformation. The AUC score chart can be seen in Figure 4.16.



	C4.5	Naive Bayes	SVM	kNN	Multilayer Perceptron	Random Forest
Correctly Classified Instances	81.11%	77.56%	81.86%	74.56%	83.28%	85.28%
Kappa statistic	62.22%	55.16%	63.74%	49.12%	66.56%	70.57%
Precision (Weighted Avg)	81.40%	81.20%	83.40%	74.80%	83.50%	85.80%
Recall (Weighted Avg)	81.10%	77.60%	81.90%	74.60%	83.30%	85.30%
FP rate	18.90%	22.40%	18.10%	25.40%	16.70%	14.90%
F-Measure(Weighted Avg)	81.10%	76.90%	81.70%	74.50%	83.30%	85.20%
AUC (Weighted Avg)	79.50%	83.60%	81.90%	74.60%	88.70%	90.50%

Figure 4.15: Performance analysis report on Vehicle dataset transformed with PCA and Ranker

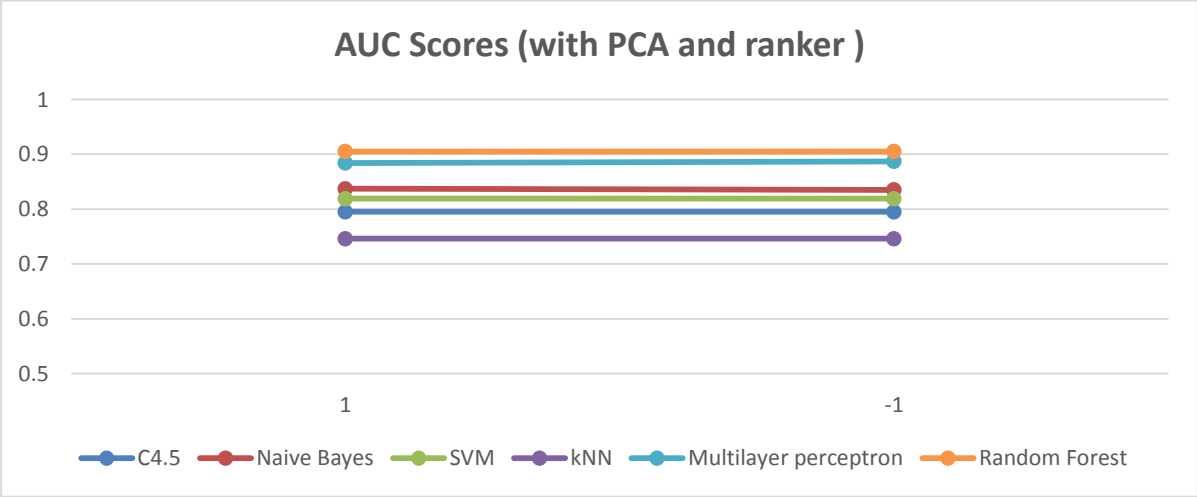


Figure 4.16: AUC score 2D -line chart for Vehicle data transformed with PCA and Ranker

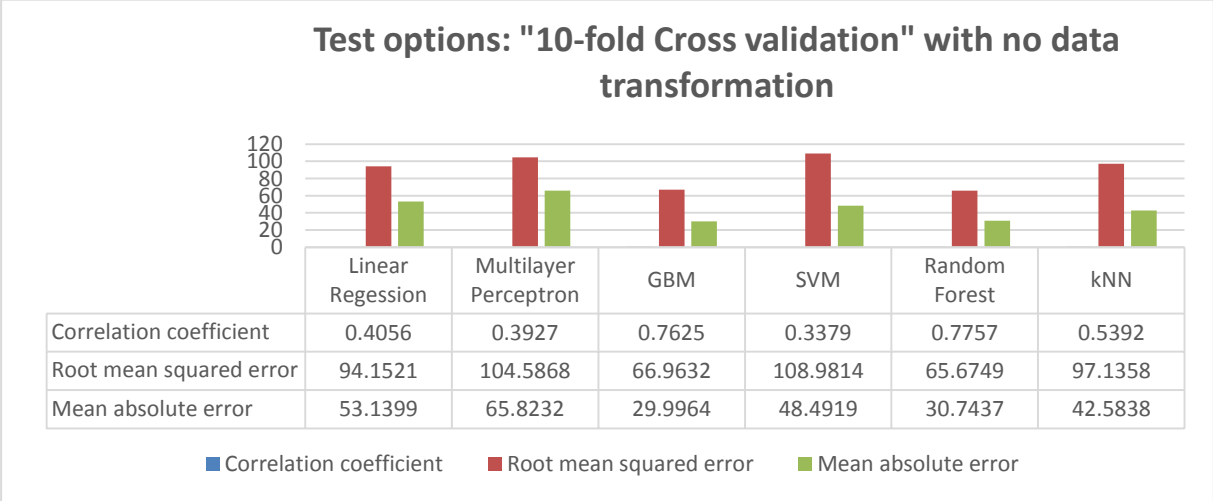
From the above experimentation it can be observed that Random Forest and multilayer models for vehicle dataset stand out in terms of high accuracy in all the test runs. Further analysis in section 5.1 will consolidate this initial findings.

4.3. Performance analysis results of regression algorithms

For regression problems schemes were cross-validated in terms of correlation coefficient, RMSE and MEA.

4.3.1. Experiment 3- Appliance energy prediction

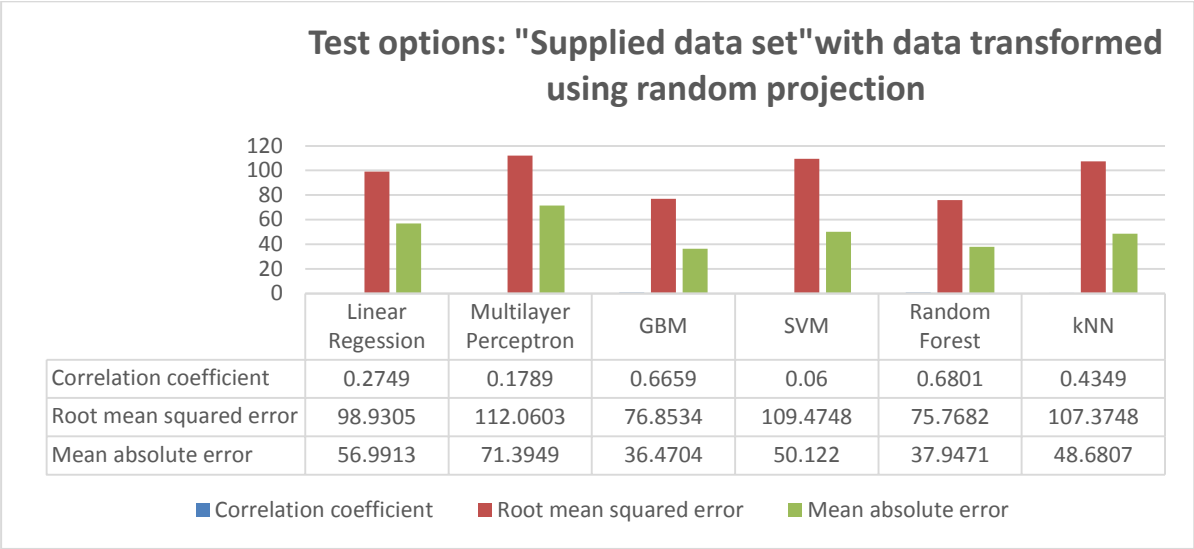
In experiment 3 ,regression models were built on appliance energy prediction dataset algorithms and evaluated with 10 fold cross validation. Correlation coefficient, RMSE and MAE metrics were measured and compared by running the WEKA test bench shown in Figure 3.3. Tests were performed first with the original data and then data was transformed with different preprocessing techniques. Figure 4.17 gives an overview of algorithms' performance when no data transformation was applied. As seen in Figure 4.17 , GBM and Random Forest have high correlation coefficient with value closer to 1 and have comparatively lower RMSE and MAE. While Random Forest had the lowest RMSE and highest correlation coefficient, SVM was the least efficient algorithm.



	Linear Regession	Multilayer Perceptron	GBM	SVM	Random Forest	kNN
Correlation coefficient	0.4056	0.3927	0.7625	0.3379	0.7757	0.5392
Root mean squared error	94.1521	104.5868	66.9632	108.9814	65.6749	97.1358
Mean absolute error	53.1399	65.8232	29.9964	48.4919	30.7437	42.5838

Figure 4.17: Performance analysis report on energy dataset with no data transformation

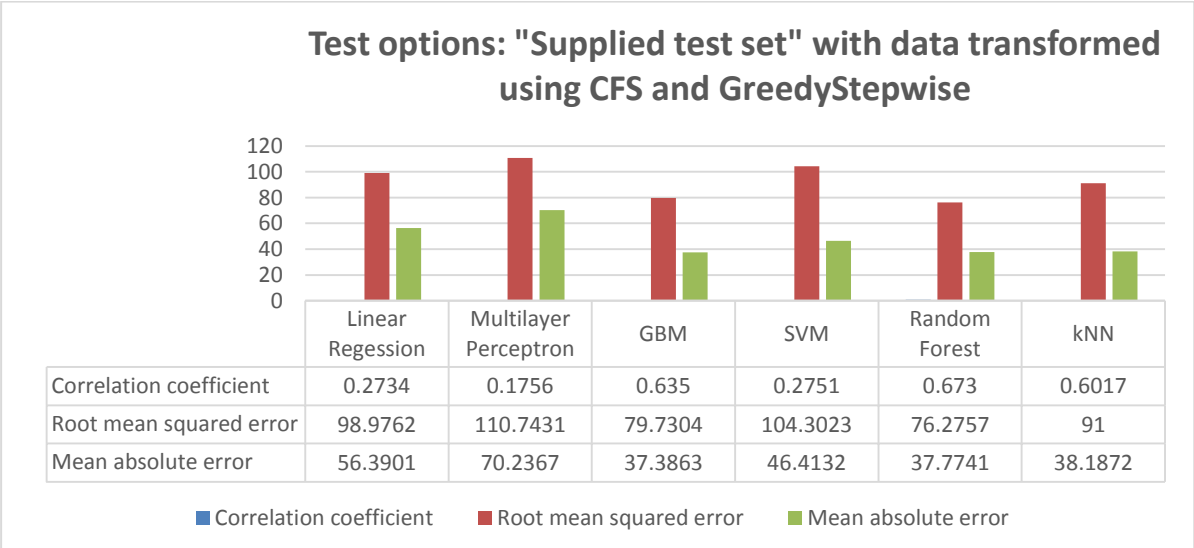
Application of random projection to the original dataset, which reduced the dataset to 11 attributes generated poorer results as seen in Figure 4.18.



	Linear Regession	Multilayer Perceptron	GBM	SVM	Random Forest	kNN
Correlation coefficient	0.2749	0.1789	0.6659	0.06	0.6801	0.4349
Root mean squared error	98.9305	112.0603	76.8534	109.4748	75.7682	107.3748
Mean absolute error	56.9913	71.3949	36.4704	50.122	37.9471	48.6807

Figure 4.18:Performance analysis report on energy dataset transformed with Random Projection

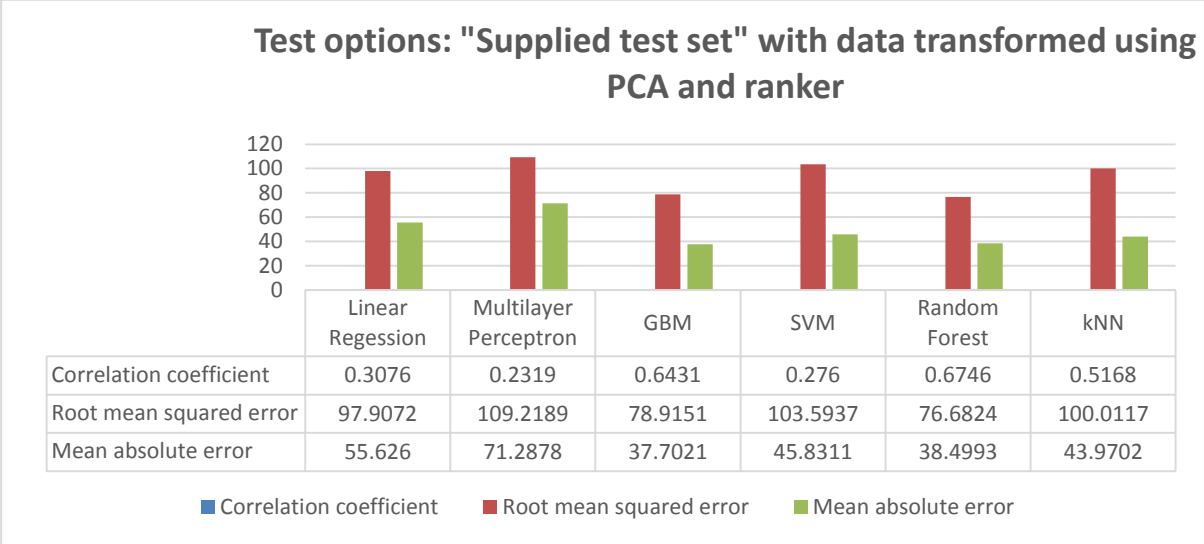
In the next experiment the data was transformed on WEKA workbench using CfsSubsetEval as attribute evaluator and GreedyStepwise for searching the attributes. The dataset was reduced to 6 attributes. It was observed that the data transformation improved the performance of kNN but degraded the performance of other algorithms as seen in figure 4.19.



	Linear Regession	Multilayer Perceptron	GBM	SVM	Random Forest	kNN
Correlation coefficient	0.2734	0.1756	0.635	0.2751	0.673	0.6017
Root mean squared error	98.9762	110.7431	79.7304	104.3023	76.2757	91
Mean absolute error	56.3901	70.2367	37.3863	46.4132	37.7741	38.1872

Figure 4.19: Performance analysis report on Energy dataset transformed with CFS and GreedyStepwise

Data was then transformed using attribute evaluator PCA and ranker as search method. This process transformed the dataset to 12 attribute dataset. The efficiency reduced for all algorithms as compared to original dataset as can be seen in Figure 4.20.



	Linear Regession	Multilayer Perceptron	GBM	SVM	Random Forest	kNN
Correlation coefficient	0.3076	0.2319	0.6431	0.276	0.6746	0.5168
Root mean squared error	97.9072	109.2189	78.9151	103.5937	76.6824	100.0117
Mean absolute error	55.626	71.2878	37.7021	45.8311	38.4993	43.9702

Figure 4.20:Performance analysis report on Energy dataset transformed with PCA and Ranker

It can be deduced from the experimentation performed on appliances energy prediction dataset that data transformation deteriorate the overall performance and was not worth the application. Random forest and GBM had the best evaluation metric statistics as compared to other selected algorithms. Further extensive validation process covered in chapter 5 assert the outcome of initial findings.

4.3.2. Experiment 4-Puma 560 robot arm

Regression algorithms were evaluated for Puma 560 robot arm dataset analysis using similar test setup as of appliance energy prediction dataset. Correlation coefficient, RMSE and MAE metrics were measured and compared by running the WEKA test bench shown in Figure 3.3. Tests were performed first with the original data and then data was transformed with different preprocessing techniques.

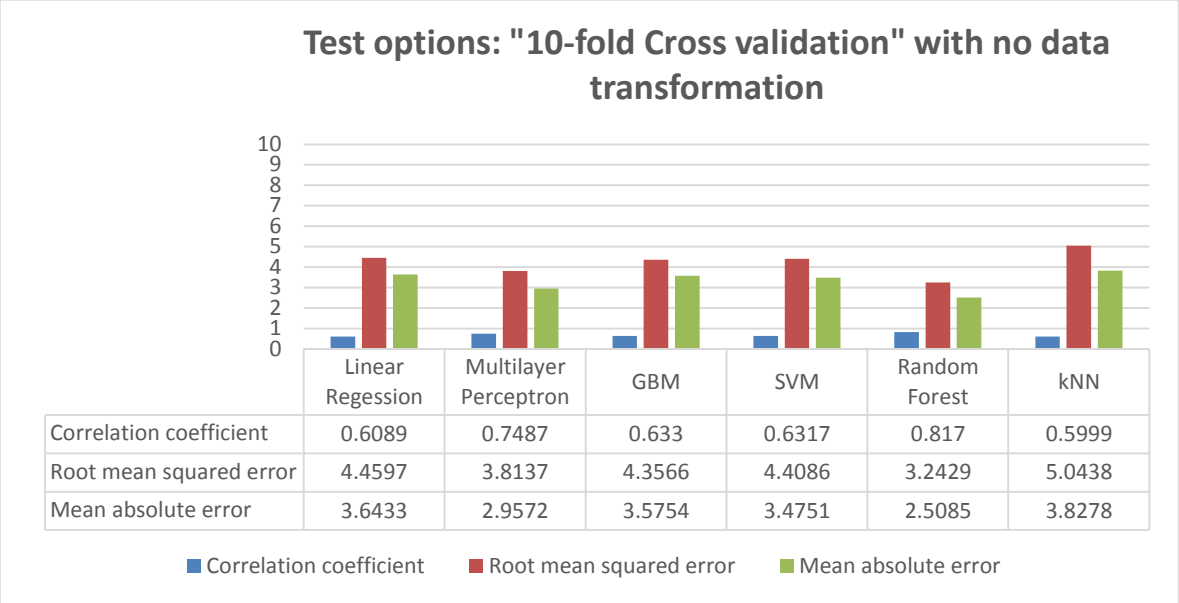


Figure 4.21: Performance analysis report on Puma 560 robot arm dataset with no transformation

As seen in Figure 4.21 models build with Random Forest and Multilayer perceptron were satisfactory with a low RMSE and high correlation coefficient. In this dataset as well random forest had the best results with a RMSE value of 3.2429 and correlation coefficient of 0.817. Models generated from data transformed with random projection generated a significantly poor validation results as seen from the number in Figure 4.22.

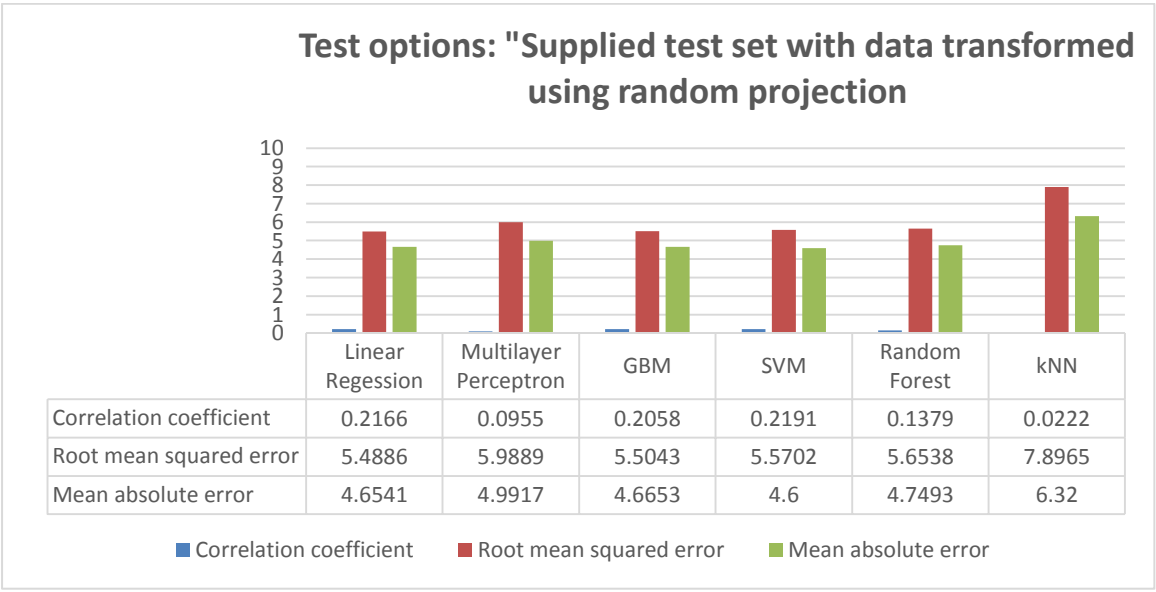
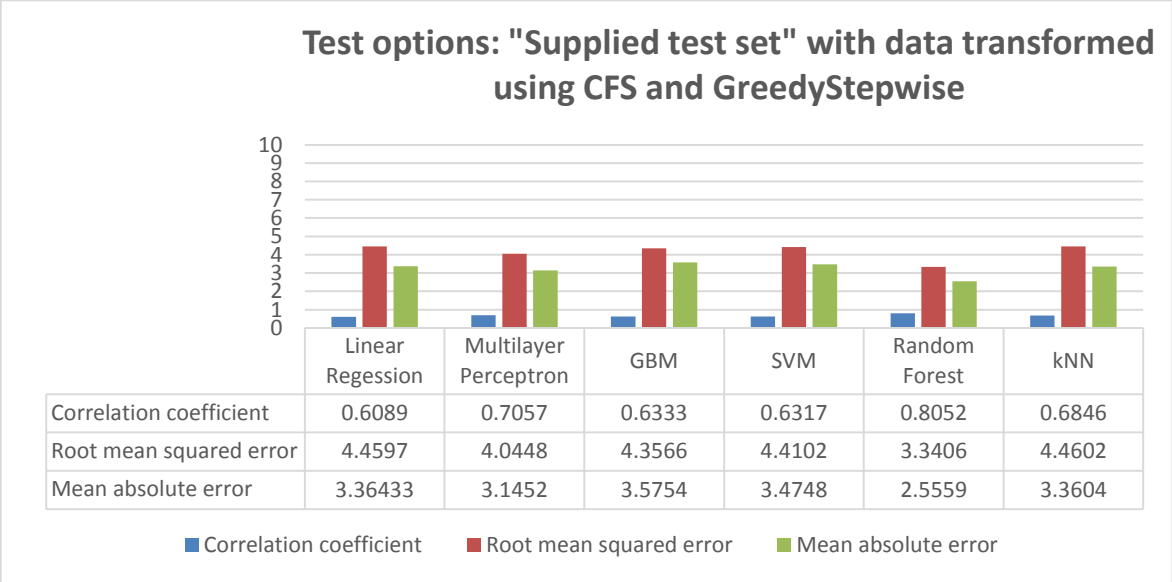


Figure 4.22: : Performance analysis report on Puma 560 robot arm dataset transformed with Random Projection

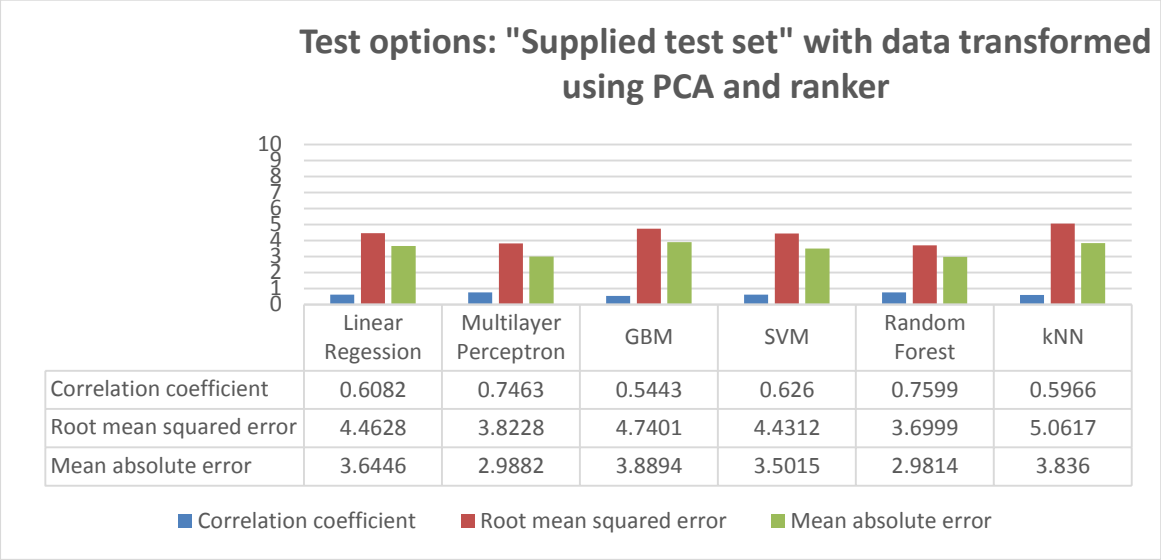
As seen in Figure 4.23 , poor results were reported for all models generated for data transformed with CFS and GreedyStepwise except kNN compared to model created with the original data(Figure 4.21). A RMSE value less than 5 and correlation coefficient of 0.6846 was observed.



	Linear Regession	Multilayer Perceptron	GBM	SVM	Random Forest	kNN
Correlation coefficient	0.6089	0.7057	0.6333	0.6317	0.8052	0.6846
Root mean squared error	4.4597	4.0448	4.3566	4.4102	3.3406	4.4602
Mean absolute error	3.36433	3.1452	3.5754	3.4748	2.5559	3.3604

Figure 4.23:Performance analysis report on Puma 560 robot arm dataset transformed with CFS and GreedyStepwise

Classifiers' evaluation with data transformed using with PCA and ranker method with original dataset being used as test data and transformed data being used as training data, also showed negligible improvement in the performance of the models and results we similar to the models generated with the original data. (Figure 4.24).



	Linear Regession	Multilayer Perceptron	GBM	SVM	Random Forest	kNN
Correlation coefficient	0.6082	0.7463	0.5443	0.626	0.7599	0.5966
Root mean squared error	4.4628	3.8228	4.7401	4.4312	3.6999	5.0617
Mean absolute error	3.6446	2.9882	3.8894	3.5015	2.9814	3.836

Figure 4.24: Performance analysis report on Puma 560 robot arm dataset transformed with PCA and Ranker

Chapter 5

This chapter covers the an empirical comparative analysis of machine learning algorithms studied in chapter 4. In the first section , paired t-test is performed on the selected datasets and algorithms are validated in terms of chosen evaluation metrics. The next section provides a conclusive scrutiny of the analysis results from chapter 4 and section 5.1. This study is finalized with an overall summary in section 5.3.

5.1 T-Test performance comparison

In many big data problems, a basic 10 fold cross validation on individual dataset is not a satisfactory decisive process for finalizing the optimal machine learning scheme for analytics design. A further extensive test is commonly carried out when conducting a performance comparative analysis. A repeated cross validation is performed for each algorithm for multiple datasets and a mean of (error estimates)/accuracy is computed to determine whether the mean of one data sample is significantly greater or less than other.. WEKA Experimenter interface uses paired t-test to compare the mean of two different data samples where the observation of one sample is paired with other observation. The conceptual background of paired t test has been presented in section 2.

WEKA Experimenter paired t-test is conducted in a three step using three panels in the GUI shown in Figure 5.1 . In Setup panel datasets are and algorithms are selected. The Run panel is used to process 10-fold cross validation multiplied by 10 repeats per algorithm for each dataset. Paired t test is performed in the Analyse panel. The confidence level is by default set to 0.05 and the comparison metric is “percent correct” by default. The base algorithm is selected in the test configuration settings. Section 5.1.1 discusses the test results for classification data samples and section 5.1.2 covers the regression data samples test results.

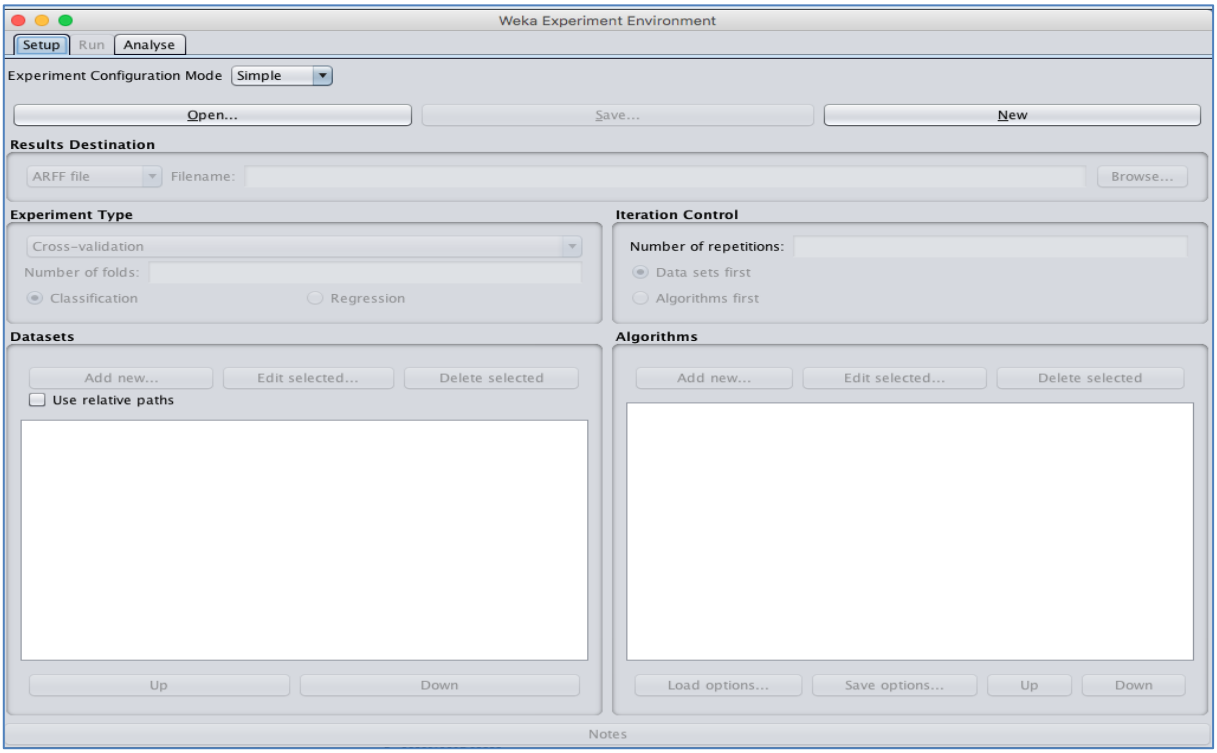


Figure 5.1 WEKA Experimenter

5.1.1: T-test analysis of classification algorithms

The Experimenter was configured to perform paired t-test on the original HAR and vehicle dataset to analyze the classifiers Random Forest, C4,5, Naïve Bayes, SVM and kNN. Random forest was configured as the baseline scheme. Accuracy("percent_correct") evaluation metric is used to compare the performance. The symbol "v" and "*" beside the result indicates that the method is statistically high value(v) or low value(*) than the base method which is Random Forest in this case at a significance level of 5%. At the bottom of each column are counts $(x/y/z)$ which indicate the number of times the scheme is better or worse than the base scheme(x being high , y being the same , z is low value) as shown in Figure 5.3. It was observed from the test outcome that SVM with an accuracy of 98.49% was rated the best algorithm with highest "percent correct" value for HAR dataset. The number of times it is better than Random forest is 1. For vehicle dataset , none of the scheme's performance was better than the baseline scheme i.e. Random Forest with accuracy of 86.26% . The bottom row statistics shows that C4.5 , Naive Bayes, kNN are worse than Random forest with a factor of 2 while SVM was worse with factor of 1. Multilayer perceptron was excluded from the paired t-test analysis of the original dataset because it was extremely slow during the initial analysis as observed in section 4.1.

```
Tester:     weka.experiment.PairedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixPlainText -mean-prec 2 -stddev-prec 2
-col-name-width 0 -row-name-width 25 -mean-width 2 -stddev-width 2 -sig-width 1 -count-width 5 -print-col-names -print-row-names -enum-col-names"
Analysing:  Percent_correct
Datasets:   2
Resultsets: 5
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       3/11/18 9:52 PM


Dataset                  (1) trees.Ra | (2) trees (3) bayes (4) lazy. (5) funct
-----------------------------------------------------------------------------
hardata_complete         (100)   98.09 |   94.41 *   75.05 *   97.20 *   98.49 v
'vehicle-weka.filters.uns(100)   86.26 |   82.64 *   80.44 *   72.54 *   84.95 *
-----------------------------------------------------------------------------
                               (v/ /*) |   (0/0/2)   (0/0/2)   (0/0/2)   (1/0/1)


Key:
(1) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(2) trees.J48 '-C 0.25 -M 2' -217733168393644444
(3) bayes.NaiveBayes '' 5995231201785697655
(4) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(5) functions.SMO '-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.RBFKernel -C 250007 -G 0.01\"' -7149606251113102827
```

Figure 5.2: Paired T-test "Percent_Correct" analysis results of classifiers on original HAR and Vehicle dataset

Since satisfactory results were seen with datasets transformed with CFS and GreedyStepwise method, paired t-test was run on both the transformed datasets as well. Results indicated Random Forest as the most efficient scheme for both the datasets as illustrated in Figure 5.4. Multilayer perceptron was also included in the test run since promising results were visible in the initial analysis in section 4.1 and also good results were seen in paired t-test with a 96.49% "percent correct". The $(x/y/z)$ statistics in the bottom row shows that all schemes are worse than the base scheme by factor of 2 which indicate Random forest as a clear winner for both type of analytics design.

```
Tester:     weka.experiment.PairedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixPlainText -mean-prec 2 -stddev-prec 2
-col-name-width 0 -row-name-width 25 -mean-width 2 -stddev-width 2 -sig-width 1 -count-width 5 -print-col-names -print-row-names -enum-col-names"
Analysing:  Percent_correct
Datasets:   2
Resultsets: 5
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       3/11/18 9:52 PM


Dataset                  (1) trees.Ra | (2)functions.M (3)functions.S (4)lazy.IBk (5)trees.J48 (4) bayes.Naive
-----------------------------------------------------------------------------------------------------------
hardata_complete         (100)   97.74 |   96.49 *      93.65 *         93.83 *     94.23 *       89.14 *
'vehicle-weka.filters.uns(100)   85.95 |   84.72 *      84.67 *         76.28 *     83.00 *       80.60 *
-----------------------------------------------------------------------------------------------------------
                               (v/ /*) |   (0/0/2)      (0/0/2)       (0/0/2)       (0/0/2)       (0/0/2)


Key:
(1) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(2) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(3) functions.SMO '-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.RBFKernel -C 250007 -G 0.01\"' -7149606251113102827
(5) trees.J48 '-C 0.25 -M 2' -217733168393644444
(6) bayes.NaiveBayes '' 5995231201785697655
(4) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
```

Figure 5.3: Paired T-test "Percent_Correct" analysis results of algorithms on HAR and Vehicle dataset transformed with CFS and GreedyStepwise.

5.1.2 T-test analysis of regression algorithms

The Experimenter was configured to perform paired t-test on the original Energy and puma8NH dataset to analyze the regression models built with Linear regression, Random Forest, GBM, multilayer perceptron, SVM and kNN. Random Forest is configured as the baseline scheme. Correlation coefficient and RMSE evaluation metrics are used to measure perform a comparative analysis. Due to limited computation capability, paired t-test was run on each datasets once at a time. It was observed that Random Forest outperformed all other schemes in terms of RMSE as it had significantly lower value than other schemes. Figure 5.5 illustrate the test results for energy dataset. Random Forest also had an optimal correlation coefficient compared to other schemes as illustrated in Figure 5.6.

```
Tester:     weka.experiment.PairedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixPlainText -mean-prec 2 -stddev-prec 2 -col-
name-width 0 -row-name-width 25 -mean-width 0 -stddev-width 0 -sig-width 0 -count-width 5 -print-col-names -print-row-names -enum-col-names"
Analysing:  Root_mean_squared_error
Datasets:   1
Resultsets: 6
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       3/7/18 6:18 PM


Dataset                   (1) trees.Ra | (2) lazy. (3) funct (4) meta.A (5) functi (6) funct
-----------------------------------------------------------------------------------------------
'energydata_complete-weka(100)   66.57 |   97.28 v   98.75 v   102.81 v   107.37 v   94.09 v
-----------------------------------------------------------------------------------------------
                              (v/ /*) |   (1/0/0)   (1/0/0)    (1/0/0)    (1/0/0)   (1/0/0)


Key:
(1) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(2) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(3) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(4) meta.AdditiveRegression '-S 0.0 -I 10 -W trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1' -2368937577670527151
(5) functions.SMOreg '-C 1.0 -N 2 -I \"functions.supportVector.RegSMOImproved -T 0.001 -V -P 1.0E-12 -L 0.001 -W 1\" -K \"functions.supportVector.RBFKernel
-C 250007 -G 0.01\"' -7149606251113102827
(6) functions.LinearRegression '-S 0 -R 1.0E-8 -num-decimal-places 4' -3364580862046573747
```

Figure 5.4: Paired T-test "RMSE" analysis of algorithms energy dataset

```
Tester:     weka.experiment.PairedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixPlainText -mean-prec 2 -stddev-prec 2 -col-
name-width 0 -row-name-width 25 -mean-width 0 -stddev-width 0 -sig-width 0 -count-width 5 -print-col-names -print-row-names -enum-col-names"
Analysing:  Correlation_coefficient
Datasets:   1
Resultsets: 6
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       3/7/18 6:18 PM


Dataset                   (1) trees.Ra | (2) lazy. (3) funct (4) meta.A (5) functi (6) funct
-----------------------------------------------------------------------------------------------
'energydata_complete-weka(100)    0.77 |    0.54 *   0.51 *     0.00 *     0.34 *    0.40 *
-----------------------------------------------------------------------------------------------
                              (v/ /*) |   (0/0/1)   (0/0/1)    (0/0/1)    (0/0/1)   (0/0/1)


Key:
(1) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(2) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(3) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(4) meta.AdditiveRegression '-S 0.0 -I 10 -W trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1' -2368937577670527151
(5) functions.SMOreg '-C 1.0 -N 2 -I \"functions.supportVector.RegSMOImproved -T 0.001 -V -P 1.0E-12 -L 0.001 -W 1\" -K \"functions.supportVector.RBFKernel
-C 250007 -G 0.01\"' -7149606251113102827
(6) functions.LinearRegression '-S 0 -R 1.0E-8 -num-decimal-places 4' -3364580862046573747
```

Figure 5.5: Paired T-test "Correlation coefficient" analysis of algorithms energy dataset

The next paired t-test was run on puma dataset with same configuration as energy dataset. In puma dataset Random Forest was observed as the optimal scheme based on the best values of RMSE and correlation coefficient compared to other schemes as illustrated in Figure 5.7 and 5.8.

```
Tester:     weka.experiment.PairedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixPlainText -mean-prec 2 -stddev-prec 2 -col-
name-width 0 -row-name-width 25 -mean-width 2 -stddev-width 2 -sig-width 1 -count-width 5 -print-col-names -print-row-names -enum-col-names"
Analysing:  Root_mean_squared_error
Datasets:   1
Resultsets: 6
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       3/11/18 3:55 PM


Dataset                   (1) trees.R | (2) func (3) lazy (4) meta (5) func (6) func
----------------------------------------------------------------------------------
puma8NH                 (100)   3.23 |   3.88 v   5.03 v   3.56 v   4.46 v   4.41 v
----------------------------------------------------------------------------------
                              (v/ /*) |  (1/0/0)  (1/0/0)  (1/0/0)  (1/0/0)  (1/0/0)


Key:
(1) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(2) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(3) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(4) meta.AdditiveRegression '-S 0.3 -I 10 -W trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1' -2368937577670527151
(5) functions.LinearRegression '-S 0 -R 1.0E-8 -num-decimal-places 4' -3364580862046573747
(6) functions.SMOreg '-C 1.0 -N 2 -I \"functions.supportVector.RegSMOImproved -T 0.001 -V -P 1.0E-12 -L 0.001 -W 1\" -K \"functions.supportVector.RBFKernel
-C 250007 -G 0.01\"' -7149606251113102827
```

Figure 5.6: Paired T-test "RMSE" analysis of algorithms puma8NH dataset

```
Tester:     weka.experiment.PairedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixPlainText -mean-prec 2 -stddev-prec 2 -col-
name-width 0 -row-name-width 25 -mean-width 2 -stddev-width 2 -sig-width 1 -count-width 5 -print-col-names -print-row-names -enum-col-names"
Analysing:  Correlation_coefficient
Datasets:   1
Resultsets: 6
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       3/11/18 4:04 PM


Dataset                   (1) trees.R | (2) func (3) lazy (4) meta (5) func (6) func
----------------------------------------------------------------------------------
puma8NH                 (100)   0.82 |   0.80 *   0.60 *   0.78 *   0.61 *   0.63 *
----------------------------------------------------------------------------------
                              (v/ /*) |  (0/0/1)  (0/0/1)  (0/0/1)  (0/0/1)  (0/0/1)


Key:
(1) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(2) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(3) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(4) meta.AdditiveRegression '-S 0.3 -I 10 -W trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1' -2368937577670527151
(5) functions.LinearRegression '-S 0 -R 1.0E-8 -num-decimal-places 4' -3364580862046573747
(6) functions.SMOreg '-C 1.0 -N 2 -I \"functions.supportVector.RegSMOImproved -T 0.001 -V -P 1.0E-12 -L 0.001 -W 1\" -K \"functions.supportVector.RBFKernel
-C 250007 -G 0.01\"' -7149606251113102827
```

Figure 5.7: Paired T-test "Correlation Coefficient" analysis of algorithms puma8NH dataset

T-test comparative analysis results indicated Random Forest as the best scheme with optimal accuracy and RMSE. SVM proved to be the optimal in terms of accuracy in case of HAR dataset when performed with no data transformation whereas overall analysis results indicated that Random Forest generate optimal results in different test scenarios.

Conclusion

The intent of this thesis is to exhibit an analogical perspective of different machine learning algorithms to aid in the data analytics design process. In this study, diverse datasets were subjected to selected machine learning algorithms and models' performance was cross-validated using WEKA and validation results were scrutinized. A separate test was conducted for regression and classification problems using two datasets in each data mining category. Most of the algorithms chosen are capable of handling both regression and classification predictive tasks, hence aided in the algorithm analogy. The initial test was performed with 10-fold cross-validation to measure the efficiency of each algorithm with respect to the specific dataset. Evaluation metrics such as accuracy, Kappa statistic, Precision, recall, FP rate, f-measure and AUC formed the basis of evaluation for classification problems and RMSE and correlation coefficient for regression type data problems. Datasets were also transformed using preprocessing techniques and then used as test set to access the impact of preprocessing on the performance of the algorithms. In the initial evaluation with 10-fold cross validation, it was found that linearity and structure of the dataset has a visible impact on accuracy percentage and the error rate. As seen in case of HAR dataset where the dataset is more structured and linear, the error rate is low and accuracy is high compared to results from another dataset. It was also seen that with the chosen dataset, preprocessing had no significant impact on the performance of the algorithms except for a couple of specific cases such as kNN had better efficiency when applied to datasets transformed with CFS and GreedyStepwise techniques. Multilayer perceptron also showed improvement in case of HAR dataset transformed with CFS and GreedyStepwise techniques. It was also seen that in case of HAR dataset, with the transformed data which reduced the dataset to 58 attributes from 561 attributes, the overall efficiency of algorithms did not degrade significantly, hence can be stated as an optimal preprocessing method for classification datasets with respect to data storage cost. But in case of regression dataset, the difference in performance was evident and hence data transformation cannot be applied to datasets to achieve an efficient data model. Random Forest is seen as optimal modeling scheme for all the datasets selected for this study .The better performance of Random Forest schemes can be justified from its basic principle which combines multiple tree predictors at training time where the final prediction is made from the average of each prediction of the individual trees using the bootstrap averaging(bagging) techniques. This process leads to significant reduction in variance and overfitting caused by single tree predictors. Another important factor that makes Random Forest one of

the most preferred model schemes is that Random Forest requires minimal parameter tuning and is simple to implement to generate a decent model which outputs faster and efficient prediction. Finally, it can be stated that the efficiency of a model is greatly dependent on the dataset characteristics and a rigorous scrutiny of the predictive capability of algorithms with respect to the data problem in question and a comprehensive cost/benefit estimate are the two key aspects to designing an intelligent data analytics system.

References

[1] Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques*. Burlington, MA: Elsevier.

[2] Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining: Practical machine learning tools and techniques*. San Francisco: Morgan Kaufman.

[3] He, S., Chen, J., Yau, D. K., & Sun, Y. (2010). Cross-Layer Optimization of Correlated Data Gathering in Wireless Sensor Networks. 2010 *7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. doi:10.1109/secon.2010.5508271.

[4] Li, H., Yu, H., & Liu, A. (n.d.). A Tree Based Data Collection Scheme for Wireless Sensor Network. *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL06)*. doi:10.1109/icniconsmcl.2006.36

[5] Abbasi, A. A., & Younis, M. (2007). A survey on clustering algorithms for wireless sensor networks. *Computer Communications,30*(14-15), 2826-2841. doi:10.1016/j.comcom.2007.05.024

[6] Zhang, S., Zhang, C., & Yang, Q. (2003). Data preparation for data mining. *Applied Artificial Intelligence,17*(5-6), 375-381. doi:10.1080/713827180

[7] Z. Marzuki and F. Ahmad, "Data Mining Discretization Methods and Performances," Proceedings of the International Conference on Electrical Engineering and Informatics, Institute Teknologi Bandung, Bandung, 17-19 June 2007, pp. 535-537

[8] M. (n.d.). Mining Models (Analysis Services - Data Mining). Retrieved April 02, 2018, from https://docs.microsoft.com/en-us/sql/analysis-services/data-mining/mining-models-analysis-services-data-mining.

[9] Hale, R., & Rollins, J. B. (2006, March 13). Embedded data mining steps to success. Retrieved April 02, 2018, from https://www.computerworld.com/article/2562243/business-intelligence/embedded-data-mining-steps-to-success.html

[10] M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," in IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 6, pp. 1437-1447, Nov.-Dec. 2003. doi: 10.1109/TKDE.2003.1245283

[11] Novakovic, J. (2009). Using Information Gain Attribute Evaluation to Classify Sonar Targets.

[12] Pierre Dangauthier, Pierre Bessiere, Anne Spalanzani. Feature Selection For Self-Supervised Learning. [Technical Report] 2005

[13] Liu, H. et. al(2002) : Discretization : An Enabling Technique.Data Mining and Knowledge Discovery, 6,393-423.

[14] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24, 417–441, and 498–520.

[15] Jonathon Shlens (2014). A Tutorial on Principal Component Analysis. CoRR, abs/1404.1100, .

[16] Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. *Conference on Modern Analysis and Probability Contemporary Mathematics,*189-206. doi:10.1090/conm/026/737400

[17] Ailon, N., & Chazelle, B. (2009). The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM Journal on Computing,39*(1), 302-322. doi:10.1137/060673096

[18] Achlioptas, D. (2003). Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences,66*(4), 671-687. doi:10.1016/s0022-0000(03)00025-4

[19] Ron Kohavi(1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *In Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2 (*IJCAI'95), Vol. 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1137-1143.

[20] Refaeilzadeh, P., Tang, L., & Liu, H. (2016). Cross-Validation. *Encyclopedia of Database Systems,*1-7. doi:10.1007/978-1-4899-7993-3_565-2

[21] Britannica, T. E. (2016, April 11). Student's t-test. Retrieved April 03, 2018, from https://www.britannica.com/science/Students-t-test

[22] T Test (Students T-Test): Definition and Examples. (n.d.). Retrieved April 03, 2018, from http://www.statisticshowto.com/probability-and-statistics/t-test/

[23] Duriqi, R., Raca, V., & Cico, B. (2016). Comparative analysis of classification algorithms on three different datasets using WEKA. *2016 5th Mediterranean Conference on Embedded Computing (MECO)*. doi:10.1109/meco.2016.7525775

[24] Haigh, K.Z., Mackay, A.M., Cook, M.R., & Lin, L.G. (2015). Machine Learning for Embedded Systems: A Case Study.

[25] Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2018). High-order possibilistic c-means algorithms based on tensor decompositions for big data in IoT. *Information Fusion,39*, 72-80. doi:10.1016/j.inffus.2017.04.002

[26] Breiman, L. (2001). *Machine Learning,45*(1), 5-32. doi:10.1023/a:1010933404324

[27] Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. *Multiple Classifier Systems Lecture Notes in Computer Science,*1-15. doi:10.1007/3-540-45014-9_1

[28] Breiman, L. (1996). *Machine Learning,24*(2), 123-140. doi:10.1023/a:1018054314350

[29] Ho, T. K. (n.d.). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*. doi:10.1109/icdar.1995.598994
[30] Jain, A., Mao, J., & Mohiuddin, K. (1996). Artificial neural networks: A tutorial. *Computer,29*(3), 31-44. doi:10.1109/2.485891

[31] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., . . . Steinberg, D. (2007). Top 10 algorithms in data mining. *Knowledge and Information Systems,14*(1), 1-37. doi:10.1007/s10115-007-0114-2

[32] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning,20*(3), 273-297. doi:10.1007/bf00994018

[33] Chang, C., & Lin, C. (2011). Libsvm. *ACM Transactions on Intelligent Systems and Technology,2*(3), 1-27. doi:10.1145/1961189.1961199

[34] Haykin, S. S. (1999). Neural networks: A comprehensive foundation. London: Prentice-Hall International.

[35] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE,78*(9), 1464-1480. doi:10.1109/5.58325

[36] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[37] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. *A Public Domain Dataset for Human Activity Recognition Using Smartphones*. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.

[38] LIBSVM Data: Classification, Regression, and Multi-label. (n.d.). Retrieved April 03, 2018, from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[39] Duarte, M. F., & Hu, Y. H. (2004). Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing,64*(7), 826-838. doi:10.1016/j.jpdc.2004.03.020

[40] Candanedo, L. M., Feldheim, V., & Deramaix, D. (2017). Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings,140*, 81-97. doi:10.1016/j.enbuild.2017.01.083

[41] Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis,38*(4), 367-378. doi:10.1016/s0167-9473(01)00065-2

[42] Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment,62*(1), 77-89. doi:10.1016/s0034-4257(97)00083-7

[43] Zweig, M H & Campbell, G (1993). *Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine.* Clinical Chemistry, 39, 561-577.

[44] Product List. (n.d.). Retrieved April 03, 2018, from https://www.dtlr.com/brands/puma.html

[45] Pentaho Acquires Weka Project. (2006, September 19). Retrieved April 08, 2018, from http://www.pentaho.com/pentaho-acquires-weka-project

Appendix A

WEKA algorithm configuration setting:

CfsSubsetEval : *weka.attributeSelection.CfsSubsetEval -P 1 -E 1*

missingSeparate:- if set then treat missing as a separate value. Otherwise, counts for missing values are distributed across other values in proportion to their frequency.
preComputeCorrelationMatrix :- Precompute the full correlation matrix at the outset, rather than compute correlations lazily (as needed) during the search.
locallyPredictive :- Identify locally predictive attributes.

GreedyStepwise: *weka.attributeSelection.GreedyStepwise -T -1.7976931348623157E308 -N -1 -num-slots 1*

generateRanking :- if true generate a ranked list. Default value is false.
numToSelect:- Specify the number of attributes to retain. The default value (-1) indicates that all attributes are to be retained.
searchBackwards:- Search backwards rather than forwards. Default is false.
threshold:- Set threshold for filtering the attributes. Default value results in no attributes being discarded.
startSet:- Set the start point for the search. This property is not set by default.

Principal Components: *weka.attributeSelection.PrincipalComponents -R 0.95 -A 5*

centerData:- Center (rather than standardize) the data. PCA will be computed from the covariance (rather than correlation) matrix
transformBackToOriginal:- Transform through the PC space and back to the original space. By default set to false.
varianceCovered:- Retain enough PC attributes to account for this proportion of variance. Default value is 0.95.
maximumAttributeNames:- The maximum number of attributes to include in transformed attribute names. Default is 5.

Ranker: *weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1*

numToSelect:- Specify the number of attributes to retain. The default value (-1) indicates that all attributes are to be retained.
threshold:- Set threshold by which attributes can be discarded.
startSet:- Specify a set of attributes to ignore and is empty by default.

Linear Regression: *weka.classifiers.functions.LinearRegression -S 1 -R 1.0E-8 -additional-stats -num-decimal-places 4*

minimal:- Discards the dataset header, means and standard deviation to conserve memory. Default value is FALSE.
ridge:- WEKA uses a ridge regularization method to minimize the square of the absolute sum of the learned coefficients, which restricts any specific coefficient from getting too high. Default value is 1.0E-8
attributeSelectionMethod:- Set the method used to select attributes for use in the linear regression. For this study this parameter is set to" no attribute selection"
eliminateColinearAttributes:- removed highly correlated attributes to improve the performance. Default value is TRUE.

Multilayer Perceptron: *weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a -R*

seed:- Value used to initialize the weights of the connections between nodes, and also for shuffling the training data
momentum:- Momentum applied to the weights during updating. Default is 0.2
hiddenLayers:- This defines the hidden layers of the neural network. It is represented as numbers. it also has option for wild card values such as a which is the average of attributes count and number of classes . Default is "a".
validationThreshold :- Used to terminate validation testing. The value here dictates how many times in a row the validation set error can get worse before training is terminated. Default value is 20.
decay:- This indicate the decay in learning rate.
normalizeAttributes:- this property is set to normalize the numeric attributes.
normalizeNumericClass :- set to normalize the class if it's numeric

trainingTime:-The number of epochs to train through.
learningRate:- The amount the weights are updated. Default is 0.3.

Random Forest : *weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1*

seed:- The random number seed to be used.
numExecutionSlots:- The number of threads to use for constructing the ensemble. Default is 1.
bagSizePercent:- Size of each bag, as a percentage of the training set size. Default is 100.
numIterations:- The number of iterations to be performed. Default is 100.
maxDepth:- The maximum depth of the tree, 0 for unlimited.
numFeatures:- Sets the number of randomly chosen attributes.

k-nearest neighbor(kNN): *weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""*

KNN:- The number of neighbours to use.
distanceWeighting:- Gets the distance weighting method used.
nearestNeighbourSearchAlgorithm -- The nearest neighbor search algorithm to use. Default is weka.core.neighboursearch.LinearNNSearch.
windowSize:- Gets the maximum number of instances allowed in the training pool. A value of 0 signifies no limit to the number of training instances.
meanSquared:- Whether the mean squared error is used rather than mean absolute error when doing cross-validation for regression problems.

Gradient Boosting Machines (GBM): *weka.classifiers.meta.AdditiveRegression -S 0.2 -A -I 10 -W weka.classifiers.trees.RandomTree -- -K 0 -M 1.0 -V 0.001 -S 1*

classifier -- The base classifier to be used. Default is DecisionStump.
shrinkage:- Shrinkage rate. Smaller values help prevent overfitting and have a smoothing effect but causes an increase in learning time. Default is 1 which means no shrinkage. In this study a shrinkage of 0.3 is used.
minimizeAbsoluteError :- Minimize absolute error instead of squared error.

Support Vector Machines (SVM) for regression: *weka.classifiers.functions.SMOreg-C 1.0 -N 0 -I "weka.classifiers.functions.supportVector.RegSMOImproved -T 0.001 -V -P 1.0E-12 -L 0.001 -W 1" -K "weka.classifiers.functions.supportVector.RBFKernel -C 250007 -G 0.01"*

SMOreg implements the support vector machine for regression. The algorithm is selected by setting the RegOptimizer.
c:- The complexity parameter C. A value of 0 allows no violations of the margin, Default value is 1.0
kernel:- The kernel to use. This is the key parameter for SVM and default is Polynomial Kernel. In this study Radial Basis Function(RBF) kernel is used and RBF is capable of learning closed polygons and complex shapes to fit the training data.
filterType:- selects the data transformation if needed.
RegOptimizer:- The learning algorithm.

C4.5: *weka.classifiers.trees.J48 -U -M 2*

seed:- seed is the random number generator when pruning is used.
unpruned:- used to specify if pruning is done.
confidenceFactor:- The confidence factor used for pruning. confidenceFactor value is directly proportional to pruning strength.
numFolds:- Determines the amount of data used for reduced-error pruning. One fold is used for pruning, the rest for growing the tree.
reducedErrorPruning :- Whether reduced-error pruning is used instead of C.4.5 pruning.
doNotMakeSplitPointActualValue:- If true, the split point is not relocated to an actual data value.
subtreeRaising:- used to select if subtree raising operation to be performed when pruning.
binarySplits:- Whether to use binary splits on nominal attributes when building the trees.

Support Vector Machines (SVM) for classification: *weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 2 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.RBFKernel -C 250007 -G 0.01" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"*

RBF Kernel: $K(x,y) = exp(-0.01*(x-y)^2)$

Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.

c:- The complexity parameter C. A value of 0 allows no violations of the margin, Default value is 1.0

kernel:- The kernel to use. This is the key parameter for SVM and default is Polynomial Kernel. In this study Radial Basis Function(RBF) kernel is used and RBF is capable of learning closed polygons and complex shapes to fit the training data.

filterType:- selects the data transformation if needed.

calibrator :- The calibration method to use. By default data is normalized but in this thesis no filter type is selected.

Naive Bayes: *weka.classifiers.bayes.NaiveBayes*

useKernelEstimator:- Use a kernel estimator for numeric attributes rather than a normal distribution.

displayModelInOldFormat:- Use old format for model output. The old format is better when there are many class values. The new format is better when there are fewer classes and many attributes.

useSupervisedDiscretization:- Use supervised discretization to convert numeric attributes to nominal ones. By default is set to false.