2017

Physical Human Activity Recognition Using Machine Learning Algorithms

Haritha Vellampalli
Technological University Dublin

Physical Human Activity Recognition Using Machine Learning Algorithms

Haritha Vellampalli

A dissertation submitted in partial fulfillment of the requirements of Dublin Institute of Technology for the degree of M.Sc. in Computing (Data Analytics)

September 2017

Declaration

I certify that this dissertation which I now submit for examination for the award of M.Sc. in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: 

Haritha Vellampalli

Date: November 10, 2017

Abstract

With the rise in ubiquitous computing, the desire to make everyday lives smarter and easier with technology is on the increase. Human activity recognition (HAR) is the outcome of a similar motive. HAR enables a wide range of pervasive computing applications by recognizing the activity performed by a user. In order to contribute to the multi facet applications that HAR is capable to offer, predicting the right activity is of utmost importance. Simplest of the issues as the use of incorrect data manipulation or utilizing a wrong algorithm to perform prediction can hinder the performance of a HAR system.

This study is designed to perform HAR by using two dimensionality reduction techniques followed by five different supervised machine learning algorithms as an aim to receive better predictive accuracy over the existing benchmark research. Correlation analysis (CA) and Principal component analysis (PCA) are used for feature reduction which resulted in 173 and 100 features respectively. Decision Tree, K Nearest Neighbor, Naive Bayes, Multinomial Logistic Regression and Artificial Neural Network algorithms were used to perform the classification task. The repeated random sub-sampling cross validation technique was used to perform the evaluation followed by a Wilcoxon signed rank test to evaluate the significance of the tests.

The study resulted in ANN performing the best classification by achieving 97% of accuracy using the CA as feature reduction technique. The KNN and LR also provided satisfactory results and have received predictive results greater than the benchmark test. However, the decision tree and Naive bayes algorithms didn't prove efficient.

Keywords: Human Activity Reduction, Supervised Machine Learning, Wearable sensors

Contents

List of Figures

List of Tables

List of Acronyms

ADL	Activities of Daily Living
ANN	Artificial Neural Network
AUC	Area Under the ROC Curve
CA	Correlation Analysis
CRISP-DM	Cross Industry Standard Process for Data Mining
DT	Decision Tree
FN	False Negative
FP	False Positive
HAR	Human Activity Recognition
HMM	Hidden Markov Model
KNN	K Nearest Neighbor
LR	Multinomial Logistic Regression
NB	Naive Bayes
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic graph
RRSS	Repeated Random Sub-Sampling cross validation
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

Chapter 1

Introduction

1.1 Background

Activity recognition is the task of recognizing the current physical action performed by one or more users from a set of observations recorded during the user activity in the context of the definitive environment. Recent times have seen the theory of Human Activity Recognition (HAR) catering to multiple challenging applications built on the increase in ubiquitous, wearable and persuasive computing. Human Activity recognition has become an important technology that is changing the landscape of people's daily routine contributing to a wide range of applications as assistive technology, health and fitness tracking, elder care and automated surveillance to name a few. Additionally, the research in activity recognition has been so rapid and advanced that it is starting to cater applications that go beyond the activity recognition. However, as the field is rich in practical applications, the challenges emerging for activity recognition are multifold.

The typical workflow of a human activity recognition task deals with data acquisition from a wearable sensor or an external device as cameras. This data is then

processed to obtain a cleaner and transformed data suitable for further processing. The data is explored to understand its nature and the type of processing that can be applied in the next stages. The design of the later stages could vary vastly on the application and its domain. But typically, the data is engineered to make it more appropriate by extracting more useful features from it. Furthermore, the data is segmented on basis of the evaluation that needs to be performed. Finally, a predictive model is created to identify the activities performed by the user and is evaluated for its performance. This detected activity can be repurposed for various applications as detecting change in the user activity, assistance in regard to the detected activity etc.

1.2 Research Problem

In the seemingly simple approach of performing HAR, there are many issues and challenges that are encountered as selecting the right tools and techniques for gathering, storing and manipulating the data. Picking the right algorithm to perform predictions is of utmost importance as there is a necessity to capture the inter-class variability and the intra class similarity. Typical resource constraints as processing power, availability of time and sufficient storage are difficult to handle and form a huge hurdle. There is also a necessity to have a trade-off between system latency, accuracy and processing power.

The initial stages of HAR deal with a different set of issues altogether. Picking the right sensor or combination of sensors, selecting the attributes and metrics to be measured, placing the sensor at the right location are all crucial in their own way. Furthermore, all these must be done by considering the user privacy and usability in context. The process must not be obtrusive for the user and must adapt to the user's behavior and their environment as entire process can be highly sensitive to the participation and the interaction with the user.

The research question that is planned to be addressed in the current study can be concisely stated as follows –

> *To what extent can supervised machine learning algorithms significantly enhance the recognition of physical human activity with inertial sensor data when compared to SVM base models?*

*Algorithms : K-Nearest Neighbors, Decision Tree, Naive Bayes, Multinomial Logistic Regression and Artificial Neural Network.

1.3 Research Objectives

A possible solution in overcoming these challenges could be by reviewing and analyzing the existing research and picking the sensor and a location with proven capabilities. This eliminates the risk of data inaccuracy and justifies the effort put in engineering the captured data. In addition, considering the computing capabilities at hand, performing dimensionality reduction is mandatory. Performing feature reduction is a better option over minimizing the captured activity data, as more amount of user activity can help in create and train a better model. Correlation analysis and principal component analysis are two of the most popular and effective feature reduction techniques discovered through the literature review. Correlation analysis identifies features that highly correlate with each other, there is no use for such features which describe the same aspect of the targeted activity value, in fact such features confuse the algorithm and prevent it in performing to the extent of its abilities. Getting rid of such redundant features could add great value. Similarly, principal component analysis also performs feature reduction but by generating new features that are a linear combination of the existing features. But with smaller number of features the PCA captures significantly more information about the target feature.

All these data manipulation tasks are a waste of effort if there is no scientific approach that can exploit these. However, picking the right algorithm to perform the task is complicated. A possible approach could be picking multiple algorithms, each having a different mechanism of action to tackle the task. Evaluating and comparing each of these models can help selecting the right approach for performing activity detection. Literature review identified four different mechanisms in which supervised machine learning can be performed namely, Information based, Similarity based, Probability based and Error based learning. Algorithms from each of these sets of techniques can be utilized to have a robust and significant solution.

So primary objective of the research is to determine the algorithm that can perform Human Activity Recognition with classification accuracy higher than the current state of the art technique. In order to achieve this goal a number of tasks must be accomplished. A list of these tasks is stated below –

1. Extensive study of the existing literature on experiments and research performed under Human Activity Recognition to identify research gaps.
2. Design a solution to perform Human Activity Recognition by detecting the inter-class differences and the intra-class similarity within the activities.
3. Implement the solution reinforced in the design and induce models to perform Human Activity Recognition to obtain the targeted accuracy.
4. Evaluate the performance of the induced models.
5. Place the findings in the field of study.

1.4 Research Methodology

As current study aims to compete with an existing HAR system, this work will primarily perform a secondary research using the existing data from the state of the art

experiment and no other data will be generated for the scope of this project. Additionally, secondary research of reviewing existing literature on the topic of HAR and its applications will be performed under the task 1.

The core experiment to be performed will be based on the quantitative objective of picking the best classifier measured through a predictive classification accuracy value. The research involves the solution to utilize mathematical modeling in creating the modified datasets and in performing an evaluation using statistical techniques. So the study can be stated as of empirical research form.

The current study involves observing and analyzing the existing literature to form hypothesis that can be proved through the experiment. The results of the hypothesis tests performed in the experiment can be transformed into theories that can be generalized to specific contexts. This establishes that the current study is inductive in approach.

1.5 Scope & Limitations

Various studies in HAR have discovered an increased predictive accuracy results when multiple datasets are correctly integrated and used for training the model (Mannini et al. (2013); Sucerquia et al. (2017)). But this study will utilize only a single dataset from an existing state of the art technique.

The benchmark experiment utilized for the study provides only the derived components of the actual data acquired from the sensors. The study could have greatly benefited if the raw signals from the sensors were provided but only the derived signals were accessible.

There are plenty of methods to perform dimensionality reduction but two of the most popular techniques from the literature review were selected due to the time and

computing constraints of the experiment. Similarly, multiple machine learning models and plenty of their parameters were discovered from the literature that can be utilized to gain better insights from the data, however, the study had to limit the number of algorithms and parameters to manipulate, so one algorithm from each of the family is employed with limited parameter tuning.

Also, two kinds of validation techniques were planned to be performed in the experiment but had to be limited to one as the results from one test were significant enough to evaluate the hypothesis.

1.6 Organization of Dissertation

The rest of the document will be structured as follows –

- Chapter 2 provides an overview of the existing literature on HAR. It outlines the different sensor technologies existing and the effect of its placement on the appropriate location of the body has on the accuracy of the study. The chapter also examines the different supervised machine learning techniques that are used to perform HAR. The theory, applications and example studies using the algorithms are discussed here. Various evaluation metrics are also studied to identify the method that is most suitable for the experiment.

- Chapter 3 outlines the basic design of the study. The structure of the chapter is based on the CRISP-DM methodology, so initially the business understanding phase is designed followed by data understanding phase. Data preparation to be performed is reviewed next, followed by the modeling stage. The chosen evaluation metric and the evaluation strategy are designed in the end. The chapter concludes by stating the strengths and limitations of the developed design.

- Chapter 4 details the practical implementation of the experiment. Each of the

utilized technique and their corresponding results obtained are illustrated here. This chapter is also structured according to the CRISP-DM methodology.

- Chapter 5 performs a critical evaluation of the experiment and the results obtained. Comparison of each algorithm and feature engineering technique is performed to understand the outcome. The hypothesis of the study are also evaluated here. This chapter concludes by stating the strengths and limitations of the implementation.

- Chapter 6 concludes the study by reiterating the research question and the problem definition. It also states the workflow of the entire process and additionally states the scope and limitations of the study. The chapter concludes by specifying areas of potential future work.

Chapter 2

Literature Review & Related Work

This chapter gives a detailed review of the relevant literature about human activity recognition and the state-of-the-art techniques involved in its detection. The appropriate sensor technologies used in similar studies are also examined along with their practical application areas. Additionally, the ideal location of the sensor on the human body is also discussed. So this chapter provides the groundwork for selecting the technology and the specific location of affixing the sensor on the body of the user before performing the actions. Furthermore, this section also discusses the technologies used for the detecting of the physical activity; firstly, the theory of detection using machine learning is discussed along with its applications; it is then followed by discussing the family of machine learning techniques that can be applied. A popular machine learning technique from each of these set of family of techniques is studied with respect to their application in similar human activity recognition papers. The chapter will conclude by highlighting the state of the art technologies in terms of gathering the data and provides justification to employ a set of techniques in order to perform better at the detection task. A graphical layout of the chapter is provided by 2.1. The aim of this section is to perform a critical analysis of the existing research and approaches so as to highlight the gap in the current body of work. The gap envisioned in this section will help establish the research question for the study.

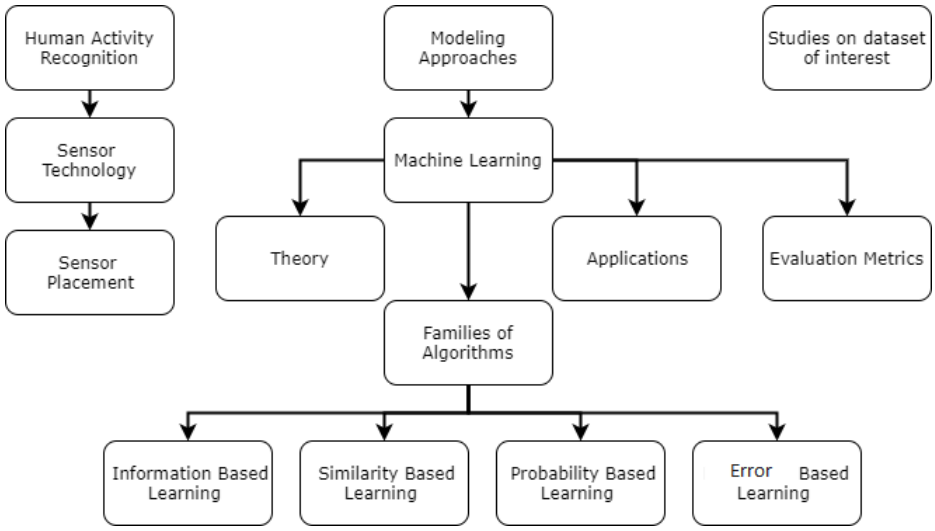

Figure 2.1: Chapter Layout

2.1 Human Activity Recognition (HAR)

Human activity recognition has gained much importance in the past decade due to its numerous applications in human centric applications as in the field of medical, security and also military (D. Lara & Labrador, 2013). In recent times, due to the increase of wearable tech devices, the task of human activity recognition has gained much more gravitas[1]. An important goal of the HAR in the current scenario is to identify the actions of the user in order to assist them with their tasks with the help of computing systems Abowd et al. (1998). Computer vision research has been contributing a lot in this aspect of the study. Human activity recognition here mainly refers to physical human activity and load, as opposite to cognitive, mental activities and workload which are part of a different, wider research field (Rizzo et al., 2016; Longo, 2011, 2012, 2015, 2016; Moustafa et al., 2017). The initial research on HAR involved detecting gestures and activities from still images and videos in restricted environments and under constrained settings Turaga et al. (2008); Mitra & Acharya

[1] https://www.forbes.com/sites/paullamkin/2016/02/17/wearable-tech-market-to-be-worth-34-billion-by-2020/#271d9cf43cb5

(2007). A significant number of domains have been discovered to benefit due to HAR as in the case of Activities of Daily Living (ADL's) by Katz et al. (1970), which was one of the initial researchers performed as an application of activity recognition, which further boosted the research as by Bao & Intille (2004); Ravi et al. (2005); Logan et al. (2007); Tapia et al. (2004). The traditional medical procedures were challenged by introducing the HAR to support patients' daily activity monitoring especially for patients with chronic impairments or other medical diagnosis or even for rehabilitation (Starner et al., 1997; J. Chen et al., 2006; Oliver & Flores-Mangas, 2007; Bachlin et al., 2009; Tessendorf et al., 2011). HAR also provided great results for other areas of lesser severity as the entertainment and sports category (Kunze et al., 2006; Minnen et al., 2006; Ladha et al., 2013), the industrial and operations sector (Maurtua et al., 2007; Stiefmeier et al., 2008). HAR was further explored to cater naive human activities as transportation routines (Krumm & Horvitz, 2006), brushing teeth (Lester et al., 2006) and medication intake (Wan, 1999; De Oliveira et al., 2010). One of the most recent and popular usages of human activity recognition was for gaming consoles as Microsoft Kinect where body gestures and movements are recognized to provide an upgraded gaming experience (Shotton et al., 2013). Human Activity Recognition

2.2 Sensor Technology for Gathering Data

Activity recognition can be performed for a single user or for multiple users. Multiple user recognition can be performed to identify an individual user and track their actions. This can be performed for surveillance and monitoring purposes using video camera footage. However, for a single user activity recognition process, there could be multiple ways to gather data. Owing to the advances in sensor technology, a popular method of HAR is identified to be using Inertial Sensors. Most inertial sensors have become portable and compact to be connected to the human body. Flexible form factors and battery levels which are designed for longer recording and monitoring purposes along with computing and dynamically consistent interaction, make them simpler and

more suitable to use (Florentino-Liano et al., 2012; Bulling et al., 2014). The first use of sensors for activity recognition was in the context of smart homes by 'Neural Network House' along with other apps that help create adaptable systems for better user experience of their smart home (Mozer, 1998; Leonhardt & Magee, 1998; Golding & Lesh, 1999; Ward et al., 1997). The inertial sensors with integrated gyroscopes and accelerometers have been utilized for various purposes as medical diagnosis and treatment (Powell et al., 2007), Tele-Rehabilitation (Winters et al., 2003), Fall detection (Wu & Xue, 2008) and human movement monitoring (Sabatini et al., 2005).

A combination of different sensors at various locations have been utilized previously to produce varying results. One of the most popular sensors used quite frequently for similar studies involving repetitive actions is an accelerometer. An accelerometer is an electromechanical device which is used to measure static and dynamic acceleration forces. For instance, the angle of tilt or inclination of the device can be calculated by measuring the gravity acceleration[2]. The accelerometer was used in multiple studies as a motion sensor, yielding good results for their area of application (Bao & Intille, 2004; Mi-hee et al., 2009; Khan et al., 2008).

Sensors as image and audio based have also been utilized for applications as image tagging and activity detection using noise levels for instance if the noise is less, the user could have been asleep etc (Qin et al., 2014; Bieber et al., 2011). Global positioning system (GPS) sensors are quite widely used as well. The GPS sensor was used to detect the user activity through location based signals across single and multiple users and was also used to track any abnormal activities by the users (Patterson et al., 2003; Ashbrook & Starner, 2003; Liao et al., 2007). Medical applications which involve detecting the patients' critical medical information as heart or respiration rate or other important metrics utilize a series of biosensors. Sung et al. (2004) detect body temperature using an arm and chest accelerometers and polar heart rate receiver sensors to detect emergencies due to weather conditions as hypothermia for soldiers living in severe weather conditions. Biosensors are also used to create smart clothing which

[2] "https://www.dimensionengineering.com/info/accelerometers"

can be used to detect body postures (Harms et al., 2008). Wren & Tapia (2006) utilize infrared sensors which can detect temperature, which was further utilized to differentiate different levels of activities producing low and high levels of heat and thus enabling to be differentiated.

There are also combinations of sensors utilized together for better detection of the signals. The accelerometer was combined with the psychological sensor to detect signals as skin temperature and energy expenditure (S.-I. Yang & Cho, 2008). Apart from these various types of sensors, the simplest and one of the efficient is the accelerometer in a combination with a gyroscope, which is a device used to detect angular velocity (Lazzarini, 2007). Due to development of mobile phone technology, the smart phones have built in accelerometers and gyroscopes which make the study of activity detection furthermore simplified (Brezmes et al., 2009; Oh et al., 2010). Anguita et al. (2013); Kwapisz et al. (2011) utilize the sensors in the mobile phone to perform the task of human activity recognition .

2.2.0.1 Placement of the Wearable Sensor

Cleland et al. (2013) investigate the importance of sensor and the optimal placement of it on the body of the user. Their study demonstrates that the acceleration signal values gradually increase in magnitude as the placement of the sensor moves from head to feet. So it is evident that the location has direct impact on the results obtained for the HAR process. As shown in figure 2.2, the sensor can be placed at multiple locations of the body (Attal et al., 2015).

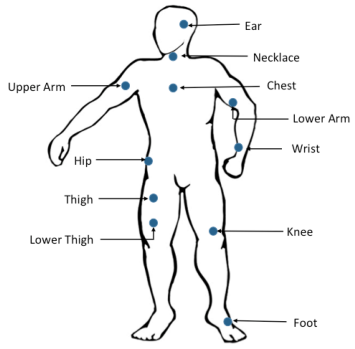

Figure 2.2: Illustration of Wearable Sensor Placement

Multiple studies have utilized the sensor placed at various body parts of the user and have received various ranges of accuracies. Parkka et al. (2006) have studied HAR by placing the sensors at the wrist and chest and performed various activities for the duration of 2 hours. The best results out of the three classifiers yielded were of 83% accuracy. Investigating multiple other studies exhibited the best results from the study by Yeoh et al. (2008) which performed the experiment using three sensors, one mounted at the waist and two more attached to the centre of the thighs. Their experiment resulted in an overall accuracy of 100% by detecting four important tasks extremely well. However, observing the studies that utilized only a single sensor at one location of the body, a sensor at waist or on the lower back have been yielding good results. Mathie et al. (2004); Gupta & Dallas (2014) have performed HAR task using a single sensor placed at the Waist and have received a classification accuracy of 98% and Bonomi et al. (2009) have received an accuracy of 93% while detecting for similar activities. Hence, placement of sensor on the waist can be seen as an ideal position as it is proven and additionally it is also closer to the centre of mass of the body (C.-C. Yang & Hsu, 2010).

2.3 Modeling Approaches for HAR

2.3.1 Machine Learning

2.3.1.1 Theory

The past two decades have seen a rise in Machine learning becoming a crucial component in Information Technology. With continual increase in data availability, it is evident that more smarter data analysis will be an integral part of the technological processes at every phase of life.

Machine learning as defined by Kelleher et al. (2015) is "an automated process that extracts patterns from the data". It can also be defined as "a method of data analysis that automates analytical model building, using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look"[3].

The goal of machine learning is to enable the machine to learn the system automatically, without any human interventions on basis of the designed algorithms. These algorithms maybe originated from many fields as mathematics, theoretical computer science etc. It is important to note that the algorithms devised for this task must serve the purpose whilst being efficient, which comprises of both time and space efficiency. For the context of learning, the amount of data required by the algorithm is of primary importance and must be utilized to the maximum extent. The algorithms must also be made flexible to enable generalization to various applications. However, the primary motive of machine learning is to harness the predictive capabilities of the machine and hence the predictive accuracies must be as high as possible with minimal error rate.

[3] "https://www.sas.com/en_us/insights/analytics/machine-learning.html"

2.3.1.2 Applications

There are huge number of applications of machine learning and its usages are ever increasing. Some of the most popular applications are (Alpaydin, 2014)–

- Creation of a good search engine requires the engine to process the search query, identify pages having the information and sort them according to prescribed algorithm. Machine learning has increasingly been utilized to automate the process of web page ranking and determining the best results for a given query.
- Recommendation engines are another area which has seen rapid development as they help entice the users with newer products or services. Ecommerce companies as Amazon and eBay utilize this system to analyze past purchases and viewing options to predict and enable future viewing and purchases. Netflix, which is a video rental store, also utilizes this mechanism. This application can also be termed as Collaborative filtering.
- Another application that is not quite defined is the text translation problem. This problem is quite tricky as the machine needs to model the grammar and the language of the document. There is also huge research in this area of application.
- Face recognition is an example of machine learning revolutionizing the current existing applications. The machine breaks down the face of the user into smaller parts as pixels to detect and classify the faces. This system is currently being utilized in various security applications and also social networking websites.
- Other applications that are utilizing machine learning are, speech recognition, handwriting recognition, named entity recognition, failure and fault detection in several industrial equipment, gaming consoles etc.

2.3.1.3 Family of Machine Learning Approaches

The study of Machine learning can be divided into multiple sectors on basis of the types of work to be done, data dealt with or working procedure. Kelleher et al. (2015) have devised one such method of segregating the algorithms on the basis of their ability to learn from the data. It must be noted that only supervised algorithms under each of the family will be dealt with.

- Information Based Learning –

  The idea behind this approach of machine learning is to utilize the intuitiveness of the data in deriving a model to perform machine learning. Information gain and Gini Impurity are the measures on which the learning process is performed by determining the features that best describe the entire data.

  Decision trees are machine learning algorithms that follow information based learning approach. A decision tree algorithm approximates the target function by creating a solution that can be represented by a tree leading to the target feature. Decision trees are widely utilized as they can be represented in the form of if-else statements for better readability and understanding. The algorithm also requires minimal data preparation or feature engineering which can help save time and effort. Hence, decision trees have been highly utilized in performing activity recognition. Fan et al. (2013) performed activity recognition using decision tree algorithms by constructing behavior and position vectors of users performing 5 different activities. The study reported high classification accuracy and less time consumption.

- Similarity Based Learning –

  The concept of similarity based learning is to observe the previously existing data in order to predict the future or unknown data. This technique utilizes the measure of similarity which denotes how similar or related multiple data points are to each other. This algorithm is described by the techniques used to process the pair wise data, their relationships and the assumptions in their relationships. The most fre-

quent and natural algorithm of similarity based learning is the K-nearest neighbor (K-NN) algorithm. This algorithm assumes that if two data points, a_i and a_j are similar to each other, the corresponding target or outcome classes, x_i and x_j are also similar (Hu et al., 2015).

Kaghyan & Sarukhanyan (2013) have studied the accelerometer of an android mobile phone and applied the K- Nearest neighbor algorithm to predict the activity of a single user and have received satisfactory results. Paul & George (2015) created a model called the Clustered KNN which is an improved KNN algorithm to detect four activities performed by four users and achieved appreciable results by utilizing limited memory and a restricted training data.

- Probability Based Learning –

  The theory of probability based learning is to utilize the estimates of the likelihood of a data point in determining the target value of the point. The Bayesian algorithm assumes an underlying probabilistic model and captures the uncertainty of the model by calculating the probabilities of the target. The Bayesian algorithms are highly scalable and are hence widely used over plenty of applications.

  Sarkar et al. (2010) have studied and compared the Naive Bayes algorithm to the Hidden Markov Model and the Conditional Random field model. They have demonstrated that previous studies have shown the Markov model outdoing the Naive Bayes algorithm. However, it stated that parameter estimation plays a huge role in a classifier and performed two types of smoothing techniques to adjust the maximum likelihood of the classifier. Their experiment offered significant improvement in the classification accuracies by the Bayes algorithm. Ravi et al. (2005) have performed a similar activity recognition system which aims to classify a set of eight different activities, at four different environment settings. Their study determined Naive bayes to outperform all the other single classifiers in two of their settings.

- Error Based Learning –

  The concept of error based learning is to initialize a parameterized model with a

set of random parameters in order to identify the parameters which correspond to the minimized error value for the specified training instances. The model and its parameters are adjusted based on the error value to achieve high accuracy values. There are multiple error metrics that can be utilized under Error based learning, one of the most popular metric is the Sum of squared errors.

Logistic regression, Artificial neural net and Support vector machine are few of the most popular error based machine learning algorithms. The logistic regression is the odds of the target feature taking a specific value modeled on the basis of a combination of values taken by the feature vectors. A binary logistic regression model estimates the likelihood that the target is present given a set of feature variables. Y. Chen et al. (2016) performed activity recognition using a multinomial logistic regression with Bayesian regularization and have received a high accuracy of 93% over other algorithms discussed in their study. Artificial neural nets are widely used for HAR. Multiple studies have observed exceptional results in detecting user activities using ANN. Oniga & Suto (2014) stated that with the right topology and parameters, the ANN can detect the most complex activities. Multiclass SVM is another popular technique as it can model data in different non linear distributions owing to its kernel function and can also be prevented from being sensitive to outliers and hence was used in multiple studies to achieve good accuracy rates (Anguita et al., 2012; D. Lara & Labrador, 2013; Ravi et al., 2005).

In addition, the survey by bin Abdullah et al. (2012) demonstrated that the popularity and efficiency of various classification algorithms that are commonly used in detecting ADL using embedded sensors from a smart phone device. The most commonly used algorithms that have been deemed efficient as identified by this study were Hidden Markov Models, Multiclass Logistic Regression, Decision Tree, Naive Bayes, Support Vector Machine, Artificial Neural Network, K - Nearest Neighbor, Gaussian mixture models and multiple other ensemble and modified algorithms.

2.4 Related Work on the dataset of interest

The most relevant work for the objectives of the thesis is the work by Anguita et al. (2013). Their study dealt with creating an activity database by recording Activities of Daily Life (ADL) of 30 users. They have utilized the inertial sensors on a smart phone device to record the activities. They have employed a multiclass Support vector machine algorithm to detect these activities. They have received improved classification accuracy in this study compared to their previous paper which was an SVM classifier with additional usage of fixed point arithmetic algorithm which as stated in the research would reduce additional computational cost (Anguita et al., 2012).

S.No.	Technique	Feature Engineering	Training Method	Validation Metric
1	Hidden Markov model (HMM) - 1,2,3 staged & Random Forest Importance Measures(RFIM)	A	A	
2	Three-stage continuous hidden Markov model (TSCHMM)	sparse locality preserving projections (SpLPP)	A	A
3	Multi class SVM	-	A	A
4	(MC-HF-SVM), SVM		A	A
5	Mel Frequency Cepstral Coefficients (MFCC), Linear Predictive Coding (LPC), LPC-Derived or Linear Predictive Cepstral Coefficients (LPCC) and Perceptual Linear Predictive Cepstral Coefficients (PLPCC)	Relative Spectra (RASTA) Filters	A	A
6	Group-based Context-aware Method (GCM)	K-means clustering algorithm	A	A
7	Two stage CHMM	PCA, Correlation, LDA, RF importance measure	A	A
8	SVM	Postural Transitions (PTs)	A	A

S.No.	Technique	Feature Engineering	Training Method	Validation Metric
9	NB, RF, KNN	Recursive Feature Elimination based on Linear Discrimination Analysis (RFELDA)	A	A
10	RKELM (Reduced Kernel Extreme Learning Machine)	Synthesized acceleration value filter	A	A
11	Naive Bayes, Decision Tree	Correlation + IB3	A	A
12	HMM (Activity Sequence Model)		B	B
13	Auto-encoder, denoising auto-encoder and PCA		A	A
14	convolutional neural networks			
15	General, User-Dependent, and Mixed models.	-	B	A
16	Gaussian Mixture Model-Universal Background Model (GMM-UBM)	MFCCs (Mel Frequency Cepstral Coefficients), PLP (Perceptual Lineal Prediction) and RASTA-PLP coefficients	B	C
17	HMM - Maximum Likelihood Estimation. Maximum Mutual Information Estimation	Maximum A Posteriori (MAP) approach	A	A

S.No.	Technique	Feature Engineering	Training Method	Validation Metric
18	Convolutional neural networks (convnets)		A	A
19	Importance score drive random forests	Gini information gain	A	A
20	Generalized Learning Vector Quantization	Kernel classification correlation matrix & Limited rank mapping matrix	C	A

Table 2.1: Literature Review of studies on the dataset of interest

The table 2.1 describes the various machine learning techniques applied, feature engineering performed and validation metrics utilized for the same dataset used in the current study. The notations for the training method are – A : Cross validation; B : K fold validation; C : Sampling and the notations for the validation metric are – A : Confusion Matrix; B : Activity Segmentation Error Rate (ASER); C : AUC and ROC curves.

A Hidden Markov Model (HMM) is the most popular algorithm used in activity recognition for the dataset considered in the thesis. Various studies performed one, two and three staged HMMs on the specified dataset of interest (Zhu & Qiu, 2016; Ronao & Cho, 2014; San-Segundo, Lorenzo-Trueba, et al., 2016; San-Segundo et al., 2017). Also the Gaussian mixture models which are slightly more complicated have been utilized to perform activity recognition using the dataset of interest. However, the algorithm couldn't manage to yield results as accurate as the others (San-Segundo, Cordoba, et al., 2016).

2.5 Model Evaluation Metrics

2.5.1 Performance Measures

Performance measures encompass the final steps in a machine learning project. They emphasize and evaluate the correctness, efficiency and usefulness of the design and the modeling process and additionally also provide reliability metrics on the entire procedure. There are multiple ways to evaluate the important criteria based on the project. The appropriate criteria and its measure must be chosen depending on the research question and also the ability and feasibility of the researcher and the study.

2.5.1.1 Confusion Matrix

Confusion matrix also termed as contingency table, provides a comprehensive overview by summarizing the classification results. It showcases the individual results for each of the classes by tabulating the predicted and actual classes. The image 2.3 shows a confusion matrix and its components.

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Figure 2.3: Confusion Matrix

TP – Indicates the true positives which is the number of positive records correctly predicted as positive by the model

TN – Indicates the true negatives which is the number of negative records correctly predicted as negative by the model

FP – Indicates the false positives which is the number of negative records incorrectly predicted as positive by the model

FN – Indicates the false negatives which is the number of positive records incorrectly predicted as negative by the model

Each of these measures have a significance of their own depending on the research question. Multiple standardized metrics can be defined from the confusion matrix –

- Accuracy – It is the total number of correct predictions proposed by the model which includes the positive and negative predictions.

$$\text{Accuracy} = \text{number of correct predictions/ total number of prediction}$$

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

- Recall or True Positive Rate – It is the proportion of positive classes identified correctly by the model.

$$\text{Recall} = \text{number of correct positive predictions / total number of positive cases}$$

$$= \frac{TP}{TP + FN}$$

- Precision – It is the fraction of positive cases correctly identified over all the positive cases predicted

$$\text{Precision} = \text{number of correct positive predictions / total number of positive predictions}$$

$$= \frac{TP}{TP + FP}$$

There are multiple other values that can be calculated from the confusion matrix as the F1 score and the error rate.

2.5.1.2 ROC Graph & AUC Value

Another important value that can be used for evaluation is the Area under the Receiver Operating Characteristic (ROC) graph (AUC) value. The ROC graph is the technique to visualize, organize and select classifiers based on their performance on a 2D space (Fawcett, 2003). The graph 2.4 represents the ROC graph with True positive rate plotted on the Y-axis and False positive rate plotted on the X-axis. So the closer the curve is to the Y-axis (True positive rate), the better the classifier is. The AUC value is the area under the ROC curve. An area closer to 1 represents the curve closer to Y-axis and represents a better classifier.

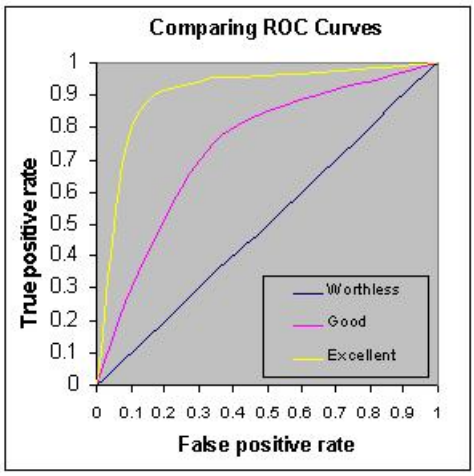

Figure 2.4: A sample Receiver Operating Characteristic (ROC) graph

2.5.2 Significance Tests

The above discussed metrics represent only the performance measures that can be used for each classifier. However, it is not sufficient to compare algorithms alone, the study must provide a statistical evidence of the result obtained from the evaluation. Statistical tests come into play in this regard. There are three tests that were quite popular in evaluating HAR and are analyzed further.

The McNemar's test primarily is used to compare the difference between proportions in two matched samples. However, Everitt (1992) utilized this test to compare a set of classifiers trained and test on the same datasets. This test doesn't measure the entire variation in the training data as a partial amount of data is used as a hold out set. The next test is the k- fold cross validation test, it is performed by creating k equal and disjoint sets of the dataset, the experiment is then run k times with each trial having a specific test set with all other sets merged to create the training set. This method permits each test set to be independent of each other which can be good technique to evaluate with, but also permits overlap of training dataset which might not result in a good variation. The last and interesting statistical test is the repeated random sub-sampling (RRSS) or the Monte Carlo cross validation. It performs a Monte Carlo repetitions of a randomly sampled data and the final results are obtained by aggregating the results for each repetition[4]. Colaprico et al. (2015) utilize this validation technique to detect tumors related to breast cancer with the help of biomarkers.

Dietterich (1998) studied each of these statistical tests for computational power and cost for running the learning algorithms. The study concluded that the cross validation tests were the most powerful followed by the McNemar's test. The study also stated for algorithms that may be executed only once, the McNemar's test is the only acceptable test, however, for tests that can be executed multiple times with computational and economic feasibility, the cross validation techniques are more suitable. Altun & Barshan (2010) studied HAR through data collected from miniature sensors. The study involved determining the best classification algorithm out of the chosen six. The experiment was evaluated using multiple techniques, out of which RRSS, K fold cross validation proved to be most effective.

[4] "https://uk.mathworks.com/discovery/cross-validation.html"

2.6 Summary

With growing industry of the wearable tech, activity recognition has become one of the popular studies with increasing number of practical applications and questions it can solve. The motivation behind the study is to utilize the best activity recognition techniques combined with suitable machine learning algorithms to obtain high predictive accuracies.

The No Free Lunch theorem states that there can never be one particular model that may be suitable for an application and that the assumptions generated for a single research question might not hold true to another question. Hence the theorem concludes that it is common for researchers to utilize multiple machine learning models to detect the one that best fits the data and domain (Wolpert, 2002). So the next direction in this research would be to evaluate the popular algorithms in activity recognition against the dataset of interest with suitable evaluation techniques. In conclusion, the research question to be answered by the current study is as follows:

> *To what extent can supervised machine learning algorithms significantly enhance the recognition of physical human activity with inertial sensor data when compared to SVM base model?*

Algorithms : K-Nearest Neighbors, Decision Tree, Naive Bayes, Multinomial Logistic Regression and Artificial Neural Network.

Chapter 3

Design & Methodology

This chapter details the plan and the design methodology for the current study. Several data mining methodologies were studied to identify a robust and well structured approach for the data mining project. The Cross Industry Standard Process for Data Mining (CRISP–DM), a well proven method is identified to conduct the current study, which at its core is a data mining project (Piatetsky, 2014). This methodology is an ideal sequence of events and the current chapter will deal with each of these phases as a separate section. However, the steps can always be traced back to previous stages to repeat or manipulate a step to better suit the following stage.

The first stage is the Business Understanding phase; this is where the business perspective of the project is understood. The desired outputs of this phase are the primary objectives of the study. The next phase is the Data understanding phase. This is the stage where the actual data utilized in the project is acquired. Each element of the data is inspected and described. The data is also further explored to produce any initial findings and their impact on the subsequent project stages. The quality of the data is also accessed at this stage. The third phase is the Data preparation stage. The unnecessary or repetitive data can be eliminated. Further cleaning of the data is also done along with development of new derived data elements of the existing

data. All the manipulated data elements are finally integrated to create a definitive dataset to be utilized in the later stages. The next step is the modeling stage. The modeling techniques analyzed from the Literature review section are induced. This stage also involves experimentation with various parameter settings aligning with the assumptions of the modeling technique. The next phase is one of the crucial stages, Evaluation phase. At this stage, each model is evaluated with the primary focus on the evaluation criteria and in contrast to the business objective. The models are then assessed of their merits and demerits.

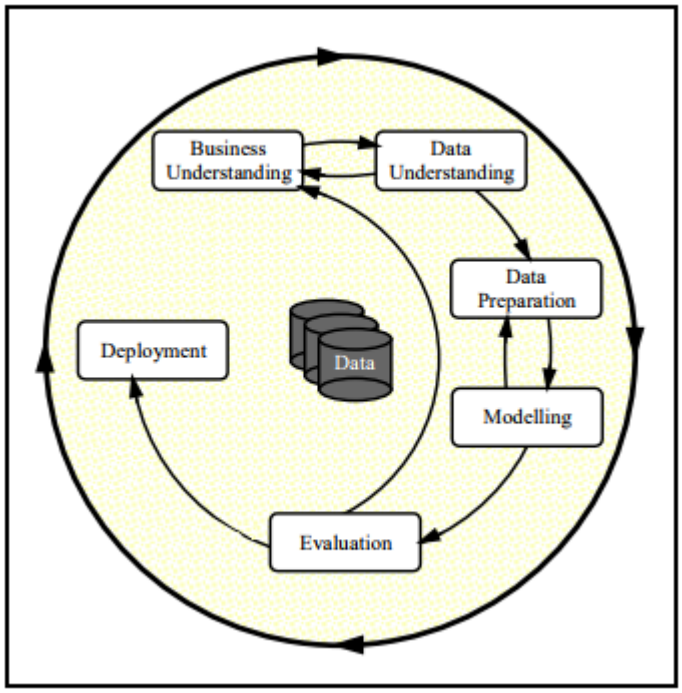

Figure 3.1: Phases of CRISP-DM Process Model for Data Mining (Wirth & Hipp, 2000)

3.1 Business Understanding

Human activity recognition, as seen from the previous chapter, has tremendous business value. The primary motive of this study is to enable patients, senior citizens or infants with immediate medical attention during a case of physical accident or

emergency. This can be ensured by detecting their current physical activity with the assistance of a wearable device connected to them. As it is a case of providing medical support, recognizing the right activity of the user is of utmost importance as it acts as a trigger to the chosen emergency action as notifying the guardian or the hospital. From an analytical perspective, in order to recognize the activities right, we must target onto high accuracy levels of prediction. To achieve the targeted accuracy levels, multiple machine learning models from various families of machine learning algorithms, as discussed in Chapter 2 will be employed.

The study assumes that with the increase in accuracy of the activity prediction, medical assistance can be improved proportionately.

An important constraint to note at this level could be the computing capabilities. As the current study is an academic project, the machine used is 64 - bit Intel i5 processor with 8GB of RAM. The coding will be done using R Language in R Studio of version 1.0.136. With no high computing power at availability, the experiment could suffer by the inability of using certain algorithms that might demand a more powerful machine.

Business Objective –

As the current study deals with enabling quick assistance to provide emergency services, they must be reported with high accuracy. This scenario requires ensuring high importance granted to false positives and false negatives, since an undetected change in activity could turn fatal to the user or an incorrect alarm could cause unnecessary panicking resulting in loss of valuable time, money and other resources. So the objective of the experiment is to implement machine learning models with high classification accuracy.

Business Success Criteria –

The solution must not only result in high classification accuracy, but also provide

evidence in proving that the experiment and results are significant and would always yield similar results when attempted to replicate the solution.

In view of the above objectives, assumptions and constraints, below hypothesis is utilized to address the research question –

> *Each supervised machine learning algorithm, modeled on the HAR dataset yields a different classification accuracy that is significantly greater than the benchmark SVM model, with a p value < 0.05*

Formally, the null hypothesis can be stated as –

$$\Big[\big(\text{Accuracy (Decision Tree)}\big) \neq \big(Accuracy(KNearestNeighbor)\big) \neq \big(Accuracy(NaiveBayes)\big) \neq \big(Accuracy(MultinomialLogisticRegression)\big) \neq \big(Accuracy(ArtificialNeuralNetwork)\big)\Big] > \big(\text{Accuracy (Base Support Vector Machine) }\big)$$

3.2 Data Understanding

The dataset used in the current study is generated at the International Workshop of Ambient Assisted Living (IWAAL) held in Spain in 2012. Anguita et al. (2012) designed an experiment by recording a set of six physical activities performed by a group of 30 volunteers. The tri-axial linear acceleration and the tri-axial angular velocity from the built-in accelerometer and gyroscope of a smart phone device are captured. These sensor signals were processed to remove noise and were sampled for every 2.56 seconds with a 50% overlap of the fixed width sliding window. The resulting signals had a combination of gravity and body motion components, and hence were passed into a low pass filter to obtain separated components with the gravitational force components cut off at the lower end of the filter. Time and frequency domain

vector of features were obtained from each of the window created. No cognitive activity, perception of load was gathered whatsoever (Longo, 2017).

Raw Signal	**Definition**
tBodyAcc-XYZ	Body acceleration in time
tGravityAcc-XYZ	Gravity acceleration in time
tBodyAccJerk-XYZ	Jerk in body acceleration in time
tBodyGyro-XYZ	Body gyroscope measure in time
tBodyGyroJerk-XYZ	Jerk in body gyroscope measure in time
tBodyAccMag	Magnitude of body acceleration in time
tGravityAccMag	Magnitude of gravity acceleration in time
tBodyAccJerkMag	Magnitude of jerk in body acceleration in time
tBodyGyroMag	Magnitude of body gyroscope measure in time
tBodyGyroJerkMag	Magnitude of jerk in body gyroscope measure in time
fBodyAcc-XYZ	Body acceleration in frequency
fBodyAccJerk-XYZ	Jerk in body acceleration in frequency
fBodyGyro-XYZ	Body gyroscope measure in frequency
fBodyAccMag	Magnitude of body acceleration in frequency
fBodyAccJerkMag	Magnitude of jerk in body acceleration in frequency
fBodyGyroMag	Magnitude of body gyroscope measure in frequency
fBodyGyroJerkMag	Magnitude of jerk in body gyroscope measure in frequency

Table 3.1: Description of raw signals from HAR experiment

Note: The 'XYZ' denotes the three axis directions X, Y, Z for each of the tri-axial signals; 't' indicates time domain variables and 'f' denotes frequency domain variables.

The features extracted from the experiment have been derived in an elaborate process and are tabulated in 3.1. The initially obtained raw signals, tAcc-XYZ and tGyro-XYZ were the tri-axial signals obtained from the accelerometer and the gyroscope, the total acceleration was further split into tBodyAcc-XYZ and tGravityAcc-XYZ. The Jerk signals, which are the rate of change in acceleration over time are derived from the raw signals and are denoted as tBodyAccJerk-XYZ and tBodyGyroJerk-XYZ. Euclidean norm is performed to calculate the magnitude of each of the signals further resulting in components as tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag, tBodyGyroJerkMag. Additionally, a Fast Fourier Transform was applied to produce the features as fBodyAcc-XYZ, fBodyAccJerk-XYZ, fBodyGyro-XYZ, fBodyAccJerkMag, fBodyGyroMag, fBodyGyroJerkMag. The table 3.2 lists the set of descriptive variables calculated for each of the above raw feature.

Descriptive	**Definition**
mean()	Average value
std()	Standard deviation
mad()	Median absolute deviation
max()	Maximum value
min()	Minimum value
sma()	Signal magnitude area
energy()	Energy value
iqr()	Interquartile range
entropy()	Signal entropy value
arCoeff()	Autoregression coefficient
correlation()	Correlation coefficient
maxInds()	Index of the largest magnitude frequency component
meanFreq()	Weighted average of the frequency component
skewness()	Skewness of the frequency domain signal
kurtosis()	Kurtosis of the frequency domain signal
bandsEnergy()	Energy of the frequency within the FFT of each window
angle()	Angle between the vectors

Table 3.2: Description of derived variables from raw signals

Note: The angle is calculated only for gravityMean, tBodyAccMean, tBodyAccJerk-Mean, tBodyGyroMean, tBodyGyroJerkMean vectors.

Additionally, an identifier variable, 'Subject', describing the user who carried out the particular activity is provided. It is a 30 factor categorical variable with labels from 1 to 30 each representing a volunteer carrying out the experiment. Finally, the feature describing the physical activity performed by the users during which the signals are collected is recorded as 'Activity'. It is a 6 factor categorical feature.The activities performed and recorded are tabulated in 3.3.

Activty ID
Walking
Walking_Upstairs
Walking_Downstairs
Sitting
Standing
Laying

Table 3.3: Activity list: Classes of target feature

3.2.1 Data Exploration

The first step in exploring the data is to view the dimensions of the dataset and to verify with the data source to ensure the data is intact and have the right number of dimensions. After validating the data integrity, each of these features has to be understood in order to proceed to the next stages of the experiment. A simplest way to perform this, is to evaluate the structure of the entire dataset. It is important to review with the structure of data as investigated from the business understanding stage to validate the structure of every feature obtained after the data import process. If any discrepancies have been found, the features can be translated to the required format in the next stage.

Reviewing the summary statistics is the simplest way to analyze and obtain an overview of the features. The summary statistics would have different elements of information for different types of features. As for categorical features, the summary statistics illustrate the cardinality of the feature and for numerical features, they detail the mean value of the feature along with median and other fixed quantile values. The categorical features, dependent and the independent variables, combined must be further evaluated for categorical count or percentage within their categories. This enables the study to identify imbalance in the cardinality between the feature under

inspection and the target feature.

Ideally, the next step is to visualize the data and to understand its aspects which cannot be observed otherwise as distribution and normality. Outliers for each feature can also be visualized to gain better insights. The relationship between multiple attributes could be visualized and understood better. Simple aggregations can also help to understanding the data at a macro level. Analyzing the sub-populations and their properties would help in getting an overview of any specific feature. Additionally, data quality can also be examined at this stage by understating the outliers, missing values, noisy data and also the inconsistent data. Each of these can have a different test performed to understand the issues and to devise a plan to resolve them in the next stages. Redundancy can also be identified at this stage.

In conclusion, the current problem is identified as a Classification task. From the data understanding phase, the dataset has been identified with a total of 563 features, with 561 numerical independent features, one categorical feature describing the target, Activity label and another categorical feature describing the user. Summary statistics with a combination of visualizations and other explorations would be a part of this section forming the data exploration report. Similarly, the quality of the data can also be judged and reported.

3.3 Data Preparation

Data preparation is one of the most crucial stages in a data mining project. This process determines the quality of the data used to extract insights which could in turn affect the quality of the insights. It is the process of manipulating the data and prepares it to be suitable for the next stages of the project. Data emerging from this stage must not contain any incomplete, noisy or inconsistent data. The major tasks in data preparation are as follows –

- Data Cleaning

  The first task under data cleaning is to evaluate the missing data in the entire data set. A data point can be said have missing data if there is either no data value stored in it or values as N/A, not applicable or 999 are stored instead of a blank field. Having missing values is a common occurrence and can have a significant effect on the feature and insights drawn from it. Once missing data is identified, it is crucial to know the mechanism behind it. Action can be taken by analyzing the type of missing data :

  - Missing Completely at Random (MCAR)
  - Missing at Random (MAR)
  - Missing Not at Random (MNAR)

  Appropriate solution must be devised and highlighted at the beginning of the experiment.

  The next task under data cleaning is outlier detection and analysis. An outlier as specified by Grubbs (1969) is a data point that is at a significant distance from the other data points. Generally, an outlier may arise due to an error in coding or data collection methods, but if the outlier is a genuine possibility, it can be quite insightful. A data point is only an outlier if it is significantly away from the normal curve, so box plots and histograms can help analyze this scenario. Once detected, it can be subjected to masking, swamping or other formal outlier tests as Grubbs test. All the details about the outlier and the test results must be noted before proceeding to the experiment (Croarkin et al., 2006).

  The next stage of data cleaning is handling noisy data. Zhu & Wu (2004) state noise to be an unavoidable problem in any data mining problem. There can be two types of noise, class noise and attribute noise. Class noise can occur when the target variable is incorrectly labeled with a different class or even contradictory examples, as similar observations yielding different class label values. Attribute noise refers to error in the values of features. It can be as an inaccurate value or unknown symbol

instead of an actual value or even incomplete values. One of the best solutions to avoid noisy data is to apply a noise filter and to collect only eligible values or binning data into several bins and thereby averaging the values to a common bin. Another solution could be to utilize modeling approaches that are resistant to noisy data as fuzzy decision trees (Blechner, 2005).

Data inconsistency is another problem that is quite common. Data points may suffer with inconsistencies in naming conventions, date formats or other data codes. This could result due to incorrect data representations, inconsistent use of codes and also due to inconsistencies while creating the original data source. The solution for this would be examining the data for unique rules, consecutive rules and null rules.

The final task of the data cleaning stage is to eliminate data redundancy. Having large amount of redundant data may distract the model from concentrating the important observations. Furthermore, redundant data may also slow down the entire process. Redundancy may be detected by performing correlation between the features which provides the relation between one feature over the other. The most popular correlation metric is the correlation coefficient. A correlation coefficient closer to positive or negative one can be treated as a strong relation, in which case, one of the features can be dropped and be represented by the other feature.

- Data Integration

  Data integration is the process of combining data from multiple sources. It is an important process and can aid in increasing the speed of the experiment and accuracy of the final prediction results. While performing integration, it is crucial to identify features that need to be matched and combined. The structure of such features has to be understood well along with their functional rules, referential constraints and dependencies which must match for both the integrated features and their datasets, if they fail to have similarities, the data and its conditions must be transformed to make required changes. Redundancy tests performed in the data cleaning stage must be evaluated to perform better integration. By end of this stage, we must be able to detect conflict between several values due to inequality in scaling, encoding

or representation and thus provide an appropriate resolution.

- Data Transformation

  The data are transformed into another form by applying a mathematical function so the resulting value is more suitable for the next stages of the data mining project. Data transformation utilizes a function that maps all the values of the feature to a new set of transformed values. There are different ways of transforming the data. An ideal method shall be devised based on the experiment and the data.

  – Aggregation is a technique when summarization is applied to the data. This transformation also enables in creating a data cube, which is a 3-D range of values which are aggregated from different aggregation and abstraction patterns[1]. Aggregation can help when each observation can be better represented as a single aggregated value as opposed to multiple values partially explaining the same feature.

  – Normalization is the technique of ensuring that the data falls within a fixed range of values. It is performed since the measurement of unit can have a significant effect on the final result. So to avoid dependency on the measuring unit, each value in a feature is scaled using the maximum - minimum or mean - standard deviation of all the values in the feature. Normalization tries to provide each feature the same amount of importance and avoids the algorithm to be influenced by higher values of a feature with lesser importance.

  – Feature construction is an important part of transformation of adding new features which can help in understanding the data better by providing a better meaning and improved accuracy. A new feature must be constructed such that it helps in improving the accuracy, computational efficiency, and can be generalized to different algorithms. However, there is always a risk of over fitting to the problem at hand while constructing new features which must be monitored. (Sondhi, 2009)

[1] "https://www.computerworld.com/article/2564238/business-intelligence/data-cubes.html"

- Data Reduction

  Data reduction is the technique of reducing the overall representation of the data while having minimal loss of domain knowledge and information content. Data reduction helps in decreasing the computational complexity of the algorithms owing to the complex data and its structures which in turn result in quicker analysis. These include methods such as sampling, feature selection and dimensionality reduction.

  – Sampling is the technique of selecting a subset of observations to represent and be analyzed instead of the entire data. It can be used when the data held is massive and computation can be expensive and/or time consuming. The notion behind sampling is to obtain a representative sample of the entire data. Types of sampling are random sampling, stratified sampling, cluster sampling and systematic sampling which can be used to derive samples from an imbalanced datasets.

  – Curse of dimensionality as stated by Bellman (2013) is the 'problem caused by the exponential increase in volume associated with extra dimensions added in the Euclidean space'. Dimensionality reduction and Feature selection are important techniques that help avoid the curse of dimensionality. Attribute subset selection is one of the methods of dimensionality reduction which detects and stores highly relevant features. Principal component analysis is a similar dimensionality reduction technique, which searches for a set of n-dimensional vectors that can best represent the data over the original feature set. The new set of vectors created is ranked based on the amount of information and variance it represents, so important information still intact.

- Data Split

  The final step in data preparation is to create training and test datasets. Each algorithm will be trained on the training dataset and will be evaluated against the testing dataset. The typical data split performed is 70 to 30 split between the training and test datasets. Also, it is important not to expose the test data set during training process. Typically a stratified random sampling is an ideal choice

to ensure that the composition of the data between the datasets matches to that of the original data source. (Liberty et al., 2016)

Once data preparation is completed after utilizing either one or more of the above techniques as per the original data conditions, quality and the experiment to be performed, the data must be Valid, Accurate, Complete, Consistent, Uniform and ready to be modeled.

3.4 Modeling

In this stage the final machine learning algorithms will be utilized to induce predictive models and gain insights from the prepared data in order to solve the research question. Each machine learning model has to be finely tuned for various variables and parameters to have the perfect model synced with the data. The literature review has identified multiple classification algorithms typically used in machine learning and in HAR domain. Comparing the potential algorithms discussed in the review and the algorithms already explored listed in table 2.1, the gaps of the research are pretty evident. Considering the No free lunch theorem introduced in the literature review (Wolpert, 2002), for the scope of this study, the below stated five models from four families of machine learning algorithms will be created.

- Information based learning –

  Under the information based family of machine learning, decision trees will be modeled. When modeling the decision trees, each and every parameter can be modified to create an apt classification model. The first parameter to be provided will be the target and the selected independent variables. The another important parameter is the splitting criteria which can either take the gini impurity or the information gain value to split the data into partitions. As the current study deals with a classification problem, the method must be specified as appropriate. There are several

controlling parameters which may be used to enhance or restrain the tree growth, the best approach would be to perform a trial and error procedure to choose the suitable parameters. The resultant decision tree can be inspected in several ways as graph of the decision tree plot, summary, cross validation results etc.

- Similarity based learning –

  For K-nearest neighbor algorithm, the independent variables and target feature must be specified to help the algorithm differentiate. The important aspect of the KNN algorithm is the 'K' value, which is the number of neighbors to consider. The experiment must be carried out with a 'tunelength' set to various values of 'K' to find the one that fits in best. The default distance used in KNN is the Euclidean distance. The final value of the target is decided by the majority vote of the K nearest neighbors in terms of the Euclidean distance.

- Probability based learning –

  The Naive bayes algorithm implementation takes in the independent and the dependent features as the first parameters. The algorithm can work with the data with default parameters alone; however, there are multiple other parameters that may be provided to customize it better. Laplacian correction can be used to smooth the categorical data which can be provided with a numerical value, the distribution type can be altered by editing the boolean value of 'usekernel' and the total bandwidth can also be adjusted.

- Error based learning –

  Multinomial logistic regression can be implemented by using a penalized logistic regression technique. The important parameter is the MaxNwts which must be set high enough for the algorithm to function. The summary of the logistic regression gives out the coefficient and the standard error for each of the independent features. Additionally, it also highlights significant values out of all the independent features.

  Artificial neural networks are complex structures that can be easily map to number of varied datasets. The size parameter plays a significant role of specifying number

of units in a hidden layer and it will take a single or a series of numbers to run the model with. The best way to choose the number of hidden units as suited by the data is to run multiple experiments and identifying the right number. The decay parameter is a regularization parameter which penalizes the model and helps avoid over-fitting. Although neural nets can be highly customized, they can be computationally expensive and also cannot be interpreted as well as other machine learning algorithms.

3.5 Evaluation

For the validation process, the K fold cross validation and the repeated random sub sampling techniques are both deemed most apt from the literature review. In a study by Kohavi et al. (1995), multiple accuracy estimation methods were studied. The experiment concluded that though K fold validation can be a good technique, with moderate K values (10–20) there could be reduced variation but the process may also suffer due to increased bias which could in turn tend to unstabilize the training process. Comparatively, stratification process in the RRSS method could be better for both variance and bias. Additionally, as the dataset under inspection has a high number of independent features, the RRSS method would be a faster approach over the K fold validation. So the repeated random sub sampling technique also called as Monte Carlo cross validation is preferred for the evaluation process.

A 5 fold variant of this technique is used. To execute the process, initially, five different samples of training and testing datasets must be created. Each of these five training sets must be individually modeled using the above discussed machine learning algorithms. Each model will be evaluated with the help of the corresponding test dataset that will be predicted using the predict function. The predict function, predicts the target value for each records of the test data set using rules and insights from each of the models created. The resultant solution is a set of values predicted

for each of the test data set as per its functionality of the model. This solution is compared to the original solutions given from the base study. The correct and incorrect answers are tabulated and displayed as the confusion matrix. The confusion matrix is selected as it is good way to evaluate the models and presents the precision, recall and the overall classification accuracy in a comprehensible format and hence can be very useful. However, since the data does not have a binary target feature, the confusion matrix has to be expanded to accommodate all the six classes of the target feature.

Using each of the classification accuracies produced for each induced machine learning model, a distribution can be created using the five cross validation results. This distribution gives the overall classification accuracy that can be expected of the algorithm. Furthermore, depending on the normality of the each of the distributions, their results can be tested for statistical significance using a corresponding significance test as T- test or Wilcoxon Signed-Rank test.

3.6 Deployment

The current project does not aim to be deployed in the real time environment, so this section will not cover any details in this aspect. However, the project aims to contribute to the body of knowledge and the current document suffices that purpose.

3.7 Strengths & Limitations

This section evaluates the strengths and limitations of the proposed design. Firstly, to increase the strength of the project, the experiment has chosen to utilize algorithms from one family of machine learning models each, this ensures, the data is being exploited through various aspects dealt under each of the machine learning family.

The parameter tuning will be performed with multiple values to ensure the model is well fitted with the data. Also, as the models are validated multiple times with various samples of the existing data, the variation due to samples will be measured and the data is well exploited to extract utmost information.

However, a potential drawback could be that the study is willing to take a risk of being affected by large bias that might occur to application when using bootstraps due to the chosen evaluation procedure (Kohavi et al., 1995). Also the RRSS validation uses only 5 samples. Using a higher number could yield more significant analysis.

3.8 Summary of Design

This chapter has produced a detailed design of the entire experiment to be conducted. Initially, the objective of the study is understood well along with the restrictions and assumptions. The data is then examined to provide an initial data quality, data description and data exploration reports. Data is then prepared by performing many of the data quality improvement operations. Data transformation techniques are then applied to create a final dataset. This dataset is sampled to create five different sets of training and testing datasets on which various models are built on. The resultant classification accuracy distributions can be used to identify the best model and evaluate against the benchmark. One potential problem of the design is the computational complexity due to the huge size while feature engineering and modeling the algorithms, hence generating each model could be a time consuming approach.

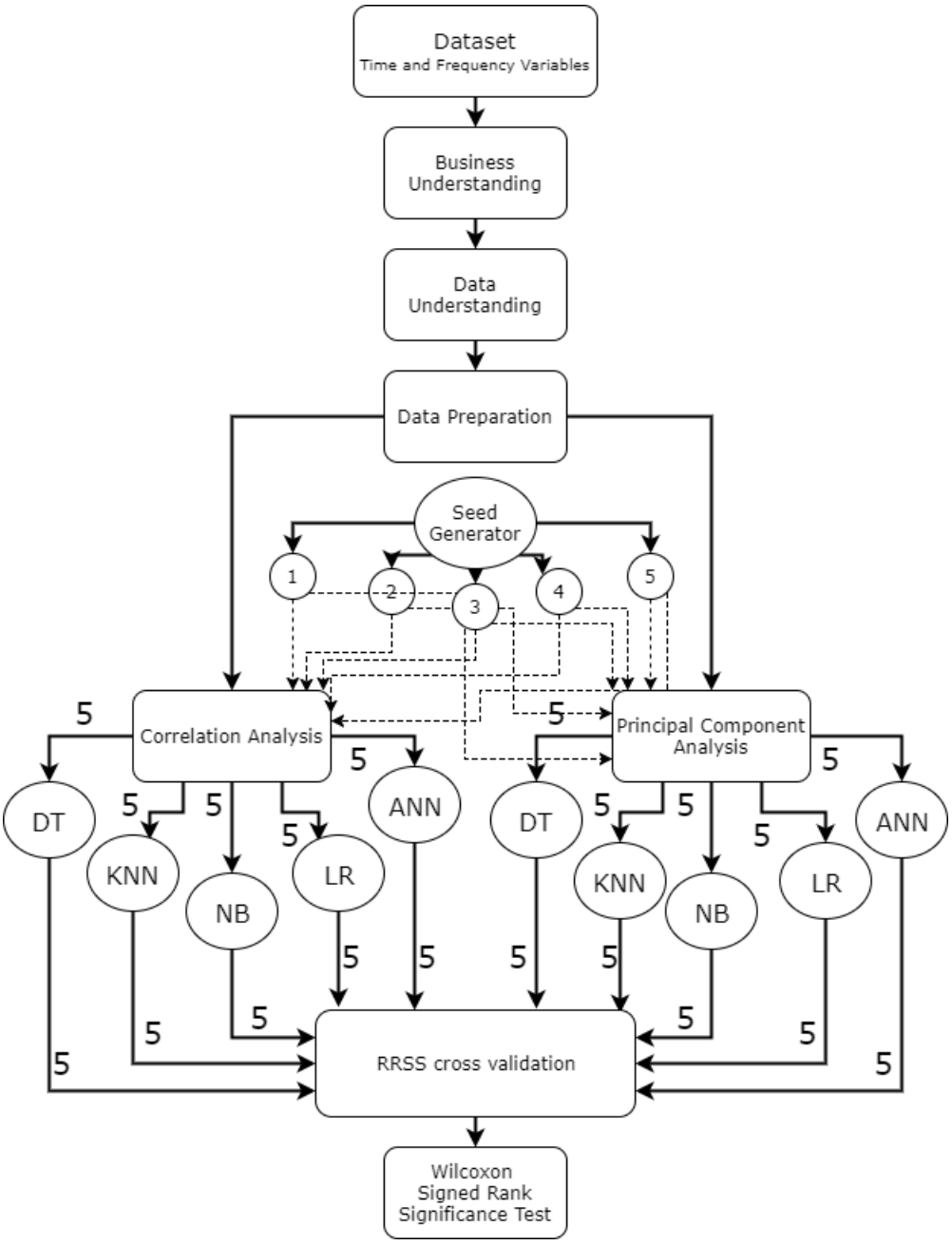

Figure 3.2: Experiment Design Diagram

The next chapter details the practical implementation of the proposed approach along with the results obtained.

Chapter 4

Implementation & Results

This chapter describes the execution procedure and the results of the experiment. The aim of this chapter is to implement the experiment in the direction of being able to answer the research questions. The structure of this chapter follows the CRISP - DM methodology to have a comprehensive and a standardized approach and is similar to the chapter 3. However, any deviations from the originally created design will be acknowledged and commented.

4.1 Data Understanding

To understand the data and its elements, the data must be initially integrated and put in an ideal format to inspect it; this process of data integration is performed and specified under the data preparation stage. The first element of investigation performed on the integrated data is calculating the dimensions using the function 'dim'. There were 10299 rows, which is the train and test set combined and 563 columns consisting of 561 features, the subject and the activity variable. The names of each of the columns are evaluated to ensure correct order of the data.

```
(names(fulldata))
 [1] "tBodyAcc-mean()-X"                "tBodyAcc-mean()-Y"
 [3] "tBodyAcc-mean()-Z"                "tBodyAcc-std()-X"
 [5] "tBodyAcc-std()-Y"                 "tBodyAcc-std()-Z"
 [7] "tBodyAcc-mad()-X"                 "tBodyAcc-mad()-Y"
 [9] "tBodyAcc-mad()-Z"                 "tBodyAcc-max()-X"
[11] "tBodyAcc-max()-Y"                 "tBodyAcc-max()-Z"
[13] "tBodyAcc-min()-X"                 "tBodyAcc-min()-Y"
[15] "tBodyAcc-min()-Z"                 "tBodyAcc-sma()"
[17] "tBodyAcc-energy()-X"              "tBodyAcc-energy()-Y"
[19] "tBodyAcc-energy()-Z"              "tBodyAcc-iqr()-X"
[21] "tBodyAcc-iqr()-Y"                 "tBodyAcc-iqr()-Z"
[23] "tBodyAcc-entropy()-X"             "tBodyAcc-entropy()-Y"
[25] "tBodyAcc-entropy()-Z"             "tBodyAcc-arCoeff()-X,1"
[27] "tBodyAcc-arCoeff()-X,2"           "tBodyAcc-arCoeff()-X,3"
```

Figure 4.1: Example set of feature names

The structure of the dataset was attempted to be analyzed but as the data size was high, it was not quite comprehensible. The six point summary of the data, which includes the minimum, maximum, mean, median and the 1st and 3rd quartiles are generated for each of the features. These didn't add value either. The attributes of the data: names of the columns, rows and the class of the dataset are viewed. To have a quick view of the data, the first two rows of the data are observed using the head function. To understand the composition of the subject and the Activity levels, their frequencies were tabulated, the figures 4.2 & 4.3 are the histograms of the Activity and the subject variables. It is observed that both the fields were free of any class noise or data inconsistency issues.

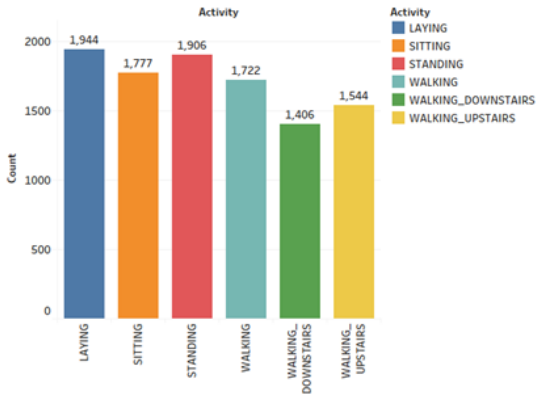

Figure 4.2: Histogram of the target variable – 'Activity'

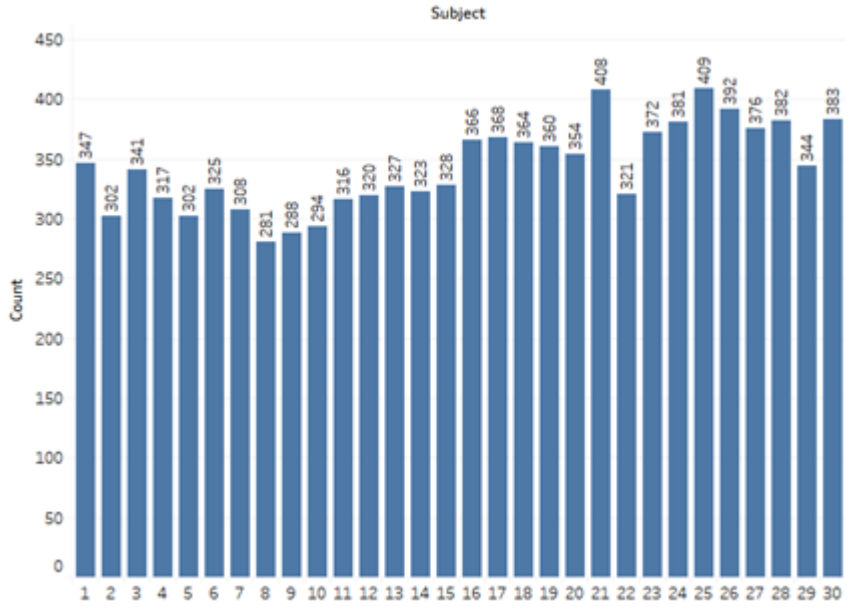

Figure 4.3: Histogram of identifier variable – 'Subject'

Unique rows were analyzed to see for any presence of attribute or class noise, but there were no signs of such errors. The maximum and minimum values of the 561 time and frequency domain variable were seen to be +1 and -1 respectively which stated that the values were normalized between this range while creating the dataset. There were no missing values found under the missing value analysis test.

The users and the activity levels were tabulated to inspect their distribution using simple aggregation methods. It is observed that their composition was quite uniform, with respect to each other and the entire data.

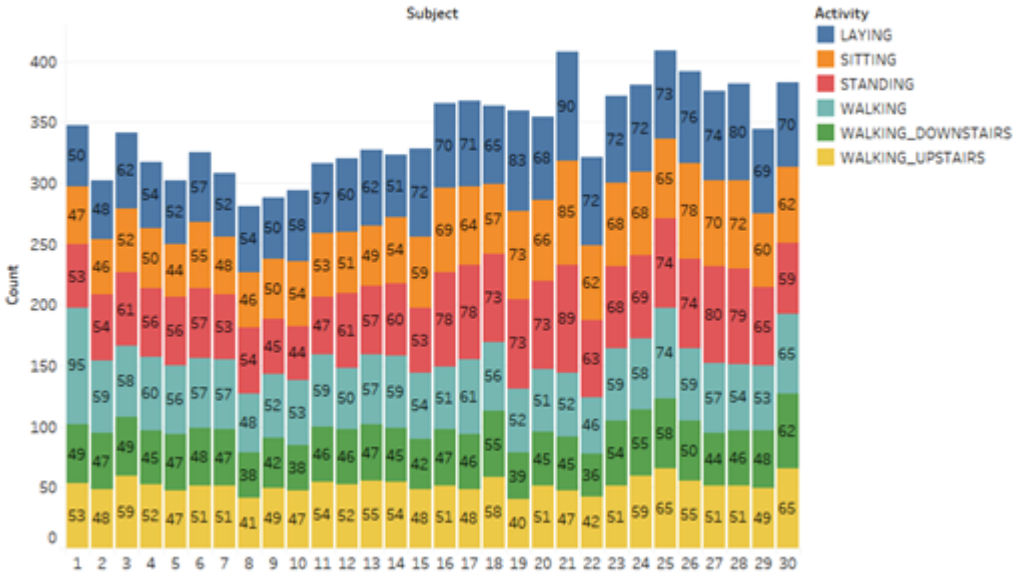

Figure 4.4: Histogram of records per user grouped by target

4.2 Data Preparation

- **Data Integration**

  The first step under preparing the data is the data integration which is performed before the data understanding stage as the original data is provided as a .zip folder containing multiple text files each representing different components of the full dataset. The first component is the 'activity labels' text file which has the 6 activities listed. The next file is the 'features' text file consisting of all the 561 features, additionally, a 'features info' text file consists of descriptions of these features. There exist two sub folders namely 'train' and 'test' representing the training and the testing datasets. Each folder has an 'X' and 'Y' text file representing the independent and the target feature values. A 'subject' file is given to provide the information of the user performing the activity. Each of these files is captured into multiple tables and are copied onto individual lists using a blank separator. Each file is also viewed and examined to validate the data consistency. The training data set is created by

merging the features (X), the subject and the activity (Y) under the train folder and the testing data set is created by merging the same files from the test folder. The final train and the test files are obtained such that the user subjects - 2, 4, 9, 10, 12, 13, 18, 20, and 24 are in the testing dataset and the remaining user form the training dataset. This process of completely ignoring users from being a part of the dataset can be misleading. Hence, both the datasets are merged to create the complete "Fulldata" dataset.

- **Data Manipulation**

  The next step under the data preparation is converting the data into their right primitive data types. The Activity and Subject variable are converting into categorical variables using the function "as.factor". All the other independent features are converted into numeric using "as.numeric". The feature names are further edited to remove unnecessary spaces and special characters as brackets, quotes and dots, hyphens were replaced with underscores to have syntactically correct character names using the 'gsub' function. Outlier analysis and distribution tests were not performed as per the design solution as they can be hard both for computation and analysis, also as the features were previously normalized to a fixed range, there could be no further issues.

- **Dimensionality Reduction**

  High dimensionality as seen previously can cause severe difficulties as it could be increasingly hard to visualize and understand the data, it can also be computationally complicated and expensive. An ideal solution to this problem is to use dimensionality reduction techniques to bring down the size of the data. The study utilized two techniques to decrease the size of the feature set.

  – Correlation Analysis

    Highly correlated features can downgrade the performance of a model. Moreover, it's not ideal to have multiple features measuring the same variability of the target feature. So in order to decrease multicollinearity in the data, highly correlated

variables are identified and removed from the dataset. For this, initially, a correlation matrix is generated for all the 561 independent features. This produces a 561x561 matrix with correlation values between all the features.

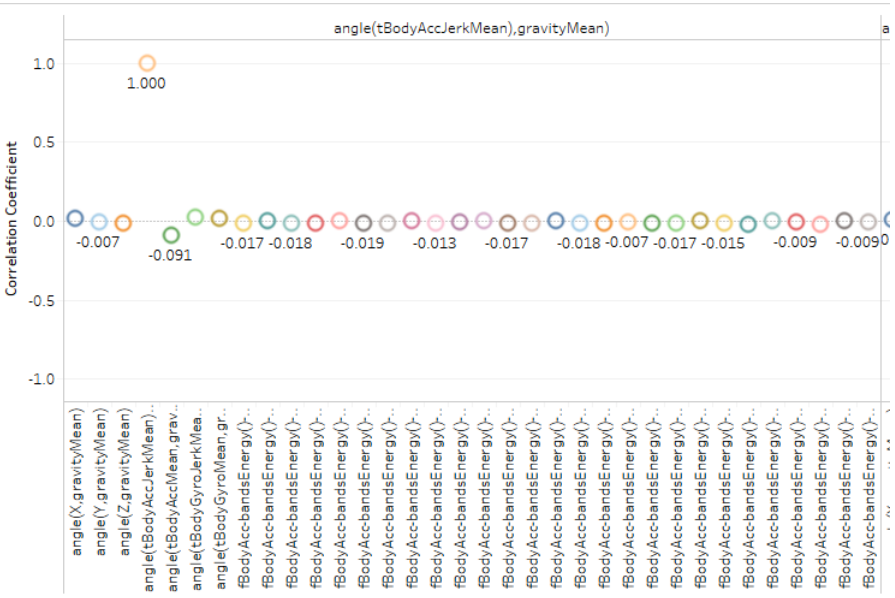

Figure 4.5: Example of feature with low correlation coefficient values

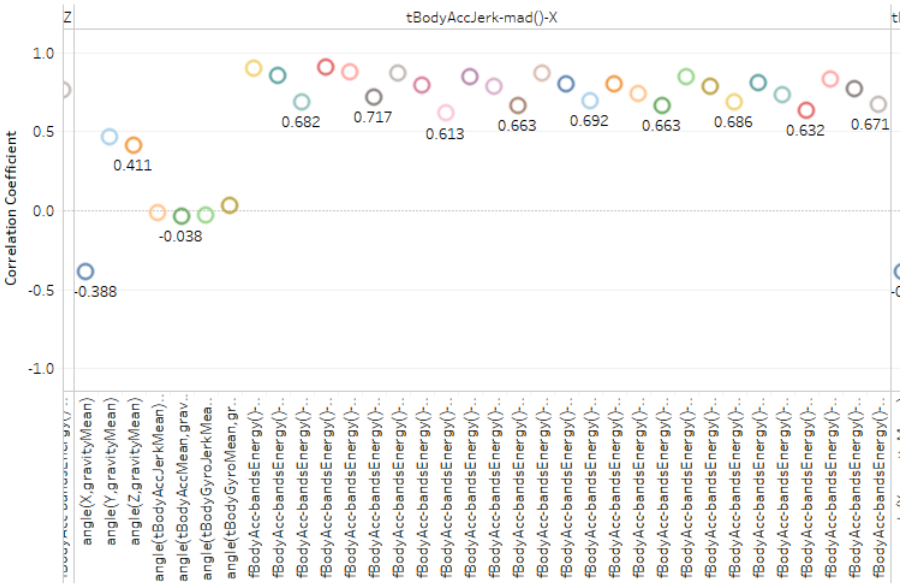

Figure 4.6: Example of feature with high positive correlation coefficient values

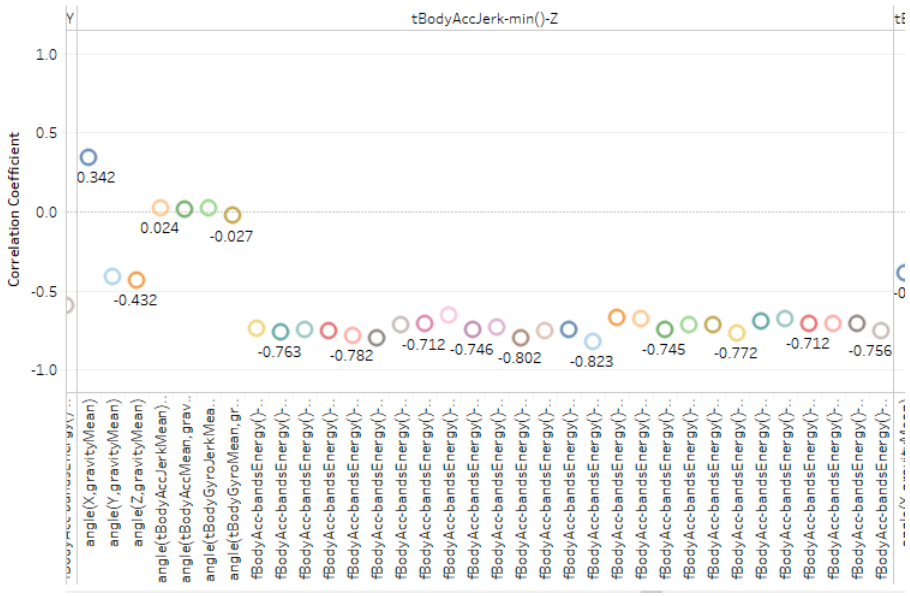

Figure 4.7: Example of feature with high negative correlation coefficient values

This correlation matrix is given as input to the 'findCorrelation' function in the caret package along with a cut off value of 0.8 of pairwise absolute correlation, which denoted strong correlation between the features (Taylor, 1990). This function resulted in the list of 389 feature names which are highly correlated with the other features and in a pair possess a higher mean absolute correlation over the other. Finally, an uncorrelated dataset is created using only the features not in the highly correlated list generated above. This resulted in a dataset with 174 uncorrelated features. From this data the Subject feature which acts as an identifier is removed. The final uncorrelated dataset is inspected for dimensions and has 173 columns

Rows	**Columns**
10299	173

Table 4.1: Dimensions of the dataset after Correlation Analysis manipulation

– Principal Component Analysis

This technique replaces all the set of features with a linear combination of them called as principal components. To generate the principal components of the

feature set, the 'prcomp' function is utilized. The input to the function are the full list of features without the subject and the activity fields and a boolean variable of scaling given as true, to ensure that the features are scaled to have unit variance. The resultant object has 5 elements as standard deviation, rotation, centre, scale and a matrix with the new components which are ordered on basis of decreasing importance and relevance levels with respect to the variability being captured.

The individual component variance measured is calculated by squaring the standard deviation values. Using the first 100 features, about 94.6% of the total variance is captured which is enough data captured for the proportion of data reduced.

```
> head(PCA_Prop_Var, 10)*100
 [1] 50.7382210  6.2391858  2.6925639  2.4528710  1.8889357
1.6313954  1.4145332  1.2162106
 [9]  0.9852476  0.9492282
> sum(head(PCA_Prop_Var, 100)*100)
[1] 94.67115
```

Figure 4.8: Variance captured by top 10 components of PCA

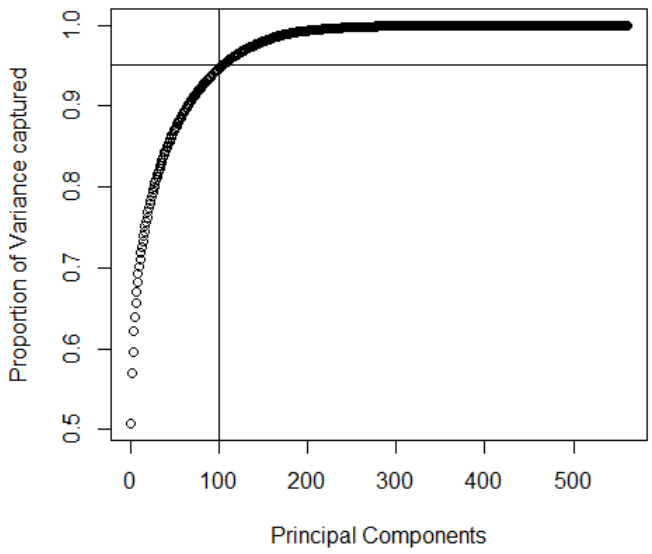

Figure 4.9: Proportion of Variance captured by top 100 components of PCA

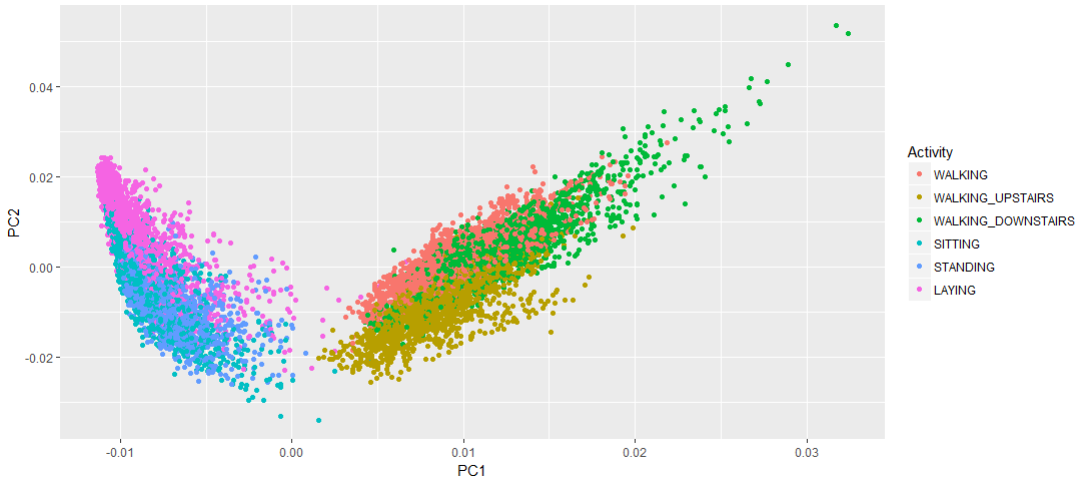

Figure 4.10: Variable space approximated by the top 2 principal components

The amount of variance captured by the top 100 components and the variable space approximation captured in 4.10 are evident that the original data can be approximated using principal components. A new data frame is hence created with the Activity field and the first 100 principal components.

Rows	**Columns**
10299	101

Table 4.2: Dimensions of the dataset after PCA manipulation

4.3 Modeling

The models must be implemented in accordance to the model validation techniques. To perform an RRSS cross validation, the data has to be modeled on 5 different training sets. The 5 training sets are created using 5 seeds, '1', '2', '3', '4', '5' are incorporated to generate random partitions using 'createDataPartitions' function from the caret package. The function takes into account the target variables and performs a stratified sampling of the data and outputs a percentage of numbers as per the parameter setting with the percentage cardinality of the target feature similar to the

original target feature. The Activity variable is given as an input to the function, the percentage of values to be generated is set as 0.7, which is the percentage of the training set against the test set. The figure 4.11 describes the first 10 numbers of the count variable from different seed values. It can be observed that each sample is random and is different from other samples.

trainCount_s1	int [1:7212, 1] 1 2 3 4 6 7 8 10 12 15 .
trainCount_s2	int [1:7212, 1] 1 2 3 5 6 8 11 12 13 14
trainCount_s3	int [1:7212, 1] 3 5 6 7 8 9 10 12 13 15
trainCount_s4	int [1:7212, 1] 1 2 6 7 8 9 10 13 14 15
trainCount_s5	int [1:7212, 1] 1 2 3 5 6 7 9 10 11 13 .

Figure 4.11: Head of train count seed variables

Train datasets are created by extracting the records corresponding to the sampled numbers. All the other rows which correspond to the remaining 30% of the total records are placed into the test datasets. The final dataset obtained from the correlation analysis will be used to derive the 5 train and test sets to follow the first method of data preparation. The dataset created with the 100 principal components will be used to prepare train and test datasets following the second method of data preparation. Once the train and test datasets corresponding to each of the manipulated dataset under every seed is created, modeling can be performed.

The train function of the caret package is used for creating each of the models. To enable resampling in the later stage, these models are being trained using a train control performed using cross validation in two folds.

The first model to be created is the decision tree with the splits being performed using the Gini impurity value as it is faster and works better for continuous attributes as in this scenario[1]. Gini impurity also works in accordance to minimize the misclassification rate, which is quite suitable for the study. The figure 4.12 shows the list of the most important features observed.

[1] "http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/"

```
> print(varImp(dt_Uncorr_s1))
rpart variable importance

  only 20 most important variables shown (out of 173)

                                Overall
fBodyAcc_bandsEnergy_9_16        100.00
fBodyGyro_bandsEnergy_9_16.2      99.12
fBodyAcc_bandsEnergy_9_16.1       99.02
fBodyAccJerk_bandsEnergy_25_32    98.93
fBodyGyro_bandsEnergy_17_24.2     98.79
tGravityAcc_energy_Y              81.34
tGravityAcc_energy_Z              64.91
angle(Z_gravityMean)              63.08
tGravityAcc_arCoeff_X_1           38.81
tGravityAccMag_arCoeff1           36.84
fBodyGyro_maxInds_Z               35.19
tGravityAcc_arCoeff_Z_4           34.64
tGravityAcc_arCoeff_Y_4           33.48
tGravityAcc_entropy_Y             30.71
tBodyAcc_correlation_Y_Z          26.69
fBodyBodyAccJerkMag_kurtosis       0.00
fBodyGyro_bandsEnergy_33_40.2      0.00
fBodyAcc_maxInds_Z                 0.00
tBodyAccJerk_arCoeff_Y_3           0.00
fBodyGyro_bandsEnergy_9_16.1       0.00
>
```

Figure 4.12: Variable importance as observed by Decision tree

For K-Nearest neighbors' algorithms, the tune length is set to '7, which seems an ideal solution to compensate between increasing value of accuracy and tuning and execution time of the model. To iterate the training procedure over K, the model used cross validation technique. Due to cross validation over multiple values of K, the modeling took a lot of time. From the summary of the KNN model shown in 4.13, it is seen that K value of 5 is best suited for the data.

```
> knn_Uncorr_s1
k-Nearest Neighbors

7212 samples
 172 predictor
   6 classes: 'WALKING', 'WALKING_UPSTAIRS', 'WALKING_DOW
NSTAIRS', 'SITTING', 'STANDING', 'LAYING'

No pre-processing
Resampling: Cross-Validated (2 fold)
Summary of sample sizes: 3606, 3606
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9148641  0.8975246
   7  0.9133389  0.8956897
   9  0.9116750  0.8936897
  11  0.9082085  0.8895230
  13  0.9057127  0.8865086
  15  0.9025236  0.8826651
  17  0.9014143  0.8813314

Accuracy was used to select the optimal model using  the
largest value.
The final value used for the model was k = 5.
```

Figure 4.13: Summary of the KNN Model

For the Naive bayes, the default settings are used with no use of kernel function. It was the quickest execution out of all the models.

The penalized multinomial logistic regression is also run on default settings, but the maximum weights has to be set high enough to enable the model to be executed. The model summary provided an AIC value of 2423.762

```
> print(varImp(lr_Uncorr_s1))
multinom variable importance

  only 20 most important variables shown (out of 172)

                              Overall
tBodyAcc_entropy_X             100.00
tGravityAcc_energy_Y            86.09
tGravityAcc_energy_Z            83.04
tBodyAcc_correlation_X_Y        68.80
tGravityAcc_entropy_X           61.25
tBodyGyroJerk_arCoeff_X_1       58.92
fBodyAccMag_kurtosis            57.08
fBodyGyro_bandsEnergy_1_8.1     56.10
tBodyAcc_entropy_Z              51.40
tBodyGyro_correlation_X_Y       50.53
tGravityAcc_entropy_Y           50.08
fBodyAcc_bandsEnergy_9_16.1     47.38
`\\`angle(Z_gravityMean)\\``    45.45
fBodyAccMag_meanFreq            45.41
tBodyGyroJerk_arCoeff_X_2       44.38
tBodyGyro_entropy_X             43.38
tBodyGyro_correlation_Y_Z       43.11
fBodyAcc_skewness_Y             42.60
tGravityAcc_sma                 41.76
tGravityAccMag_min              39.74
>
```

Figure 4.14: Variable importance of the Multinomial Logistic Regression model

The neural net is initially modeled with a combination of units of hidden layer and decay value. It is identified that by using 30 units in the hidden layers with a decay value of 0.1 the algorithm converges considerably faster and the model doesn't over-fit either.

4.4 Evaluation

The evaluation is performed by predicting the target values for the testing datasets using their corresponding models. The prediction is implemented using the 'predict.train' function, from the caret package, by supplying the model and the test dataset to it.

It is important to remove the original target feature from the test dataset. Once the predictive model results are generated, a confusion matrix is plotted using the function 'confusionMatrix' with data input as the predicted values and reference as the original target feature labels. The overall component of the confusion matrix generated has the overall statistics as accuracy levels, sensitivity, specificity, kappa value etc.

A set of five accuracy values are generated for every model under each of the feature engineering technique. The resultant distribution of classification accuracies is analyzed to pick the best algorithm for the data. The results for each of the algorithm with both the methods are listed for each of the seed value.

Decision Tree		
	Dimensionality Reduction	
Seed	**Correlation Analysis**	**Principal Component Analysis**
1	0.6242306	0.5231616
2	0.6135407	0.5176547
3	0.617104	0.5192744
4	0.5150632	0.6096534
5	0.6103013	0.6161322

Table 4.3: Accuracy list for Decision Tree algorithm

K Nearest Neighbor		
	Dimensionality Reduction	
Seed	**Correlation Analysis**	**Principal Component Analysis**
1	0.9342404	0.9089731
2	0.9497894	0.9063816
3	0.9335925	0.9060577
4	0.9280855	0.9115646
5	0.9345643	0.9028183

Table 4.4: Accuracy list for K Nearest Neighbor algorithm

Naive Bayes		
Dimensionality Reduction		
Seed	**Correlation Analysis**	**Principal Component Analysis**
1	0.8519598	0.8088759
2	0.8224814	0.8234532
3	0.8496923	0.8176223
4	0.8568189	0.8253968
5	0.8386783	0.8231293

Table 4.5: Accuracy list for Naive Bayes algorithm

Logistic Regression		
Dimensionality Reduction		
Seed	**Correlation Analysis**	**Principal Component Analysis**
1	0.9617752	0.9549725
2	0.9624231	0.9540006
3	0.9656625	0.9585358
4	0.9663103	0.9598316
5	0.9721412	0.9582119

Table 4.6: Accuracy list for Logistic Regression algorithm

Artificial Neural Network		
Dimensionality Reduction		
Seed	**Correlation Analysis**	**Principal Component Analysis**
1	0.9773243	0.9689018
2	0.9708455	0.968254
3	0.9701976	0.9766764
4	0.9782961	0.9714934
5	0.9750567	0.9737609

Table 4.7: Accuracy list for Artificial Neural Network algorithm

The next chapter critiques the implementation and working of every model. Each of these accuracy values are also plotted to and evaluation is performed on their distributions.

A final statistical test is then performed on the distribution to get the statistical significance of the tests in order to answer the research question.

Chapter 5

Analysis, Evaluation & Discussion

This chapter performs an in depth analysis of the experiment and the results obtained from the design implementation as stated in the Chapter 4. The results are obtained from the predictive capabilities of each of the models induced based on algorithms using the different data preparation methods as Correlation Analysis and Principal Component Analysis applied to each of the different seeds of the data. Each of these results will be discussed individually before commenting on the general implementation and design strategy. The results of statistical significance of the models shall also be commented upon. The chapter will then conclude by stating the strengths and limitations of the experiment.

This chapter will also follow the structure and order of content as of the previous chapters to enable better association and understanding.

5.1 Data Understanding

The Data Understanding phase proved to be quite insightful in establishing the context of the data preparation to be performed or in order to estimate the complexity in

terms of space and time for creation of the models as well as predicting the final results. The initial analysis of the data depicted that the Activity feature which is the target variable for the experiment is well balanced as seen in figure 4.2. The balance of the target feature between its classes is an important aspect to investigate as necessary action must be taken care for any deviation in the balance during sampling stage or while creating train and test datasets. Furthermore, the subject, which is the identifier variable is also well balanced within itself as seen in figure 4.3 as well as when contrasted with the target feature, seen in 4.4. This eases the case of creating train and test datasets without requiring any stratified sampling to ensure the composition of each of the data splits which will be analogous to the original data composition.

It is also seen that all the independent features fall in a specific range of +1 to –1. It can be understood that the data was range normalized to fit to this particular range. It permits the algorithms to provide equal emphasis on every feature and allow better standardized data into the next stages and not allowing various ranges of the data to affect the significance of any feature.

5.2 Data Preparation

The independent features of the experiment are derived from the raw signals from two sensors of a single device. When performing an action, it is evident that the data is highly correlated and that multiple values could be representing the same component of variance in the target feature as in any tri-axial dataset (Mannini & Sabatini, 2010). Furthermore, each of these features measure tiniest aspects of the activity performed as for instance, the magnitude of the jerk measured through the body acceleration in the x- axis. This value would not be as different as the value measure in y or z axis or that of its minimum or maximum value during the sampling period of 2.5 seconds. It can be understood that there is a potential for redundant data to mislead the algorithm. In addition, excessive and irrelevant data might also lead to over fitting

in the model.

As dimensionality reduction was a crucial step in preparing the data for the modeling stage, the idea was to choose more than one technique for this task. Under correlation analysis, a not very stringent value of the correlation coefficient was chosen but it proved quite significant by identifying about 70% of the total features as highly correlated with each other. The example of uncorrelated feature was the jerk signal in the X–axis obtained from the body linear acceleration derived in time. It was partially correlated with the Y and Z axis values of the same metric but the correlation wasn't high enough to eliminate one of them. However, the derived features as describing the maximum, minimum values have observed to have a high correlation among the other derived features. It is quite intuitive that the interquartile range and standard deviation features possess high positive correlation whereas energy band and the minimum values of a metric demonstrate high negative correlation. Many such highly correlated features, both positive and negative were eliminated.

Principal component analysis was thought to be a better dimensionality technique during the design phase, due to the fact that the features could be replaced with their linear combination, with each component capturing certain amount of measurable variance in the target feature. The results of the technique were quite impressive too. With just 18% of the total number of features, about 95% of the total variance was being captured. Plotting all the principal components against the target feature in figure 4.10 also demonstrated a good understanding of the activity as all the stationary activities as standing, sitting and lying down were differentiated from the motile activities as walking in a line and on the stairs.

5.3 Modeling

To utilize the repeated random sub samping cross validation technique for the evaluation phase, each of the models must be trained and tested on five different combinations of the complete data to ensure that the created models and the prediction results were significant and can be replicated when necessary. In order to create the different samples of train and test datasets, stratified sampling technique was used. However, the random sampling could have worked equally good as the data is well balanced in terms of the target feature but stratified sampling provides greater certainty in avoiding any bias in the models due to the dataset composition. The image 5.1 shows the composition of the target features in the entire dataset and also in each of the created train and test data samples.

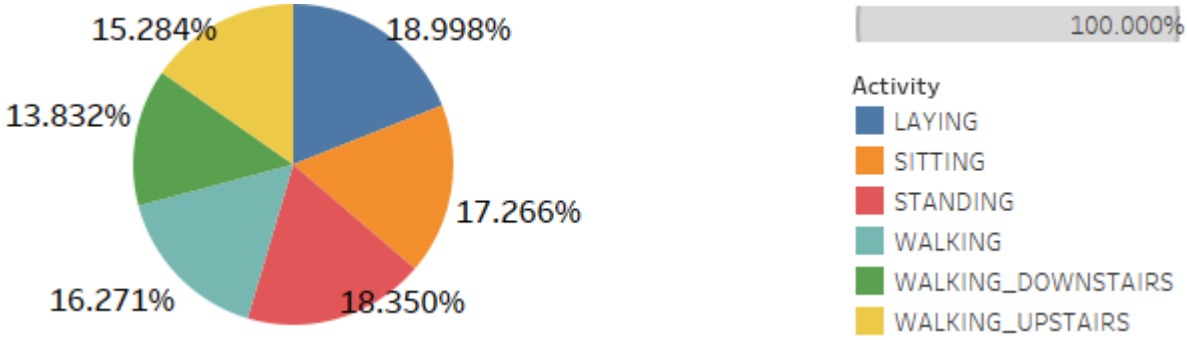

Figure 5.1: Target Feature Distribution

Models are created using the appropriate modeling technique and settings to yield various predictive results. The first model implemented was the decision tree, which is one the most powerful algorithms and yet can be highly influenced by its structure and thus yield drastically different results. The table 5.1 exhibits the predictive results for the decision tree with best accuracy value of all the seeds and methods. It can be observed that the decision tree has suffered in differentiating the activities from each other. Especially, the sitting activity is repeatedly misclassified as standing. One of the major flaws of the decision tree that could have impacted the performance of the algorithm over the given data could be the fact that the decision tree works better with

categorical variables. The reason is quite intuitive that with categorical independent variables, it is easier for the algorithm to create rules to perform partitions in the data in order to develop a tree with leaf nodes stating the appropriate target feature. However, in the current case, as the independent features are all numeric, it can be quite challenging for the algorithm to find the right value that can be used to bifurcate the tree to create partitions. Additionally, the decision trees are sought out due to its ability to provide highly interpretable results but in this case, with such huge amounts of data features, interpreting the rules and the tree partitions could add very little value.

	Reference					
Prediction	Walking	Walking_ Upstairs	Walking_ Downstairs	Sitting	Standing	Laying
Walking	441	68	111	0	1	0
Walking_ Upstairs	75	395	310	0	0	4
Walking_ Down-stairs	0	0	0	0	0	0
Sitting	0	0	0	0	0	0
Standing	0	0	0	475	556	44
Laying	0	0	0	58	14	535

Table 5.1: Confusion Matrix For Decision Tree

The next model generated is the K nearest neighbors algorithm. It was one of the models that has performed remarkably in producing results with accuracies better than the benchmark result. The table 5.2 depicts the confusion matrix for the best K nearest neighbor algorithm. It is seen that the algorithm has very less misclassifications on the whole. Considering the nature of the activity, the two clusters of activities, stationary and mobile, the KNN algorithm ensured high inter cluster classification accuracy.

But the intra cluster classification accuracy was not very satisfying especially for the stationary activities. The sitting and standing activities were highly misclassified. As the current experiment involves detecting the user activity over the span of 2.5 seconds, it is evident that the activity is influenced by the actions preceding and following it. So the KNN algorithm which takes into account the specified number of neighbours for performing classification justifies its satisfactory performance. But it must be noted that the modeling and evaluation of the algorithm was computationally very demanding and has taken the highest amount of time.

	Reference					
Prediction	Walking	Walking_ Upstairs	Walking_ Downstairs	Sitting	Standing	Laying
Walking	513	3	12	0	0	0
Walking_ Upstairs	1	456	6	3	0	0
Walking_ Down-stairs	2	4	403	0	0	0
Sitting	0	0	0	473	52	10
Standing	0	0	0	54	519	5
Laying	0	0	0	3	0	568

Table 5.2: Confusion Matrix For K Nearest Neighbours

The Naive bayes model is simple generative model that performed an indirect computation of the required probability through the Bayes function. The table 5.3 shows the confusion matrix of the Naive bayes algorithm with the best accuracy. The classification accuracy is not noteworthy compared to the results from other models and is also inadequate compared to the benchmark results. The poor performance of the Naive bayes can be implied to the innate assumption of parameter independence.

In the current experiment, each of the features is a derived attribute from the originally collected metrics and hence neglecting the interactions between the features can result in loss of plenty of information and proved damaging. But it must also be noted that the algorithm was the fastest to converge compared to all the other algorithms.

	Reference					
Prediction	Walking	Walking_ Upstairs	Walking_ Downstairs	Sitting	Standing	Laying
Walking	443	7	16	0	0	0
Walking_ Upstairs	47	432	67	10	11	11
Walking_ Down-stairs	26	24	338	0	0	1
Sitting	0	0	0	399	87	4
Standing	0	0	0	112	466	0
Laying	0	0	0	12	7	567

Table 5.3: Confusion Matrix For Naive Bayes

The multinomial logistic regression model was aimed to act as a baseline and was expected to provide mediocre results. However, the table 5.4 can showcase the efficiency of the algorithm for the given data. The logistic regression was one of the best performing models and provided exceptional classification accuracy and was greater than the benchmark accuracy value. The ability of the algorithm to handle any non linear effects and reduce the influence of noise could have been the factors corresponding to the increased predictive accuracies. It can be observed that there is low inter cluster misclassification by the model but some amount of intra cluster misclassification for the stationary activities. The model also took plenty of time to converge but was more efficient compared to the time complexity of the KNN

algorithm. Additionally, the virtue of the algorithm of not expecting the data to adhere to normality and linearity assumptions has proved favorable to the experiment.

	Reference					
Prediction	Walking	Walking_ Upstairs	Walking_ Downstairs	Sitting	Standing	Laying
Walking	515	0	8	0	0	0
Walking_ Upstairs	1	460	4	0	0	0
Walking_ Down-stairs	0	3	409	0	0	0
Sitting	0	0	0	478	47	0
Standing	0	0	0	54	524	0
Laying	0	0	0	1	0	583

Table 5.4: Confusion Matrix For Multinomial Logistic Regression

The final individual algorithm was the artificial neural net. As seen in section 2, the ANN algorithm was highly credited in the HAR task. The ability of the ANN to implicitly detect complex nonlinear relationships and also to acknowledge and utilize the interactions amongst the independent variables are two of the biggest advantages of the algorithm that have permitted in capturing the essence of the data resulting in the high accuracy values. The table 5.5 presents the confusion matrix of the best ANN model. It can be seen that the mobile activities have been classified exceptionally well and so was the lying down activity with 100% classification accuracy. However, standing and sitting activities were slightly misclassified and the reason is unknown as the algorithm is a black box and cannot be used to interpret. Additionally, the ability of the algorithm to generalize well due to its associative memory ensures that the classification results for all the different samples of data are quite similar to each other.

	Reference					
Prediction	Walking	Walking_ Upstairs	Walking_ Downstairs	Sitting	Standing	Laying
Walking	515	0	8	0	0	0
Walking_ Upstairs	1	460	4	0	0	0
Walking_ Down-stairs	0	3	409	0	0	0
Sitting	0	0	0	478	47	0
Standing	0	0	0	54	524	0
Laying	0	0	0	1	0	583

Table 5.5: Confusion Matrix For Artificial Neural Network

5.4 Comparison of Dimensionality Reduction Techniques

The dimensionality reduction techniques utilized to defend the experiment from the curse of dimensionality have helped in enabling to execute the algorithms on the given machine with limited space and time constraints. The techniques of feature subset selection and feature extraction were used. The correlation analysis was aimed to create a subset of features which has each feature highly correlating with the target and very less correlated with the each other. Principal component analysis performs linear feature extraction. It uses the covariance matrix, eigenvalues and eigenvectors to derive the uncorrected eigenvectors, each representing a part of the variance in the target feature.

The figure 5.2 demonstrates the accuracy values for each of the model grouped by the feature reduction techniques. It is evident that the Correlation Analysis technique has increased accuracy values for every algorithm. The PCA despite being a strong technique failed in providing sufficient results. It could be as a result of no direct relationship between the variance and the predictive power, which resulted in eliminating useful information that could have been effective for classification. However, by selecting the highly correlated features, there was little scope to lose information.

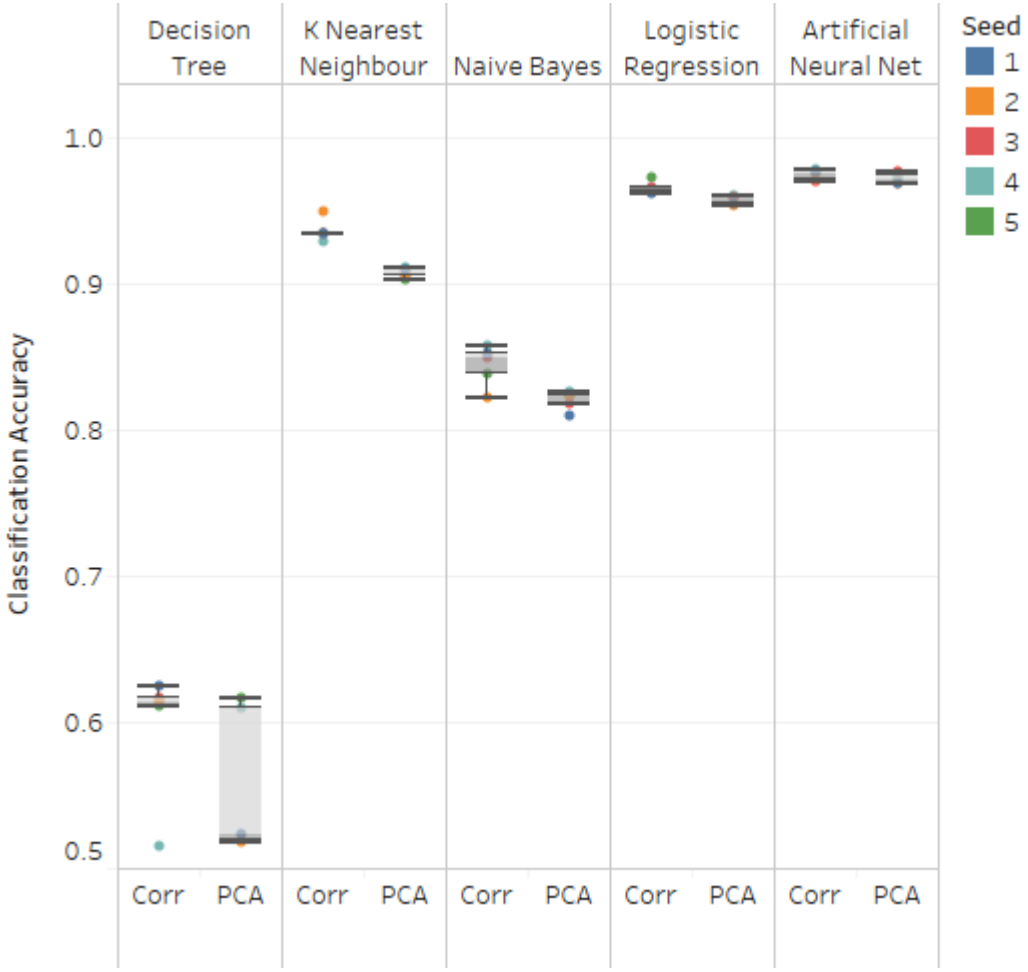

Figure 5.2: Boxplots of accuracies grouped by dimensionality reduction technique

5.4.1 Distribution of Accuracies

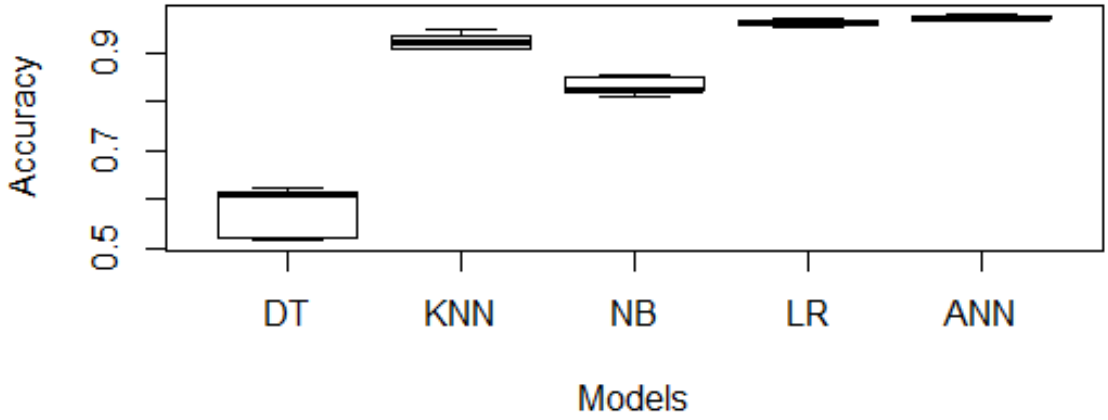

Figure 5.3: Distribution of accuracies grouped by modeling technique

The figure 5.3 illustrates the distribution of the accuracies for each of the classification algorithm. The accuracy distributions for ANN and LR seem to be quite normal with low standard deviation but the distributions of KNN and NB with the data from correlation analysis technique seems positively skewed. It is also evident that the distributions of the accuracies of the decision tree are not normal. To perform the significance tests of the model with better predictive accuracies for the data with is not normal; a non parametric test is the best solution. The Wilcoxon signed ranked test is the apt test to compare the results of two models for significance (Whitley & Ball, 2002).

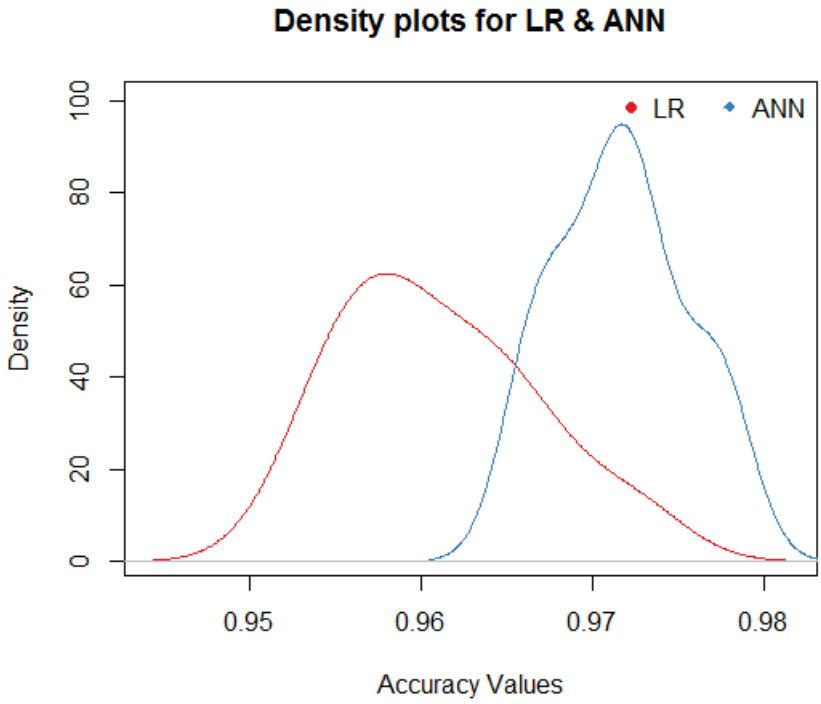

Figure 5.4: Density plot for ANN and LR models

The results report that the artificial neural net has slightly better classification accuracy over the logistic regression. The figure 5.4 illustrates the results of the ANN and the LR models. It can be observed that the results of these two test overlap. Hence, these two models were tested for statistical significance in order to prove that the results of the ANN algorithm are significant over the LR algorithm.

```
> wilcox.test(ann_Acc, lr_Acc, paired = TRUE)

        Wilcoxon signed rank test with continuity correction

data:  ann_Acc and lr_Acc
V = 55, p-value = 0.005889
alternative hypothesis: true location shift is not equal to 0
```

Figure 5.5: Wilcoxon test

The hypothesis of the Wilcoxon test is that the two accuracy lists are from an identical population and don't differ. The test resulted in a p value of 0.005889 which

is quite less than 0.05, so the hypothesis can be rejected. So at a significance level of 0.5, it can be concluded that the ANN accuracy is from a different population and can be treated as of greater classification accuracy.

5.5 Hypothesis Evaluation

The hypothesis of the current experiment is restated below –

H1 Each supervised machine learning algorithm modeled on the HAR dataset yields a different classification accuracy that is significantly greater than the benchmark SVM model, with a p value < 0.05.

Earlier in this chapter, the results for each of the machine learning models created were discussed. In addition, the statistical significance for two of the tests was also calculated to evaluate the fidelity of the results while comparing each other.

To analyze the results of the first hypothesis, the results of benchmark test and the current best performing algorithm, artificial neural net are tabulated in table 5.7.

Benchmark SVM Accuracy	Primary ANN Accuracy
89.3	0.9714934
89.0	0.9737609
	0.9727891
	0.9773243
	0.9773243

Table 5.7: Accuracy of Benchmark SVM and current ANN models

It must be noted that only the ANN results using the correlation analysis as the feature engineering technique is used for evaluation. From the table 5.7 it is evident

Model	Average Accuracy
Decision Tree	0.5960479
K Nearest Neighbors	0.9366375
Naive Bayes	0.8439261
Logistic Regression	0.9656625
Artificial Neural Network	**0.9745384**

Table 5.8: Average Accuracy List

that the current experiment results produced by ANN are significantly higher than the benchmark multiclass hardware friendly SVM model.

From the results tabulated in table 5.8 and the figure 5.3, it is evident that artificial neural network performed significantly better than all the other machine learning algorithms created in this experiment followed by the LR and KNN which managed to perform better than the benchmark test. However, the research hypothesis requires every model to perform significantly better than the base SVM model but only 3 of the models were able to do so. The DT and NB models failed in this case.

Hence, combining all the experiment and statistical test results, it can be stated that there isnt enough evidence to reject the null hypothesis.

5.6 Strengths & Limitations

One of the main strengths of the study is the low standard deviation in the accuracy values for ANN, LR and KNN models. These 3 models not only had accuracies higher than the benchmark but the variation in the results is quite small. This evidence suggests that the models have been developed with high robustness. It also signifies that these results can be replicated when necessary.

Additionally, all the models have shown greater understanding of inter cluster differences by effectively differentiating the stationary and mobile activities. This suggests that the models were able to capture the underlying concepts of the data in order to model it.

However, one of the important limitations of the study was that the significance value established for the study has a scope of 5% of error gradient. The study could have highly benefitted with the use of a smaller p value as 0.01.

Another limitation determined in the analysis is that most of the models werent able to capture the intra cluster differences as all the models failed in differentiating sitting and standing activities which fall under stationary activities.

5.7 Summary of the Analysis

This chapter has analyzed the working of the major experimental stages and the results achieved from each of them. Additionally, distributions of the results were studied along with their statistical significance values which in turn decided the answer for the research question. The contributions of the study were stated along with the strengths and limitations of the results.

Even though the Artificial neural net outperformed the benchmark SVM model and along with KNN and LR models, the alternate hypothesis failed to be accepted due to lack of sufficient evidence as the DT and NB algorithms had a significantly weaker performance. Detailed analysis was performed on the implementation of each of the models and the results of the models were also carefully studied to understand the success or failure of the model when compared to the expectations of the model as stated in the previous chapters.

It was observed that the correlation analysis worked well in performing dimension-

ality reduction by not losing any important information that could help train better. But the principal component analysis performed poorly and resulted in decreased accuracy for all the algorithms. Performing two feature engineering techniques helped in gaining good insights on the overall data and its composition. Evaluation technique also proved quite beneficial for the study in establishing sufficient significance in stating the results.

The next chapter concludes the study and also suggests improvements and scope for future research in this regard.

Chapter 6

Conclusion

This chapter performs a review of the current study. It reiterates the research question and all the different stages involved in answering it. The objectives of the research and all the important phases are quickly walked through. Additionally, the contributions of the research are also stated. The chapter concludes by highlighting the areas of further research.

6.1 Research Overview

Primarily, the research aimed to recognize the physical human activity of a user using the data generated from a wearable sensor device. The initial study was concentrated on understanding the current state of the art techniques in performing human activity recognition.

After acquiring relevant data for the research, the next main area of focus in the research was to perform suitable feature engineering to alter the data to be compliant to apply machine learning algorithms in the restricted time and memory conditions. Correlation analysis was the first technique which calculates correlation coefficients

between each of the independent features. It is a feature reduction technique which ensures to eliminate the features that highly correlate with each other. Principal component analysis was the other method which was used to create latent features by capturing the underlying variance of the target feature. Both these techniques can be proved effective for the experiment as they minimize the total number of feature to about 30% of the original features.

The goal of the research was to model the condensed data using machine learning algorithms in order to obtain high predictive accuracies.

6.2 Problem Definition

Based on the literature review, a gap in the current body of knowledge was exposed. Multiple studies on HAR have utilized the basic algorithms from the various families of machine learning and resulted in high accuracy values, however the set of 5 algorithms were never used against feature reduction techniques to perform classification of activities on the dataset under inspection. The research work sought to empirically determine the better classification algorithm out of the 5 popular ones using either of the feature reduction techniques. The research question investigated in the study is -

> *To what extent can supervised machine learning algorithms significantly enhance the recognition of physical human activity with inertial sensor data when compared to SVM base models?*

*Algorithms : K-Nearest Neighbors, Decision Tree, Naive Bayes, Multinomial Logistic Regression and Artificial Neural Network.

6.3 Workflow

The following depict the stages followed as an aim to answer the research question –

1. Performed extensive study on the existing literature of Human Activity Recognition.

   Gaps have been identified in the research domain.

2. A solution was designed to address the gaps in the Human Activity Recognition research.

   Primary motive of the design was to maximize the accuracy of the detection process.

3. The solution was implemented primarily based on the design.

   Additionally, occasional tweaking was performed where the solution commanded.

4. Induced models are evaluated using RRSS cross validation technique. With primary focus on classification accuracy.

5. Future areas of research are identified to extend the field of study.

   Multiple recommendations on the study have also been made

By efficiently exploiting the various aspects of the knowledge acquired from 1, the solution built consists of feature reduction techniques identified from 1 to be utilized. On the reduced dataset, an algorithm from each of the family of machine learning algorithms as decision tree, K nearest neighbor, Naive bayes, logistic regression and artificial neural network are induced to generate the models for stage 4. The evaluation metric chosen from stage 1 is the Classification accuracy as it provides a comprehensive insight of the performance. The ANN performed the best in overall accuracy levels of 97% averaged for both the feature engineering techniques. The second best algorithm is the logistic regression with an average of 96% followed by KNN algorithm with

92%. These three algorithms have managed to perform better than the benchmark multiclass hardware friendly SVM. The Naive bayes resulted in comparable accuracy of 83% which is similar to that of the benchmark. But the worst performance was by the decision tree with an unsatisfactory 57%. In the end, the logistic regression and ANN were tested for statistical significance which resulted in a p-value of 0.0058 demonstrating that the results were statistically significant. Additionally, the results from the PCA feature engineering technique were considerably poor compared to that of the correlation analysis.

6.4 Contributions & Impact

The primary contributions to the body of knowledge from the current research are as follows -

- Comprehensive literature review was performed which can help readers in understanding the domain and the current state of the art techniques in Human Activity Recognition.
- Systematic empirical investigation of the quantitative properties of the Human Activity Recognition dataset was performed. This can be used as a baseline for future solutions uding this dataset.
- Demonstrated that ANN performs best in specific cases of Human Activity Recognition
- Demonstrated that Artificial Neural Network, Multinomial Logistic Regression and K-Nearest Neighbor algorithms can be used to model Human Activity Recognition
- Demonstrated that Correlation analysis can yield better results compared to PCA in the context of Human Activity Recognition

6.5 Future Work & Recommendations

There are multiple ways in which the study could have been manipulated which subsequently could have yield in completely different insights been discovered.

The decision tree was modeled using the recursive partitioning method. There are plenty of other methods in decision trees that may be utilized to get better results. The splitting criteria provided in the current study is Gini impurity, additionally, information gain may also be used. For the KNN algorithm, the tune length was used a 7 as it was seen that the increase in the K value only decreased the performance. However, it is not evident if the final k value of K was the local minima or the global minima of convergence. So a higher value of tune length could have helped resolve the ambiguity. For NB classifier, the laplacian correction and bandwidth adjustment values can be manipulated to observe the changes. For logistic regression, the study utilized a penalized multinomial regression and there are multiple other models as bagged, boosted, regularized or ordered logistic regression. All the models have a different protocol and tuning parameters which may be exploited.

In addition to PCA and correlation analysis, exploratory factor analysis can also be tested as a dimensionality reduction technique. For evaluation, it was advised that K fold cross validation can also be used for better confirmation of results. However, due to time constraints, it wasn't performed.

Another interesting aspect of investigation would be to divide the data into train and test with respect to users. The models can be trained and evaluated exactly as the current experiment to observe if the models can generalize to unknown users.

References

Abowd, D., Dey, A. K., Orr, R., & Brotherton, J. (1998, Sep 01). Context-awareness in wearable and ubiquitous computing. *Virtual Reality*, *3*(3), 200–211. Retrieved from `https://doi.org/10.1007/BF01408562` doi: 10.1007/BF01408562

Alpaydin, E. (2014). *Introduction to machine learning.* MIT press.

Altun, K., & Barshan, B. (2010). Human activity recognition using inertial/magnetic sensor units. In *International workshop on human behavior understanding* (pp. 38–51).

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living* (pp. 216–223).

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *Esann.*

Ashbrook, D., & Starner, T. (2003). Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous computing*, *7*(5), 275–286.

Attal, F., Mohammed, S., Dedabrishvili, M., Chamroukhi, F., Oukhellou, L., & Amirat, Y. (2015). Physical human activity recognition using wearable sensors. *Sensors*, *15*(12), 31314–31338.

Bachlin, M., Roggen, D., Troster, G., Plotnik, M., Inbar, N., Meidan, I., ... others (2009). Potentials of enhanced context awareness in wearable assistants for parkinson's disease patients with the freezing of gait syndrome. In *Wearable computers, 2009. iswc'09. international symposium on* (pp. 123–130).

Bao, L., & Intille, S. (2004). Activity recognition from user-annotated acceleration data. *Pervasive computing*, 1–17.

Bellman, R. (2013). *Dynamic programming*. Courier Corporation.

Bieber, G., Luthardt, A., Peter, C., & Urban, B. (2011). The hearing trousers pocket: activity recognition by alternative sensors. In *Proceedings of the 4th international conference on pervasive technologies related to assistive environments* (p. 44).

bin Abdullah, M. F. A., Negara, A. F. P., Sayeed, M. S., Choi, D.-J., & Muthu, K. S. (2012). Classification algorithms in human activity recognition using smartphones. *International Journal of Computer and Information Engineering*, *6*, 77–84.

Blechner, M. D. (2005). Behavior of various machine learning models in the face of noisy data. *Harvard-MIT Division of Health Sciences and Technology, J. Medical Decision Support, Final project*.

Bonomi, A. G., Goris, A., Yin, B., Westerterp, K. R., et al. (2009). Detection of type, duration, and intensity of physical activity using an accelerometer. *Med Sci Sports Exerc*, *41*(9), 1770–1777.

Brezmes, T., Gorricho, J.-L., & Cotrina, J. (2009). Activity recognition from accelerometer data on a mobile phone. *Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living*, 796–799.

Bulling, A., Blanke, U., & Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, *46*(3), 33.

Chen, J., Kwong, K., Chang, D., Luk, J., & Bajcsy, R. (2006). Wearable sensors for reliable fall detection. In *Engineering in medicine and biology society, 2005. ieee-embs 2005. 27th annual international conference of the* (pp. 3551–3554).

Chen, Y., Guo, M., & Wang, Z. (2016). An improved algorithm for human activity recognition using wearable sensors. In *Advanced computational intelligence (icaci), 2016 eighth international conference on* (pp. 248–252).

Cleland, I., Kikhia, B., Nugent, C., Boytsov, A., Hallberg, J., Synnes, K., ... Finlay, D. (2013). Optimal placement of accelerometers for the detection of everyday activities. *Sensors*, *13*(7), 9183–9200.

Colaprico, A., Cava, C., Bertoli, G., Bontempi, G., & Castiglioni, I. (2015). Integrative analysis with monte carlo cross-validation reveals mirnas regulating pathways cross-talk in aggressive breast cancer. *BioMed research international*, *2015*.

Croarkin, C., Tobias, P., Filliben, J., et al. (2006). Nist/sematech e-handbook of statistical methods. *NIST/SEMATECH, July. Available online: http://www. itl. nist. gov/div898/handbook*.

De Oliveira, R., Cherubini, M., & Oliver, N. (2010). Movipill: improving medication compliance for elders using a mobile persuasive social game. In *Proceedings of the 12th acm international conference on ubiquitous computing* (pp. 251–260).

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, *10*(7), 1895–1923.

D. Lara, O., & Labrador, M. (2013, 01). A survey on human activity recognition using wearable sensors. , *15*, 1192-1209.

Everitt, B. S. (1992). *The analysis of contingency tables*. CRC Press.

Fan, L., Wang, Z., & Wang, H. (2013). Human activity recognition model based on decision tree. In *Advanced cloud and big data (cbd), 2013 international conference on* (pp. 64–68).

Fawcett, T. (2003). Graphs: notes and practical considerations for data mining researchers. *Tech Reports HPL-2003-4. HP Laboratories, Palo Alto, CA*.

Florentino-Liano, B., O'Mahony, N., & Artés-Rodríguez, A. (2012). Human activity recognition using inertial sensors with invariance to sensor orientation. In *Cognitive information processing (cip), 2012 3rd international workshop on* (pp. 1–6).

Golding, A. R., & Lesh, N. (1999). Indoor navigation using a diverse set of cheap, wearable sensors. In *Wearable computers, 1999. digest of papers. the third international symposium on* (pp. 29–36).

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics*, *11*(1), 1–21.

Gupta, P., & Dallas, T. (2014). Feature selection and activity recognition system using a single triaxial accelerometer. *IEEE Transactions on Biomedical Engineering*, *61*(6), 1780–1786.

Harms, H., Amft, O., Tröster, G., & Roggen, D. (2008). Smash: A distributed sensing and processing garment for the classification of upper body postures. In *Proceedings of the icst 3rd international conference on body area networks* (p. 22).

Hu, Y.-J., Yu, M.-C., Wang, H.-A., & Ting, Z.-Y. (2015). A similarity-based learning algorithm using distance transformation. *IEEE Transactions on Knowledge and Data Engineering*, *27*(6), 1452–1464.

Kaghyan, S., & Sarukhanyan, H. (2013). Accelerometer and gps sensor combination based system for human activity recognition. In *Computer science and information technologies (csit), 2013* (pp. 1–9).

Katz, S., Downs, T. D., Cash, H. R., & Grotz, R. C. (1970). Progress in development of the index of adl. *The gerontologist*, *10*(1_Part_1), 20–30.

Kelleher, J. D., Mac Namee, B., & D'Arcy, A. (2015). *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press.

Khan, A. M., Lee, Y.-K., & Kim, T.-S. (2008). Accelerometer signal-based human activity recognition using augmented autoregressive model coefficients and artificial neural nets. In *Engineering in medicine and biology society, 2008. embs 2008. 30th annual international conference of the ieee* (pp. 5172–5175).

Kohavi, R., et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (Vol. 14, pp. 1137–1145).

Krumm, J., & Horvitz, E. (2006). Predestination: Inferring destinations from partial trajectories. *UbiComp 2006: Ubiquitous Computing*, 243–260.

Kunze, K., Barry, M., Heinz, E. A., Lukowicz, P., Majoe, D., & Gutknecht, J. (2006). Towards recognizing tai chi-an initial experiment using wearable sensors. In *Applied wearable computing (ifawc), 2006 3rd international forum on* (pp. 1–6).

Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, *12*(2), 74–82.

Ladha, C., Hammerla, N. Y., Olivier, P., & Plötz, T. (2013). Climbax: skill assessment for climbing enthusiasts. In *Proceedings of the 2013 acm international joint conference on pervasive and ubiquitous computing* (pp. 235–244).

Lazzarini, V. (2007). Eduardo r. miranda and marcelo m. wanderley: New digital musical instruments: Control and interaction beyond the keyboard. *Computer Music Journal*, *31*(4), 75-77. Retrieved from `http://dx.doi.org/10.1162/comj.2007.31.4.75` doi: 10.1162/comj.2007.31.4.75

Leonhardt, U., & Magee, J. (1998). Multi-sensor location tracking. In *Proceedings of the 4th annual acm/ieee international conference on mobile computing and networking* (pp. 203–214).

Lester, J., Choudhury, T., & Borriello, G. (2006). A practical approach to recognizing physical activities. *Pervasive Computing*, 1–16.

Liao, L., Patterson, D. J., Fox, D., & Kautz, H. (2007). Learning and inferring transportation routines. *Artificial Intelligence*, *171*(5-6), 311–331.

Liberty, E., Lang, K., & Shmakov, K. (2016). Stratified sampling meets machine learning. In *International conference on machine learning* (pp. 2320–2329).

Logan, B., Healey, J., Philipose, M., Tapia, E. M., & Intille, S. (2007). A long-term evaluation of sensing modalities for activity recognition. In *International conference on ubiquitous computing* (pp. 483–500).

Longo, L. (2011). Human-computer interaction and human mental workload: Assessing cognitive engagement in the world wide web. In *Proceedings of the 13th ifip tc 13 international conference on human-computer interaction - volume part iv* (pp. 402–405). Berlin, Heidelberg: Springer-Verlag. Retrieved from `http://dl.acm.org/citation.cfm?id=2042283.2042335`

Longo, L. (2012). Formalising human mental workload as non-monotonic concept for adaptive and personalised web-design. In J. Masthoff, B. Mobasher, M. C. Desmarais, & R. Nkambou (Eds.), *User modeling, adaptation, and personalization: 20th international conference, umap 2012, montreal, canada, july 16-20, 2012. proceedings* (pp. 369–373). Berlin, Heidelberg: Springer Berlin Heidelberg.

Longo, L. (2015, June). Designing medical interactive systems via assessment of human mental workload. In *2015 ieee 28th international symposium on computer-based medical systems* (p. 364-365). doi: 10.1109/CBMS.2015.67

Longo, L. (2016, June). Mental workload in medicine: Foundations, applications, open problems, challenges and future perspectives. In *2016 ieee 29th international symposium on computer-based medical systems (cbms)* (p. 106-111). doi: 10.1109/CBMS.2016.36

Longo, L. (2017). Subjective usability, mental workload assessments and their impact on objective human performance. In R. Bernhaupt, G. Dalvi, A. Joshi, D. K. Balkrishan, J. O'Neill, & M. Winckler (Eds.), *Human-computer interaction - interact 2017: 16th ifip tc 13 international conference, mumbai, india, september 25-29, 2017, proceedings, part ii* (pp. 202–223). Cham: Springer International Publishing.

Mannini, A., Intille, S. S., Rosenberger, M., Sabatini, A. M., & Haskell, W. (2013). Activity recognition using a single accelerometer placed at the wrist or ankle. *Medicine and science in sports and exercise*, *45*(11), 2193.

Mannini, A., & Sabatini, A. M. (2010). Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, *10*(2), 1154–1175.

Mathie, M., Celler, B. G., Lovell, N. H., & Coster, A. (2004). Classification of basic daily movements using a triaxial accelerometer. *Medical and Biological Engineering and Computing*, *42*(5), 679–687.

Maurtua, I., Kirisci, P. T., Stiefmeier, T., Sbodio, M. L., & Witt, H. (2007). A wearable computing prototype for supporting training activities in automotive production. In *Applied wearable computing (ifawc), 2007 4th international forum on* (pp. 1–12).

Mi-hee, L., Jungchae, K., Kwangsoo, K., Inho, L., Ha Jee, S., & Kook Yoo, S. (2009, 10). Physical activity recognition using a single tri-axis accelerometer. , *2178*.

Minnen, D., Starner, T., Essa, I., & Isbell, C. (2006). Discovering characteristic actions from on-body sensor data. In *Wearable computers, 2006 10th ieee international symposium on* (pp. 11–18).

Mitra, S., & Acharya, T. (2007, May). Gesture recognition: A survey. *Trans. Sys. Man Cyber Part C*, *37*(3), 311–324. Retrieved from `http://dx.doi.org/10.1109/TSMCC.2007.893280` doi: 10.1109/TSMCC.2007.893280

Moustafa, K., Luz, S., & Longo, L. (2017). Assessment of mental workload: A comparison of machine learning methods and subjective assessment techniques. In L. Longo & M. C. Leva (Eds.), *Human mental workload: Models and applications: First international symposium, h-workload 2017, dublin, ireland, june 28-30, 2017, revised selected papers* (pp. 30–50). Cham: Springer International Publishing.

Mozer, M. C. (1998). The neural network house: An environment hat adapts to its inhabitants. In *Proc. aaai spring symp. intelligent environments* (Vol. 58).

Oh, K., Park, H.-S., & Cho, S.-B. (2010). A mobile context sharing system using activity and emotion recognition with bayesian networks. In *Ubiquitous intelligence & computing and 7th international conference on autonomic & trusted computing (uic/atc), 2010 7th international conference on* (pp. 244–249).

Oliver, N., & Flores-Mangas, F. (2007). Healthgear: Automatic sleep apnea detection and monitoring with a mobile phone. *JCM*, *2*(2), 1–9.

Oniga, S., & Suto, J. (2014). Human activity recognition using neural networks. In *Control conference (iccc), 2014 15th international carpathian* (pp. 403–406).

Parkka, J., Ermes, M., Korpipaa, P., Mantyjarvi, J., Peltola, J., & Korhonen, I. (2006). Activity classification using realistic data from wearable sensors. *IEEE Transactions on information technology in biomedicine*, *10*(1), 119–128.

Patterson, D. J., Liao, L., Fox, D., & Kautz, H. (2003). Inferring high-level behavior from low-level sensors. In *Ubicomp* (pp. 73–89).

Paul, P., & George, T. (2015). An effective approach for human activity recognition on smartphone. In *Engineering and technology (icetech), 2015 ieee international conference on* (pp. 1–3).

Piatetsky, G. (2014). Crisp-dm, still the top methodology for analytics, data mining, or data science projects. *KDD News*.

Powell, H., Hanson, M. A., & Lach, J. (2007). A wearable inertial sensing technology for clinical assessment of tremor. In *Biomedical circuits and systems conference, 2007. biocas 2007. ieee* (pp. 9–12).

Qin, C., Bao, X., Choudhury, R. R., & Nelakuditi, S. (2014). Tagsense: Leveraging smartphones for automatic image tagging. *IEEE Transactions on Mobile Computing*, *13*(1), 61–74.

Ravi, N., Dandekar, N., Mysore, P., & Littman, M. L. (2005). Activity recognition from accelerometer data. In *Aaai* (Vol. 5, pp. 1541–1546).

Rizzo, L., Dondio, P., Delany, S. J., & Longo, L. (2016). Modeling mental workload via rule-based expert system: A comparison with nasa-tlx and workload profile. In L. Iliadis & I. Maglogiannis (Eds.), *Artificial intelligence applications and innovations: 12th ifip wg 12.5 international conference and workshops, aiai 2016, thessaloniki, greece, september 16-18, 2016, proceedings* (pp. 215–229). Cham: Springer International Publishing.

Ronao, C. A., & Cho, S.-B. (2014). Human activity recognition using smartphone sensors with two-stage continuous hidden markov models. In *Natural computation (icnc), 2014 10th international conference on* (pp. 681–686).

Sabatini, A. M., Martelloni, C., Scapellato, S., & Cavallo, F. (2005). Assessment of walking features from foot inertial sensing. *IEEE Transactions on biomedical engineering*, *52*(3), 486–494.

San-Segundo, R., Cordoba, R., Ferreiros, J., & D'Haro-Enríquez, L. F. (2016). Frequency features and gmm-ubm approach for gait-based person identification using smartphone inertial signals. *Pattern Recognition Letters*, *73*, 60–67.

San-Segundo, R., Echeverry-Correa, J. D., Salamea-Palacios, C., Lutfi, S. L., & Pardo, J. M. (2017). I-vector analysis for gait-based person identification using smartphone inertial signals. *Pervasive and Mobile Computing*, *38*, 140–153.

San-Segundo, R., Lorenzo-Trueba, J., Martínez-González, B., & Pardo, J. M. (2016). Segmenting human activities based on hmms using smartphone inertial sensors. *Pervasive and Mobile Computing*, *30*, 84–96.

Sarkar, A. J., Lee, Y.-K., & Lee, S. (2010). A smoothed naive bayes-based classifier for activity recognition. *IETE Technical Review*, *27*(2), 107–119.

Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., ... Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, *56*(1), 116–124.

Sondhi, P. (2009). Feature construction methods: a survey. *sifaka. cs. uiuc. edu*, *69*, 70–71.

Starner, T., Weaver, J., & Pentland, A. (1997). A wearable computer-based american sign language recogniser. *Personal Technologies*, *1*(4), 241–250.

Stiefmeier, T., Roggen, D., Ogris, G., Lukowicz, P., & Tröster, G. (2008). Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing*, *7*(2).

Sucerquia, A., López, J. D., & Vargas-Bonilla, J. F. (2017). Sisfall: a fall and movement dataset. *Sensors*, *17*(1), 198.

Sung, M., DeVaul, R., Jimenez, S., Gips, J., & Pentland, A. (2004). Shiver motion and core body temperature classification for wearable soldier health monitoring systems. In *Wearable computers, 2004. iswc 2004. eighth international symposium on* (Vol. 1, pp. 192–193).

Tapia, E. M., Intille, S. S., & Larson, K. (2004). Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive* (Vol. 4, pp. 158–175).

Taylor, R. (1990). Interpretation of the correlation coefficient: A basic review. *Journal of Diagnostic Medical Sonography*, *6*(1), 35-39. Retrieved from `http://dx.doi.org/10.1177/875647939000600106` doi: 10.1177/875647939000600106

Tessendorf, B., Bulling, A., Roggen, D., Stiefmeier, T., Feilner, M., Derleth, P., & Tröster, G. (2011). Recognition of hearing needs from body and eye movements to improve hearing instruments. *Pervasive Computing*, 314–331.

Turaga, P., Chellappa, R., Subrahmanian, V. S., & Udrea, O. (2008, November). Machine recognition of human activities: A survey. *IEEE Trans. Cir. and Sys. for Video Technol.*, *18*(11), 1473–1488. Retrieved from `http://dx.doi.org/10.1109/TCSVT.2008.2005594` doi: 10.1109/TCSVT.2008.2005594

Wan, D. (1999). Magic medicine cabinet: A situated portal for consumer healthcare. In *International symposium on handheld and ubiquitous computing* (pp. 352–355).

Ward, A., Jones, A., & Hopper, A. (1997). A new location technique for the active office. *IEEE Personal communications*, *4*(5), 42–47.

Whitley, E., & Ball, J. (2002). Statistics review 6: Nonparametric methods. *Critical care*, *6*(6), 509.

Winters, J. M., Wang, Y., & Winters, J. M. (2003). Wearable sensors and telerehabilitation. *IEEE Engineering in Medicine and Biology Magazine*, *22*(3), 56–65.

Wirth, R., & Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining* (pp. 29–39).

Wolpert, D. H. (2002). The supervised learning no-free-lunch theorems. In *Soft computing and industry* (pp. 25–42). Springer.

Wren, C. R., & Tapia, E. M. (2006). Toward scalable activity recognition for sensor networks. In *Loca* (Vol. 3987, pp. 168–185).

Wu, G., & Xue, S. (2008). Portable preimpact fall detector with inertial sensors. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *16*(2), 178–183.

Yang, C.-C., & Hsu, Y.-L. (2010). A review of accelerometry-based wearable motion detectors for physical activity monitoring. *Sensors*, *10*(8), 7772–7788.

Yang, S.-I., & Cho, S.-B. (2008). Recognizing human activities from accelerometer and physiological sensors. In *Multisensor fusion and integration for intelligent systems, 2008. mfi 2008. ieee international conference on* (pp. 100–105).

Yeoh, W.-S., Pek, I., Yong, Y.-H., Chen, X., & Waluyo, A. B. (2008). Ambulatory monitoring of human posture and walking speed using wearable accelerometer sensors. In *Engineering in medicine and biology society, 2008. embs 2008. 30th annual international conference of the ieee* (pp. 5184–5187).

Zhu, X., & Qiu, H. (2016). High accuracy human activity recognition based on sparse locality preserving projections. *PloS one*, *11*(11), e0166567.

Zhu, X., & Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, *22*(3), 177–210.

Appendix A

Experiment Implementation

```
1
2  #R Code
3  #Clear Global Environment
4  rm(list=ls(all=TRUE))
5
6  ######-------------------Data Integration-------------------------##
7  #1. Data Integration
8  dataFile = "dataset.RData"
9
10 setwd("C:/Users/Haritha Vellam/Desktop/DA Sem 2/Research
11 Design/activity_recognition/UCI HAR Dataset")
12
13 if (!file.exists(dataFile)) {
14   temp = read.table("activity_labels.txt", sep = "")
15   activityLabels = as.character(temp$V2)
16   temp = read.table("features.txt", sep = "")
17   attributeNames = temp$V2
18
19   Xtrain = read.table("train/X_train.txt", sep = "")
20   names(Xtrain) = attributeNames
21   Ytrain = read.table("train/y_train.txt", sep = "")
22   names(Ytrain) = "Activity"
23   Ytrain$Activity = as.factor(Ytrain$Activity)
24   levels(Ytrain$Activity) = activityLabels
25   trainSubjects = read.table("train/subject_train.txt",
26   sep = "")
27   names(trainSubjects) = "subject"
28   trainSubjects$subject = as.factor(trainSubjects$subject)
```

```
  traindata = cbind(Xtrain, trainSubjects, Ytrain)

  Xtest = read.table("test/X_test.txt", sep = "")
  names(Xtest) = attributeNames
  Ytest = read.table("test/y_test.txt", sep = "")
  names(Ytest) = "Activity"
  Ytest$Activity = as.factor(Ytest$Activity)
  levels(Ytest$Activity) = activityLabels
  testSubjects = read.table("test/subject_test.txt",
  sep = "")
  names(testSubjects) = "subject"
  testSubjects$subject = as.factor(testSubjects$subject)
  testdata = cbind(Xtest, testSubjects, Ytest)

  save(traindata, testdata, file = dataFile)
  rm(train, test, temp, Ytrain, Ytest, Xtrain, Xtest, trainSubjects, testSubjects,
      activityLabels, attributeNames)
}

##--------------------Data Input-------------------------##
#2. Data Input & merger
load(dataFile)
fulldata <- rbind(train, test)
rm(train, test, dataFile)

##-------------------Data
Exploration-------------------------##
#3. Data Exploration
#Dimensions
dim(fulldata)

#Feature names
names(fulldata)

#Structure
str(fulldata, list.len=ncol(fulldata))

#Attributes
attributes(fulldata)

#First 2 rows
head(fulldata, 2)

#Distribution of features
summary(fulldata[, 560:563])
```

```
73
74 #Frequency
75 #Target
76 table(fulldata$Activity)
77 pie(table(fulldata$Activity))
78 barplot(table(fulldata$Activity), cex.names = 0.6,
79 space = 03)
80 #Subject
81 table(fulldata$subject)
82 barplot(table(fulldata$subject))
83
84 #Range
85 max(fulldata[, 1:561], na.rm = FALSE)
86 min(fulldata[, 1:561], na.rm = FALSE)
87
88 #Missing values
89 sum(is.na(fulldata))
90
91 ##-------------------Data
92 Preparation-------------------------##
93
94 #Factorization
95 fulldata$subject <- as.factor(fulldata$subject)
96 fulldata$Activity <- as.factor(fulldata$Activity)
97 fulldata[, 1:561] <- as.data.frame(lapply(fulldata[,
98 1:561], as.numeric))
99
100 #Syntactically correct character names
101 names(fulldata) <- gsub("()", "", names(fulldata),
102 fixed = T)
103 names(fulldata) <- gsub("-", "_", names(fulldata),
104 fixed = T)
105 names(fulldata) <- gsub(",", "_", names(fulldata),
106 fixed = T)
107
108 #########Dimensionality Reduction###############
109 #Method 1
110 #Correlated features
111 library(caret)
112 FullCorrelationMatrix <- cor(sapply(fulldata[,-c(562,563)],
113 as.numeric))
114 #write.csv(FullCorrelationMatrix, file = "CorrMat.csv")
115 plot(FullCorrelationMatrix)
116 dim(FullCorrelationMatrix)
117 table(round(FullCorrelationMatrix, digits = 1))
```

```
118
119 HighCorrelatedList  <-
120 findCorrelation(na.omit(FullCorrelationMatrix), cutoff =
121 0.8, names = T, exact = T)
122 length(HighCorrelatedList)
123 Uncorrelated <- fulldata[, !colnames(fulldata) %in%
124 HighCorrelatedList]
125 Uncorrelated <- subset(Uncorrelated, select = -c(subject))
126 dim(Uncorrelated)
127 rm(FullCorrelationMatrix, HighCorrelatedList,
128 HighCorrelatedList1,
129    HighCorrelatedList2, HighCorrelatedList3)
130
131
132 ###Method 2
133 #PCA
134 PCA_comps <- prcomp(fulldata[,-c(562,563)],scale. = TRUE)
135 summary(PCA_comps)
136 dim(PCA_comps$x)
137 PCA_comps$rotation[1:5,1:5]
138 PCA_var <- PCA_comps$sdev^2
139 head(PCA_var, 20)
140 PCA_Prop_Var <- PCA_var/ncol(fulldata[,-c(562,563)])
141 head(PCA_Prop_Var, 10)*100
142 sum(head(PCA_Prop_Var, 100)*100)
143
144 plot(PCA_Prop_Var[1:100], xlab = "PC",ylab = "Proportion")
145 plot(cumsum(PCA_Prop_Var), xlab = "Principal
146 Components",ylab = "Proportion of Variance captured")
147 abline(h=0.95)
148 abline(v=100)
149
150 PCA_Fulldata <- data.frame(fulldata$Activity ,
151 PCA_comps$x[, 1:100])
152 colnames(PCA_Fulldata)[1] <- 'Activity'
153
154
155 ######Modeling & Evaluation #################
156 #Create Seeds
157 set.seed(1)
158 trainCount_s1 <- createDataPartition(y = fulldata$Activity,
159 p = 0.70, list = FALSE)
160
161 set.seed(2)
162 trainCount_s2 <- createDataPartition(y = fulldata$Activity,
```

```
p = 0.70, list = FALSE)

set.seed(3)
trainCount_s3 <- createDataPartition(y = fulldata$Activity,
p = 0.70, list = FALSE)

set.seed(4)
trainCount_s4 <- createDataPartition(y = fulldata$Activity,
p = 0.70, list = FALSE)

set.seed(5)
trainCount_s5 <- createDataPartition(y = fulldata$Activity,
p = 0.70, list = FALSE)

#Prepare Train and test data
#Method 1 - Uncorrelated Data
#seed1 Data
trainUncorr_s1 <- Uncorrelated[trainCount_s1,]
testUncorr_s1 <- Uncorrelated[-trainCount_s1,]

#seed2 Data
trainUncorr_s2 <- Uncorrelated[trainCount_s2,]
testUncorr_s2 <- Uncorrelated[-trainCount_s2,]

#seed3 Data
trainUncorr_s3 <- Uncorrelated[trainCount_s3,]
testUncorr_s3 <- Uncorrelated[-trainCount_s3,]

#seed4 Data
trainUncorr_s4 <- Uncorrelated[trainCount_s4,]
testUncorr_s4 <- Uncorrelated[-trainCount_s4,]

#seed5 Data
trainUncorr_s5 <- Uncorrelated[trainCount_s5,]
testUncorr_s5 <- Uncorrelated[-trainCount_s5,]

#Method 2 - PCA Data
#seed1 Data
trainPCA_s1 <- PCA_Fulldata[trainCount_s1,]
testPCA_s1 <- PCA_Fulldata[-trainCount_s1,]

#seed2 Data
trainPCA_s2 <- PCA_Fulldata[trainCount_s2,]
testPCA_s2 <- PCA_Fulldata[-trainCount_s2,]

```

```
#seed3 Data
trainPCA_s3 <- PCA_Fulldata[trainCount_s3,]
testPCA_s3 <- PCA_Fulldata[-trainCount_s3,]

#seed4 Data
trainPCA_s4 <- PCA_Fulldata[trainCount_s4,]
testPCA_s4 <- PCA_Fulldata[-trainCount_s4,]

#seed5 Data
trainPCA_s5 <- PCA_Fulldata[trainCount_s5,]
testPCA_s5 <- PCA_Fulldata[-trainCount_s5,]

rm(trainCount_s1, trainCount_s2, trainCount_s3,
   trainCount_s4,trainCount_s5)


#Final Modeling
#Decision Tree
library(rpart)
library(plyr)
library(caret)
#Gini is to minimize misclassification and for continuous
attributes
#and is faster

#Method 1
dt_Uncorr_s1 <-
train(x=trainUncorr_s1[,-173],y=trainUncorr_s1[,173],
                      method = 'rpart',trControl =
                      trainControl(method="cv", number =
                      2),
                      parms=list(split='gini'))
summary(dt_Uncorr_s1)

dt_Uncorr_s2 <-
train(x=trainUncorr_s2[,-173],y=trainUncorr_s2[,173],
                      method = 'rpart',trControl =
                      trainControl(method="cv", number =
                      2),
                      parms=list(split='gini'))

dt_Uncorr_s3 <-
train(x=trainUncorr_s3[,-173],y=trainUncorr_s3[,173],
                      method = 'rpart',trControl =
                      trainControl(method="cv", number =
```

```
                      2),

                      parms=list(split='gini'))

dt_Uncorr_s4 <-
train(x=trainUncorr_s4[,-173],y=trainUncorr_s4[,173],
                      method = 'rpart',trControl =
                      trainControl(method="cv", number =
                      2),

                      parms=list(split='gini'))

dt_Uncorr_s5 <-
train(x=trainUncorr_s5[,-173],y=trainUncorr_s5[,173],
                      method = 'rpart',trControl =
                      trainControl(method="cv", number =
                      2),

                      parms=list(split='gini'))

#Method 2
dt_PCA_s1 <- train(x = trainPCA_s1[,-1], y =
trainPCA_s1[,1],
                   method = 'rpart',trControl =
                   trainControl(method="cv", number = 2),
                   parms=list(split='gini'))

dt_PCA_s2 <- train(x = trainPCA_s2[,-1], y =
trainPCA_s2[,1],
                   method = 'rpart',trControl =
                   trainControl(method="cv", number = 2),
                   parms=list(split='gini'))

dt_PCA_s3 <- train(x = trainPCA_s3[,-1], y =
trainPCA_s3[,1],
                   method = 'rpart',trControl =
                   trainControl(method="cv", number = 2),
                   parms=list(split='gini'))

dt_PCA_s4 <- train(x = trainPCA_s4[,-1], y =
trainPCA_s4[,1],
                   method = 'rpart',trControl =
                   trainControl(method="cv", number = 2),
                   parms=list(split='gini'))

```

```
dt_PCA_s5 <- train(x = trainPCA_s5[,-1], y =
trainPCA_s5[,1],
                   method = 'rpart', trControl =
                   trainControl(method="cv", number = 2),
                   parms=list(split='gini'))

##KNN
#Method 1
knn_Uncorr_s1 <- train(Activity ~ ., data = trainUncorr_s1,

                       method = "knn",
                       trControl =
                       trainControl(method="cv", number =
                       2),
                       #preProcess = c("center","scale"),
                       tuneLength = 7)
summary(knn_Uncorr_s1)

knn_Uncorr_s2 <- train(Activity ~ ., data = trainUncorr_s2,
                       method = "knn",
                       trControl =
                       trainControl(method="cv", number =2,
                       #preProcess = c("center","scale"),
                       tuneLength = 7)

knn_Uncorr_s3 <- train(Activity ~ ., data =
trainUncorr_s3,method = "knn",
trControl = trainControl(method="cv", number = 2),
                       preProcess = c("center","scale"),
                       tuneLength = 7)

knn_Uncorr_s4 <- train(Activity ~ ., data = trainUncorr_s4,

                       method = "knn",

                       trControl =
                       trainControl(method="cv", number =
                       2),
                       preProcess = c("center","scale"),
                       tuneLength = 7)

knn_Uncorr_s5 <- train(Activity ~ ., data = trainUncorr_s5,
                       method = "knn",
                       trControl = trainControl(method="cv", number = 2),
                       preProcess = c("center","scale"),
```

```
343                       tuneLength = 7)
344
345 #Method 2
346 knn_PCA_s1 <- train(Activity ~ ., data = trainPCA_s1,
347                     method = "knn",
348                     trControl = trainControl(method="cv", number = 2),
349                     preProcess = c("center","scale"),
350                     tuneLength = 7)
351
352 knn_PCA_s2 <- train(Activity ~ ., data = trainPCA_s2,
353                     method = "knn",
354                     trControl = trainControl(method="cv", number = 2),
355                     preProcess = c("center","scale"),
356                     tuneLength = 7)
357
358 knn_PCA_s3 <- train(Activity ~ ., data = trainPCA_s3,
359                     method = "knn",
360                     trControl = trainControl(method="cv", number = 2),
361                     preProcess = c("center","scale"),
362                     tuneLength = 7)
363
364 knn_PCA_s4 <- train(Activity ~ ., data = trainPCA_s4,
365                     method = "knn",
366                     trControl = trainControl(method="cv", number = 2),
367                     preProcess = c("center","scale"),
368                     tuneLength = 7)
369
370 knn_PCA_s5 <- train(Activity ~ ., data = trainPCA_s5,
371                     method = "knn",
372                     trControl = trainControl(method="cv", number = 2),
373                     preProcess = c("center","scale"),
374                     tuneLength = 7)
375 #Naive Bayes
376 #Method 1
377 library(klaR)
378 nb_Uncorr_s1 <- train(Activity ~ ., data=trainUncorr_s1,
379                       method = "nb",
380                       trControl = trainControl(method="cv", number=2),
381                       tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
382
383
384 nb_Uncorr_s2 <- train(Activity ~ ., data=trainUncorr_s2,
385                       method = "nb",
386                       trControl = trainControl(method="cv", number=2),
387                       tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
```

```
388
389
390 nb_Uncorr_s3 <- train(Activity ~ ., data=trainUncorr_s3,
391                       method = "nb",
392                       trControl = trainControl(method="cv", number=2),
393                       tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
394
395
396 nb_Uncorr_s4 <- train(Activity ~ ., data=trainUncorr_s4,
397                       method = "nb",
398                       trControl = trainControl(method="cv", number=2),
399                       tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
400
401
402 nb_Uncorr_s5 <- train(Activity ~ ., data=trainUncorr_s5,
403                       method = "nb",
404                       trControl = trainControl(method="cv", number=2),
405                       tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
406
407 #Method 2
408 nb_PCA_s1 <- train(Activity ~ ., data=trainPCA_s1,
409                     method = "nb",
410                     trControl = trainControl(method="cv", number=2),
411                     tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
412
413
414 nb_PCA_s2 <- train(Activity ~ ., data=trainPCA_s2,
415                     method = "nb",
416                     trControl = trainControl(method="cv", number=2),
417                     tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
418
419
420 nb_PCA_s3 <- train(Activity ~ ., data=trainPCA_s3,
421                     method = "nb",
422                     trControl = trainControl(method="cv", number=2),
423                     tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
424
425
426 nb_PCA_s4 <- train(Activity ~ ., data=trainPCA_s4,
427                     method = "nb",
428                     trControl = trainControl(method="cv", number=2),
429                     tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))
430
431
432 nb_PCA_s5 <- train(Activity ~ ., data=trainPCA_s5,
```

```
                    method = "nb",
                    trControl = trainControl(method="cv", number=2),
                    tuneGrid = data.frame(fL=0, usekernel=FALSE, adjust = 0))

#Logistic Regression
#Method 1
library(nnet)
lr_Uncorr_s1 <- train(Activity ~ ., data=trainUncorr_s1,
                        method = "multinom",
                        trControl = trainControl(method="cv", number=2),
                        MaxNWts = 1045)

lr_Uncorr_s2 <- train(Activity ~ ., data=trainUncorr_s2,
                        method = "multinom",
                        trControl = trainControl(method="cv", number=2),
                        MaxNWts = 1045)

lr_Uncorr_s3 <- train(Activity ~ ., data=trainUncorr_s3,
                        method = "multinom",
                        trControl = trainControl(method="cv", number=2),
                        MaxNWts = 1045)

lr_Uncorr_s4 <- train(Activity ~ ., data=trainUncorr_s4,
                        method = "multinom",
                        trControl = trainControl(method="cv", number=2),
                        MaxNWts = 1045)

lr_Uncorr_s5 <- train(Activity ~ ., data=trainUncorr_s5,
                        method = "multinom",
                        trControl = trainControl(method="cv", number=2),
                        MaxNWts = 1045)

#Method 2

lr_PCA_s1 <- train(Activity ~ ., data=trainPCA_s1,
                    method = "multinom",
                    trControl = trainControl(method="cv", number=2),
                    MaxNWts = 1045)

lr_PCA_s2 <- train(Activity ~ ., data=trainPCA_s2,
                    method = "multinom",
                    trControl = trainControl(method="cv", number=2),
                    MaxNWts = 1045)

lr_PCA_s3 <- train(Activity ~ ., data=trainPCA_s3,
```

```
                    method = "multinom",
                    trControl = trainControl(method="cv", number=2),
                    MaxNWts = 1045)

lr_PCA_s4 <- train(Activity ~ ., data=trainPCA_s4,
                    method = "multinom",
                    trControl = trainControl(method="cv", number=2),
                    MaxNWts = 1045)

lr_PCA_s5 <- train(Activity ~ ., data=trainPCA_s5,
                    method = "multinom",
                    trControl = trainControl(method="cv", number=2),
                    MaxNWts = 1045)

#Neural Net

ann_Uncorr_s1 <- train(Activity ~ .,data=trainUncorr_s1,
                        method = 'nnet',
                        trControl = trainControl(method="cv", number=2),
                        tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts =
                            8000)

ann_Uncorr_s2 <- train(Activity ~ .,data=trainUncorr_s2,
                        method = 'nnet',
                        trControl = trainControl(method="cv", number=2),
                        tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts =
                            8000)

ann_Uncorr_s3 <- train(Activity ~ .,data=trainUncorr_s3,
                        method = 'nnet',
                        trControl = trainControl(method="cv", number=2),
                        tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts =
                            8000)

ann_Uncorr_s4 <- train(Activity ~ .,data=trainUncorr_s4,
                        method = 'nnet',
                        trControl = trainControl(method="cv", number=2),
                        tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts =
                            8000)

ann_Uncorr_s5 <- train(Activity ~ .,data=trainUncorr_s5,
                        method = 'nnet',
                        trControl = trainControl(method="cv", number=2),
                        tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts =
                            8000)
```

```
518
519 #Method2
520 ann_PCA_s1 <- train(Activity ~ .,data=trainPCA_s1,
521                     method = 'nnet',
522                     trControl = trainControl(method="cv", number=2),
523                     tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts = 8000)
524
525 ann_PCA_s2 <- train(Activity ~ .,data=trainPCA_s2,
526                     method = 'nnet',
527                     trControl = trainControl(method="cv", number=2),
528                     tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts = 8000)
529
530 ann_PCA_s3 <- train(Activity ~ .,data=trainPCA_s3,
531                     method = 'nnet',
532                     trControl = trainControl(method="cv", number=2),
533                     tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts = 8000)
534
535 ann_PCA_s4 <- train(Activity ~ .,data=trainPCA_s4,
536                     method = 'nnet',
537                     trControl = trainControl(method="cv", number=2),
538                     tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts = 8000)
539
540 ann_PCA_s5 <- train(Activity ~ .,data=trainPCA_s5,
541                     method = 'nnet',
542                     trControl = trainControl(method="cv", number=2),
543                     tuneGrid=expand.grid(size=c(30), decay=c(0.1)), MaxNWts = 8000)
544
545
546 #Ensemble
547 #Bagging
548 #Random Forest
549 #Method 1
550 library(randomForest)
551 rf_Uncorr_s1 <- train(Activity ~ ., data=trainUncorr_s1,
552                       method = "rf", metric="Accuracy",
553                       trControl = trainControl(method="cv", number=2),
554                       tuneGrid=expand.grid(mtry = sqrt(ncol(trainUncorr_s1))))
555
556 rf_Uncorr_s2 <- train(Activity ~ ., data=trainUncorr_s2,
557                       method = "rf", metric="Accuracy",
558                       trControl = trainControl(method="cv", number=2),
559                       tuneGrid=expand.grid(mtry = sqrt(ncol(trainUncorr_s2))))
560
561 rf_Uncorr_s3 <- train(Activity ~ ., data=trainUncorr_s3,
562                       method = "rf", metric="Accuracy",
```

```
                    trControl = trainControl(method="cv", number=2),
                    tuneGrid=expand.grid(mtry = sqrt(ncol(trainUncorr_s3))))

rf_Uncorr_s4 <- train(Activity ~ ., data=trainUncorr_s4,
                    method = "rf", metric="Accuracy",
                    trControl = trainControl(method="cv", number=2),
                    tuneGrid=expand.grid(mtry = sqrt(ncol(trainUncorr_s4))))

rf_Uncorr_s5 <- train(Activity ~ ., data=trainUncorr_s5,
                    method = "rf", metric="Accuracy",
                    trControl = trainControl(method="cv", number=2),
                    tuneGrid=expand.grid(mtry = sqrt(ncol(trainUncorr_s5))))

#Method2

rf_PCA_s1 <- train(Activity ~ ., data=trainPCA_s1,
                   method = "rf", metric="Accuracy",
                   trControl = trainControl(method="cv", number=2),
                   tuneGrid=expand.grid(mtry = sqrt(ncol(trainPCA_s1))))

rf_PCA_s2 <- train(Activity ~ ., data=trainPCA_s2,
                   method = "rf", metric="Accuracy",
                   trControl = trainControl(method="cv", number=2),
                   tuneGrid=expand.grid(mtry = sqrt(ncol(trainPCA_s2))))

rf_PCA_s3 <- train(Activity ~ ., data=trainPCA_s3,
                   method = "rf", metric="Accuracy",
                   trControl = trainControl(method="cv", number=2),
                   tuneGrid=expand.grid(mtry = sqrt(ncol(trainPCA_s3))))

rf_PCA_s4 <- train(Activity ~ ., data=trainPCA_s4,
                   method = "rf", metric="Accuracy",
                   trControl = trainControl(method="cv", number=2),
                   tuneGrid=expand.grid(mtry = sqrt(ncol(trainPCA_s4))))

rf_PCA_s5 <- train(Activity ~ ., data=trainPCA_s5,
                   method = "rf", metric="Accuracy",
                   trControl = trainControl(method="cv", number=2),
                   tuneGrid=expand.grid(mtry = sqrt(ncol(trainPCA_s5))))

library(kernlab)
svm_PCA_s5 <- train(Activity ~ ., data=trainPCA_s5,
                   method = "svmRadial", metric="Accuracy",
                   trControl = trainControl(method="cv", number=2)
                   #,tuneGrid=expand.grid(mtry = sqrt(ncol(trainPCA_s5)))
```

```
608                   )
609 svm_PCA_s5_pred<- predict.train(svm_PCA_s5, newdata = testPCA_s5[,-1])
610 svm_PCA_s5_CF <- confusionMatrix(data = svm_PCA_s5_pred, reference = testPCA_s5[,1])
611 svm_PCA_s5_CF$overall
612
613 ##Evaluation
614 #Decision Tree
615 #Method1
616 dt_Uncorr_s1_pred<- predict.train(dt_Uncorr_s1,
617                                   newdata = testUncorr_s1[,-173])
618 dt_Uncorr_s1_CF <- confusionMatrix(data = dt_Uncorr_s1_pred,
619                                    reference = testUncorr_s1[,173])
620 dt_Uncorr_s1_CF$overall
621 t1<- dt_Uncorr_s1_CF$table
622 dt_Uncorr_s2_pred<- predict.train(dt_Uncorr_s2,
623                                   newdata = testUncorr_s2[,-173])
624 dt_Uncorr_s2_CF <- confusionMatrix(data = dt_Uncorr_s2_pred,
625                                    reference = testUncorr_s2[,173])
626 dt_Uncorr_s2_CF$overall
627
628 dt_Uncorr_s3_pred<- predict.train(dt_Uncorr_s3, newdata = testUncorr_s3[,-173])
629 dt_Uncorr_s3_CF <- confusionMatrix(data = dt_Uncorr_s3_pred, reference = testUncorr_
        s3[,173])
630 dt_Uncorr_s3_CF$overall
631
632 dt_Uncorr_s4_pred<- predict.train(dt_Uncorr_s4, newdata = testUncorr_s4[,-173])
633 dt_Uncorr_s4_CF <- confusionMatrix(data = dt_Uncorr_s4_pred, reference = testUncorr_
        s4[,173])
634 dt_Uncorr_s4_CF$overall
635
636 dt_Uncorr_s5_pred<- predict.train(dt_Uncorr_s5, newdata = testUncorr_s5[,-173])
637 dt_Uncorr_s5_CF <- confusionMatrix(data = dt_Uncorr_s5_pred, reference = testUncorr_
        s5[,173])
638 dt_Uncorr_s5_CF$overall
639
640 #Method 2
641 dt_PCA_s1_pred<- predict.train(dt_PCA_s1, newdata = testPCA_s1[,-1])
642 dt_PCA_s1_CF <- confusionMatrix(data = dt_PCA_s1_pred, reference = testPCA_s1[,1])
643 dt_PCA_s1_CF$overall
644
645 dt_PCA_s2_pred<- predict.train(dt_PCA_s2, newdata = testPCA_s2[,-1])
646 dt_PCA_s2_CF <- confusionMatrix(data = dt_PCA_s2_pred, reference = testPCA_s2[,1])
647 dt_PCA_s2_CF$overall
648
649 dt_PCA_s3_pred<- predict.train(dt_PCA_s3, newdata = testPCA_s3[,-1])
```

```
650 dt_PCA_s3_CF <- confusionMatrix(data = dt_PCA_s3_pred, reference = testPCA_s3[,1])
651 dt_PCA_s3_CF$overall
652
653 dt_PCA_s4_pred<- predict.train(dt_PCA_s4, newdata = testPCA_s4[,-1])
654 dt_PCA_s4_CF <- confusionMatrix(data = dt_PCA_s4_pred, reference = testPCA_s4[,1])
655 dt_PCA_s4_CF$overall
656
657 dt_PCA_s5_pred<- predict.train(dt_PCA_s5, newdata = testPCA_s5[,-1])
658 dt_PCA_s5_CF <- confusionMatrix(data = dt_PCA_s5_pred, reference = testPCA_s5[,1])
659 dt_PCA_s5_CF$overall
660
661
662 #KNN
663 #Method 1
664
665 knn_Uncorr_s1_pred <- predict.train(knn_Uncorr_s1,newdata = testUncorr_s1[,-173] )
666 knn_Uncorr_s1_CF <- confusionMatrix(knn_Uncorr_s1_pred, testUncorr_s1[,173] )
667 knn_Uncorr_s1_CF$overall
668
669 knn_Uncorr_s2_pred <- predict.train(knn_Uncorr_s2,newdata = testUncorr_s2[,-173] )
670 knn_Uncorr_s2_CF <- confusionMatrix(knn_Uncorr_s2_pred, testUncorr_s2[,173] )
671 knn_Uncorr_s2_CF$overall
672
673 knn_Uncorr_s3_pred <- predict.train(knn_Uncorr_s3,newdata = testUncorr_s3[,-173] )
674 knn_Uncorr_s3_CF <- confusionMatrix(knn_Uncorr_s3_pred, testUncorr_s3[,173] )
675 knn_Uncorr_s3_CF$overall
676
677 knn_Uncorr_s4_pred <- predict.train(knn_Uncorr_s4,newdata = testUncorr_s4[,-173] )
678 knn_Uncorr_s4_CF <- confusionMatrix(knn_Uncorr_s4_pred, testUncorr_s4[,173] )
679 knn_Uncorr_s4_CF$overall
680
681 knn_Uncorr_s5_pred <- predict.train(knn_Uncorr_s5,newdata = testUncorr_s5[,-173] )
682 knn_Uncorr_s5_CF <- confusionMatrix(knn_Uncorr_s5_pred, testUncorr_s5[,173] )
683 knn_Uncorr_s5_CF$overall
684
685 #Method 2
686
687 knn_PCA_s1_pred <- predict.train(knn_PCA_s1,newdata = testPCA_s1[,-1] )
688 knn_PCA_s1_CF <- confusionMatrix(knn_PCA_s1_pred, testPCA_s1[,1] )
689 knn_PCA_s1_CF$overall
690
691 knn_PCA_s2_pred <- predict.train(knn_PCA_s2,newdata = testPCA_s2[,-1] )
692 knn_PCA_s2_CF <- confusionMatrix(knn_PCA_s2_pred, testPCA_s2[,1] )
693 knn_PCA_s2_CF$overall
694
```

```
knn_PCA_s3_pred <- predict.train(knn_PCA_s3,newdata = testPCA_s3[,-1] )
knn_PCA_s3_CF <- confusionMatrix(knn_PCA_s3_pred, testPCA_s3[,1] )
knn_PCA_s3_CF$overall

knn_PCA_s4_pred <- predict.train(knn_PCA_s4,newdata = testPCA_s4[,-1] )
knn_PCA_s4_CF <- confusionMatrix(knn_PCA_s4_pred, testPCA_s4[,1] )
knn_PCA_s4_CF$overall

knn_PCA_s5_pred <- predict.train(knn_PCA_s5,newdata = testPCA_s5[,-1] )
knn_PCA_s5_CF <- confusionMatrix(knn_PCA_s5_pred, testPCA_s5[,1] )
knn_PCA_s5_CF$overall

#Naive Bayes
#Method 1
nb_Uncorr_s1_pred <- predict.train(nb_Uncorr_s1, newdata = testUncorr_s1[,-173])
nb_Uncorr_s1_CF <- confusionMatrix(nb_Uncorr_s1_pred, testUncorr_s1[,173])
nb_Uncorr_s1_CF$overall

nb_Uncorr_s2_pred <- predict.train(nb_Uncorr_s2, newdata = testUncorr_s2[,-173])
nb_Uncorr_s2_CF <- confusionMatrix(nb_Uncorr_s2_pred, testUncorr_s2[,173])
nb_Uncorr_s2_CF$overall

nb_Uncorr_s3_pred <- predict.train(nb_Uncorr_s3, newdata = testUncorr_s3[,-173])
nb_Uncorr_s3_CF <- confusionMatrix(nb_Uncorr_s3_pred, testUncorr_s3[,173])
nb_Uncorr_s3_CF$overall

nb_Uncorr_s4_pred <- predict.train(nb_Uncorr_s4, newdata = testUncorr_s4[,-173])
nb_Uncorr_s4_CF <- confusionMatrix(nb_Uncorr_s4_pred, testUncorr_s4[,173])
nb_Uncorr_s4_CF$overall

nb_Uncorr_s5_pred <- predict.train(nb_Uncorr_s5, newdata = testUncorr_s5[,-173])
nb_Uncorr_s5_CF <- confusionMatrix(nb_Uncorr_s5_pred, testUncorr_s5[,173])
nb_Uncorr_s5_CF$overall

#Method 2
nb_PCA_s1_pred <- predict.train(nb_PCA_s1, newdata = testPCA_s1[,-1])
nb_PCA_s1_CF <- confusionMatrix(nb_PCA_s1_pred, testPCA_s1[,1])
nb_PCA_s1_CF$overall

nb_PCA_s2_pred <- predict.train(nb_PCA_s2, newdata = testPCA_s2[,-1])
nb_PCA_s2_CF <- confusionMatrix(nb_PCA_s2_pred, testPCA_s2[,1])
nb_PCA_s2_CF$overall

nb_PCA_s3_pred <- predict.train(nb_PCA_s3, newdata = testPCA_s3[,-1])
nb_PCA_s3_CF <- confusionMatrix(nb_PCA_s3_pred, testPCA_s3[,1])
```

```
740 nb_PCA_s3_CF$overall
741
742 nb_PCA_s4_pred <- predict.train(nb_PCA_s4, newdata = testPCA_s4[,-1])
743 nb_PCA_s4_CF <- confusionMatrix(nb_PCA_s4_pred, testPCA_s4[,1])
744 nb_PCA_s4_CF$overall
745
746 nb_PCA_s5_pred <- predict.train(nb_PCA_s5, newdata = testPCA_s5[,-1])
747 nb_PCA_s5_CF <- confusionMatrix(nb_PCA_s5_pred, testPCA_s5[,1])
748 nb_PCA_s5_CF$overall
749
750 #Logistic Regression
751 #Method 1
752 lr_Uncorr_s1_pred <- predict.train(lr_Uncorr_s1, newdata = testUncorr_s1[,-173])
753 lr_Uncorr_s1_CF <- confusionMatrix(lr_Uncorr_s1_pred, testUncorr_s1[,173])
754 lr_Uncorr_s1_CF$overall
755
756 lr_Uncorr_s2_pred <- predict.train(lr_Uncorr_s2, newdata = testUncorr_s2[,-173])
757 lr_Uncorr_s2_CF <- confusionMatrix(lr_Uncorr_s2_pred, testUncorr_s2[,173])
758 lr_Uncorr_s2_CF$overall
759
760 lr_Uncorr_s3_pred <- predict.train(lr_Uncorr_s3, newdata = testUncorr_s3[,-173])
761 lr_Uncorr_s3_CF <- confusionMatrix(lr_Uncorr_s3_pred, testUncorr_s3[,173])
762 lr_Uncorr_s3_CF$overall
763
764 lr_Uncorr_s4_pred <- predict.train(lr_Uncorr_s4, newdata = testUncorr_s4[,-173])
765 lr_Uncorr_s4_CF <- confusionMatrix(lr_Uncorr_s4_pred, testUncorr_s4[,173])
766 lr_Uncorr_s4_CF$overall
767
768 lr_Uncorr_s5_pred <- predict.train(lr_Uncorr_s5, newdata = testUncorr_s5[,-173])
769 lr_Uncorr_s5_CF <- confusionMatrix(lr_Uncorr_s5_pred, testUncorr_s5[,173])
770 lr_Uncorr_s5_CF$overall
771
772 #Method 2
773 lr_PCA_s1_pred <- predict.train(lr_PCA_s1, newdata = testPCA_s1[,-1])
774 lr_PCA_s1_CF <- confusionMatrix(lr_PCA_s1_pred, testPCA_s1[,1])
775 lr_PCA_s1_CF$overall
776
777 lr_PCA_s2_pred <- predict.train(lr_PCA_s2, newdata = testPCA_s2[,-1])
778 lr_PCA_s2_CF <- confusionMatrix(lr_PCA_s2_pred, testPCA_s2[,1])
779 lr_PCA_s2_CF$overall
780
781 lr_PCA_s3_pred <- predict.train(lr_PCA_s3, newdata = testPCA_s3[,-1])
782 lr_PCA_s3_CF <- confusionMatrix(lr_PCA_s3_pred, testPCA_s3[,1])
783 lr_PCA_s3_CF$overall
784
```

```
lr_PCA_s4_pred <- predict.train(lr_PCA_s4, newdata = testPCA_s4[,-1])
lr_PCA_s4_CF <- confusionMatrix(lr_PCA_s4_pred, testPCA_s4[,1])
lr_PCA_s4_CF$overall

lr_PCA_s5_pred <- predict.train(lr_PCA_s5, newdata = testPCA_s5[,-1])
lr_PCA_s5_CF <- confusionMatrix(lr_PCA_s5_pred, testPCA_s5[,1])
lr_PCA_s5_CF$overall

#Artificial Neural Net
#Method1
ann_Uncorr_s1_pred <- predict.train(ann_Uncorr_s1, newdata = testUncorr_s1[,-173])
ann_Uncorr_s1_CF <- confusionMatrix(ann_Uncorr_s1_pred, testUncorr_s1[,173])
ann_Uncorr_s1_CF$overall

ann_Uncorr_s2_pred <- predict.train(ann_Uncorr_s2, newdata = testUncorr_s2[,-173])
ann_Uncorr_s2_CF <- confusionMatrix(ann_Uncorr_s2_pred, testUncorr_s2[,173])
ann_Uncorr_s2_CF$overall

ann_Uncorr_s3_pred <- predict.train(ann_Uncorr_s3, newdata = testUncorr_s3[,-173])
ann_Uncorr_s3_CF <- confusionMatrix(ann_Uncorr_s3_pred, testUncorr_s3[,173])
ann_Uncorr_s3_CF$overall

ann_Uncorr_s4_pred <- predict.train(ann_Uncorr_s4, newdata = testUncorr_s4[,-173])
ann_Uncorr_s4_CF <- confusionMatrix(ann_Uncorr_s4_pred, testUncorr_s4[,173])
ann_Uncorr_s4_CF$overall

ann_Uncorr_s5_pred <- predict.train(ann_Uncorr_s5, newdata = testUncorr_s5[,-173])
ann_Uncorr_s5_CF <- confusionMatrix(ann_Uncorr_s5_pred, testUncorr_s5[,173])
ann_Uncorr_s5_CF$overall

#Method 2
ann_PCA_s1_pred <- predict.train(ann_PCA_s1, newdata = testPCA_s1[,-1])
ann_PCA_s1_CF <- confusionMatrix(ann_PCA_s1_pred, testPCA_s1[,1])
ann_PCA_s1_CF$overall

ann_PCA_s2_pred <- predict.train(ann_PCA_s2, newdata = testPCA_s2[,-1])
ann_PCA_s2_CF <- confusionMatrix(ann_PCA_s2_pred, testPCA_s2[,1])
ann_PCA_s2_CF$overall

ann_PCA_s3_pred <- predict.train(ann_PCA_s3, newdata = testPCA_s3[,-1])
ann_PCA_s3_CF <- confusionMatrix(ann_PCA_s3_pred, testPCA_s3[,1])
ann_PCA_s3_CF$overall

ann_PCA_s4_pred <- predict.train(ann_PCA_s4, newdata = testPCA_s4[,-1])
ann_PCA_s4_CF <- confusionMatrix(ann_PCA_s4_pred, testPCA_s4[,1])
```

```
ann_PCA_s4_CF$overall

ann_PCA_s5_pred <- predict.train(ann_PCA_s5, newdata = testPCA_s5[,-1])
ann_PCA_s5_CF <- confusionMatrix(ann_PCA_s5_pred, testPCA_s5[,1])
ann_PCA_s5_CF$overall

#Random Forest
#Method 1
rf_Uncorr_s1_pred <- predict.train(rf_Uncorr_s1, newdata = testUncorr_s1[,-173])
rf_Uncorr_s1_CF <- confusionMatrix(rf_Uncorr_s1_pred, testUncorr_s1[,173])
rf_Uncorr_s1_CF$overall

rf_Uncorr_s2_pred <- predict.train(rf_Uncorr_s2, newdata = testUncorr_s2[,-173])
rf_Uncorr_s2_CF <- confusionMatrix(rf_Uncorr_s2_pred, testUncorr_s2[,173])
rf_Uncorr_s2_CF$overall

rf_Uncorr_s3_pred <- predict.train(rf_Uncorr_s3, newdata = testUncorr_s3[,-173])
rf_Uncorr_s3_CF <- confusionMatrix(rf_Uncorr_s3_pred, testUncorr_s3[,173])
rf_Uncorr_s3_CF$overall

rf_Uncorr_s4_pred <- predict.train(rf_Uncorr_s4, newdata = testUncorr_s4[,-173])
rf_Uncorr_s4_CF <- confusionMatrix(rf_Uncorr_s4_pred, testUncorr_s4[,173])
rf_Uncorr_s4_CF$overall

rf_Uncorr_s5_pred <- predict.train(rf_Uncorr_s5, newdata = testUncorr_s5[,-173])
rf_Uncorr_s5_CF <- confusionMatrix(rf_Uncorr_s5_pred, testUncorr_s5[,173])
rf_Uncorr_s5_CF$overall

#Method 2
rf_PCA_s1_pred <- predict.train(rf_PCA_s1, newdata = testPCA_s1[,-1])
rf_PCA_s1_CF <- confusionMatrix(rf_PCA_s1_pred, testPCA_s1[,1])
rf_PCA_s1_CF$overall

rf_PCA_s2_pred <- predict.train(rf_PCA_s2, newdata = testPCA_s2[,-1])
rf_PCA_s2_CF <- confusionMatrix(rf_PCA_s2_pred, testPCA_s2[,1])
rf_PCA_s2_CF$overall

rf_PCA_s3_pred <- predict.train(rf_PCA_s3, newdata = testPCA_s3[,-1])
rf_PCA_s3_CF <- confusionMatrix(rf_PCA_s3_pred, testPCA_s3[,1])
rf_PCA_s3_CF$overall

rf_PCA_s4_pred <- predict.train(rf_PCA_s4, newdata = testPCA_s4[,-1])
rf_PCA_s4_CF <- confusionMatrix(rf_PCA_s4_pred, testPCA_s4[,1])
rf_PCA_s4_CF$overall

```

```
875 rf_PCA_s5_pred <- predict.train(rf_PCA_s5, newdata = testPCA_s5[,-1])
876 rf_PCA_s5_CF <- confusionMatrix(rf_PCA_s5_pred, testPCA_s5[,1])
877 rf_PCA_s5_CF$overall
878
879 rm(trainUncorr_s1,trainUncorr_s2,trainUncorr_s3,trainUncorr_s4,trainUncorr_s5,
880    trainPCA_s1,trainPCA_s2,trainPCA_s3,trainPCA_s4,trainPCA_s5,
881    testUncorr_s1,testUncorr_s2,testUncorr_s3,testUncorr_s4,testUncorr_s5,
882    testPCA_s1,testPCA_s2,testPCA_s3,testPCA_s4,testPCA_s5)
883
884 #Combine Results
885
886 dt_M1_Acc <- data.frame(dt_Uncorr_s1_CF$overall[1], dt_Uncorr_s2_CF$overall[1],dt_
        Uncorr_s3_CF$overall[1],
887                 dt_Uncorr_s4_CF$overall[1],dt_Uncorr_s5_CF$overall[1])
888
889 dt_M2_Acc <- data.frame(dt_PCA_s1_CF$overall[1], dt_PCA_s2_CF$overall[1],dt_PCA_s3_CF
        $overall[1],
890                       dt_PCA_s4_CF$overall[1],dt_PCA_s5_CF$overall[1])
891
892 dt_Acc <- cbind(dt_M1_Acc, dt_M2_Acc)
893
894
895 knn_M1_Acc <- data.frame(knn_Uncorr_s1_CF$overall[1], knn_Uncorr_s2_CF$overall[1],knn
        _Uncorr_s3_CF$overall[1],
896                        knn_Uncorr_s4_CF$overall[1],knn_Uncorr_s5_CF$overall[1])
897
898 knn_M2_Acc <- data.frame(knn_PCA_s1_CF$overall[1], knn_PCA_s2_CF$overall[1],knn_PCA_
        s3_CF$overall[1],
899                        knn_PCA_s4_CF$overall[1],knn_PCA_s5_CF$overall[1])
900
901 knn_Acc <- cbind(knn_M1_Acc, knn_M2_Acc)
902
903
904 nb_M1_Acc <- data.frame(nb_Uncorr_s1_CF$overall[1], nb_Uncorr_s2_CF$overall[1],nb_
        Uncorr_s3_CF$overall[1],
905                       nb_Uncorr_s4_CF$overall[1],nb_Uncorr_s5_CF$overall[1])
906
907 nb_M2_Acc <- data.frame(nb_PCA_s1_CF$overall[1], nb_PCA_s2_CF$overall[1],nb_PCA_s3_CF
        $overall[1],
908                       nb_PCA_s4_CF$overall[1],nb_PCA_s5_CF$overall[1])
909
910 nb_Acc <- cbind(nb_M1_Acc, nb_M2_Acc)
911
912
```

```
913 lr_M1_Acc <- data.frame(lr_Uncorr_s1_CF$overall[1], lr_Uncorr_s2_CF$overall[1],lr_
        Uncorr_s3_CF$overall[1],
914                         lr_Uncorr_s4_CF$overall[1],lr_Uncorr_s5_CF$overall[1])
915
916 lr_M2_Acc <- data.frame(lr_PCA_s1_CF$overall[1], lr_PCA_s2_CF$overall[1],lr_PCA_s3_CF
        $overall[1],
917                         lr_PCA_s4_CF$overall[1],lr_PCA_s5_CF$overall[1])
918
919 lr_Acc <- cbind(lr_M1_Acc, lr_M2_Acc)
920
921
922 ann_M1_Acc <- data.frame(ann_Uncorr_s1_CF$overall[1], ann_Uncorr_s2_CF$overall[1],ann
        _Uncorr_s3_CF$overall[1],
923                          ann_Uncorr_s4_CF$overall[1],ann_Uncorr_s5_CF$overall[1])
924
925 ann_M2_Acc <- data.frame(ann_PCA_s1_CF$overall[1], ann_PCA_s2_CF$overall[1],ann_PCA_
        s3_CF$overall[1],
926                          ann_PCA_s4_CF$overall[1],ann_PCA_s5_CF$overall[1])
927
928 ann_Acc <- cbind(ann_M1_Acc, ann_M2_Acc)
929
930
931 rf_M1_Acc <- data.frame(rf_Uncorr_s1_CF$overall[1], rf_Uncorr_s2_CF$overall[1],rf_
        Uncorr_s3_CF$overall[1],
932                         rf_Uncorr_s4_CF$overall[1],rf_Uncorr_s5_CF$overall[1])
933
934 rf_M2_Acc <- data.frame(rf_PCA_s1_CF$overall[1], rf_PCA_s2_CF$overall[1],rf_PCA_s3_CF
        $overall[1],
935                         rf_PCA_s4_CF$overall[1],rf_PCA_s5_CF$overall[1])
936
937 rf_Acc <- cbind(rf_M1_Acc, rf_M2_Acc)
938 l1<- (t(rf_M1_Acc))
939 boxplot(rf_M2_Acc)
940 boxplot(l1)
941
942 dt_Acc<- sort(t(dt_Acc))
943 dt_Acc
944 plot(dt_Acc)
945
946 knn_Acc<- sort(t(knn_Acc))
947 knn_Acc
948 plot(knn_Acc)
949
950 nb_Acc<- sort(t(nb_Acc))
951 nb_Acc
```

```
952 plot(nb_Acc)
953
954 lr_Acc<- sort(t(lr_Acc))
955 lr_Acc
956 plot(lr_Acc)
957
958 ann_Acc<- sort(t(ann_Acc))
959 ann_Acc
960 plot(ann_Acc)
961 boxplot(ann_Acc)
962 histogram(ann_Acc)
963
964 rf_Acc<- sort(t(rf_Acc))
965 rf_Acc
966 plot(rf_Acc)
967 boxplot(rf_Acc)
968 histogram(rf_Acc)
969
970 finalAcc <- data.frame(dt_Acc, knn_Acc,nb_Acc,lr_Acc,ann_Acc)
971 sigs <- list()
972 for(i in 1:ncol(finalAcc)){
973   sigs[[i]] <- wilcox.test(finalAcc[,i], finalAcc[,i+1], paired = TRUE)
974 }
975
976 wilcox.test(dt_Acc, knn_Acc, paired = TRUE)
977 wilcox.test(dt_Acc, nb_Acc, paired = TRUE)
978 wilcox.test(dt_Acc, lr_Acc, paired = TRUE)
979 wilcox.test(dt_Acc, ann_Acc, paired = TRUE)
980
981 wilcox.test(knn_Acc, dt_Acc, paired = TRUE)
982 wilcox.test(knn_Acc, nb_Acc, paired = TRUE)
983 wilcox.test(knn_Acc, lr_Acc, paired = TRUE)
984 wilcox.test(knn_Acc, ann_Acc, paired = TRUE)
985
986 wilcox.test(nb_Acc, dt_Acc, paired = TRUE)
987 wilcox.test(nb_Acc, knn_Acc, paired = TRUE)
988 wilcox.test(nb_Acc, lr_Acc, paired = TRUE)
989 wilcox.test(nb_Acc, ann_Acc, paired = TRUE)
990
991 wilcox.test(lr_Acc, dt_Acc, paired = TRUE)
992 wilcox.test(lr_Acc, knn_Acc, paired = TRUE)
993 wilcox.test(lr_Acc, nb_Acc, paired = TRUE)
994 wilcox.test(lr_Acc, ann_Acc, paired = TRUE)
995
996 wilcox.test(ann_Acc, dt_Acc, paired = TRUE)
```

```
997 wilcox.test(ann_Acc, knn_Acc, paired = TRUE)
998 wilcox.test(ann_Acc, nb_Acc, paired = TRUE)
999 wilcox.test(ann_Acc, lr_Acc, paired = TRUE)
1000
1001 t.test(ann_Acc, dt_Acc, paired = TRUE)
1002 t.test(ann_Acc, knn_Acc, paired = TRUE)
1003 t.test(ann_Acc, nb_Acc, paired = TRUE)
1004 t.test(ann_Acc, lr_Acc, paired = TRUE)
1005
1006 t.test(lr_Acc, ann_Acc, paired = TRUE)
1007
1008
1009 plot(knn_Acc, rf_Acc)
1010 d1 <-density(lr_Acc)
1011 d2 <-density(ann_Acc)
1012 plot(d1, d2)
1013 plot(d1, col = "#E41A1C", main = "Density plots for LR & ANN", xlab= "Accuracy Values
     ",
1014      ylim=range(1:100))
1015 lines(d2,col = "#377EB8")
1016 cols<-brewer.pal(n=1,name="Set1")
1017 legend("topright",legend=rep(c("LR","ANN")),col=rep(cols,times=2),
1018        pch=rep(c(16,18)),bty="n",ncol=2,cex=1,pt.cex=1,xpd=TRUE)
1019
1020 ###################Resample########
1021
1022 models_DT <- list(dt_Uncorr_s1=dt_Uncorr_s1, dt_Uncorr_s2=dt_Uncorr_s2,
1023                       dt_Uncorr_s3=dt_Uncorr_s3, dt_Uncorr_s4=dt_Uncorr_s4,
1024                       dt_Uncorr_s5=dt_Uncorr_s5, dt_PCA_s1=dt_PCA_s1,
1025                       dt_PCA_s2=dt_PCA_s2,dt_PCA_s3=dt_PCA_s3,dt_PCA_s4=dt_PCA_s4,
1026                       dt_PCA_s5=dt_PCA_s5)
1027
1028 dt_M1 <- list(dt_Uncorr_s1=dt_Uncorr_s1, dt_Uncorr_s2=dt_Uncorr_s2,
1029               dt_Uncorr_s3=dt_Uncorr_s3, dt_Uncorr_s4=dt_Uncorr_s4,
1030               dt_Uncorr_s5=dt_Uncorr_s5)
1031 dt_M2 <- list(dt_PCA_s1=dt_PCA_s1, dt_PCA_s2=dt_PCA_s2,
1032               dt_PCA_s3=dt_PCA_s3,dt_PCA_s4=dt_PCA_s4,
1033               dt_PCA_s5=dt_PCA_s5)
1034
1035 results_DT <- resamples(models_DT)
1036 summary(results_DT)
1037 dotplot(results_DT)
1038 bwplot(results_DT)
1039 sort(results_DT, decreasing = TRUE, metric = results_DT$metrics[1])
1040
```

```
results_DT_M1 <- resamples(dt_M1)
bwplot(results_DT_M1)

results_DT_M2 <- resamples(dt_M2)
bwplot(results_DT_M1)


models_KNN <- list(knn_Uncorr_s1=knn_Uncorr_s1, knn_Uncorr_s2=knn_Uncorr_s2,
                   knn_Uncorr_s3=knn_Uncorr_s3, knn_Uncorr_s4=knn_Uncorr_s4,
                   knn_Uncorr_s5=knn_Uncorr_s5, knn_PCA_s1=knn_PCA_s1,
                   knn_PCA_s2=knn_PCA_s2,knn_PCA_s3=knn_PCA_s3,knn_PCA_s4=knn_PCA_s4,
                   knn_PCA_s5=knn_PCA_s5)

results_KNN <- resamples(models_KNN)
summary(results_KNN)
dotplot(results_KNN)
bwplot(results_KNN)
list1<- (sort(results_KNN, decreasing = TRUE, metric = results_KNN$metrics[1]))
plot(list1)

knn_M1 <- list(knn_Uncorr_s1=knn_Uncorr_s1, knn_Uncorr_s2=knn_Uncorr_s2,
                   knn_Uncorr_s3=knn_Uncorr_s3, knn_Uncorr_s4=knn_Uncorr_s4,
                   knn_Uncorr_s5=knn_Uncorr_s5)

knn_M2 <- list(knn_PCA_s1=knn_PCA_s1,knn_PCA_s2=knn_PCA_s2,
               knn_PCA_s3=knn_PCA_s3,knn_PCA_s4=knn_PCA_s4,
               knn_PCA_s5=knn_PCA_s5)

results_KNN_M1 <- resamples(knn_M1)
bwplot(results_KNN_M1)

results_KNN_M2 <- resamples(knn_M2)
bwplot(results_KNN_M2)


models_NB <- list(nb_Uncorr_s1=nb_Uncorr_s1, nb_Uncorr_s2=nb_Uncorr_s2,
                  nb_Uncorr_s3=nb_Uncorr_s3, nb_Uncorr_s4=nb_Uncorr_s4,
                  nb_Uncorr_s5=nb_Uncorr_s5, nb_PCA_s1=nb_PCA_s1,
                  nb_PCA_s2=nb_PCA_s2,nb_PCA_s3=nb_PCA_s3,nb_PCA_s4=nb_PCA_s4,
                  nb_PCA_s5=nb_PCA_s5)

results_NB <- resamples(models_NB)
summary(results_NB)
dotplot(results_NB)
bwplot(results_NB)
```

```
sort(results_NB, decreasing = TRUE, metric = results_NB$metrics[1])


nb_M1 <- list(nb_Uncorr_s1=nb_Uncorr_s1, nb_Uncorr_s2=nb_Uncorr_s2,
              nb_Uncorr_s3=nb_Uncorr_s3, nb_Uncorr_s4=nb_Uncorr_s4,
              nb_Uncorr_s5=nb_Uncorr_s5)

nb_M2 <- list(nb_PCA_s1=nb_PCA_s1,nb_PCA_s2=nb_PCA_s2,
              nb_PCA_s3=nb_PCA_s3,nb_PCA_s4=nb_PCA_s4,
              nb_PCA_s5=nb_PCA_s5)

results_NB_M1 <- resamples(nb_M1)
bwplot(results_NB_M1)

results_NB_M2 <- resamples(nb_M2)
bwplot(results_NB_M2)


models_LR <- list(lr_Uncorr_s1=lr_Uncorr_s1, lr_Uncorr_s2=lr_Uncorr_s2,
                  lr_Uncorr_s3=lr_Uncorr_s3, lr_Uncorr_s4=lr_Uncorr_s4,
                  lr_Uncorr_s5=lr_Uncorr_s5, lr_PCA_s1=lr_PCA_s1,
                  lr_PCA_s2=lr_PCA_s2,lr_PCA_s3=lr_PCA_s3,lr_PCA_s4=lr_PCA_s4,
                  lr_PCA_s5=lr_PCA_s5)

results_LR <- resamples(models_LR)
summary(results_LR)
dotplot(results_LR)
bwplot(results_LR)
sort(results_LR, decreasing = TRUE, metric = results_LR$metrics[1])


lr_M1 <- list(lr_Uncorr_s1=lr_Uncorr_s1, lr_Uncorr_s2=lr_Uncorr_s2,
              lr_Uncorr_s3=lr_Uncorr_s3, lr_Uncorr_s4=lr_Uncorr_s4,
              lr_Uncorr_s5=lr_Uncorr_s5)

lr_M2 <- list(lr_PCA_s1=lr_PCA_s1,lr_PCA_s2=lr_PCA_s2,
              lr_PCA_s3=lr_PCA_s3,lr_PCA_s4=lr_PCA_s4,
              lr_PCA_s5=lr_PCA_s5)

results_LR_M1 <- resamples(lr_M1)
bwplot(results_LR_M1)

results_LR_M2 <- resamples(lr_M2)
bwplot(results_LR_M2)

```

```
1131
1132 models_ANN <- list(ann_Uncorr_s1=ann_Uncorr_s1, ann_Uncorr_s2=ann_Uncorr_s2,
1133                    ann_Uncorr_s3=ann_Uncorr_s3, ann_Uncorr_s4=ann_Uncorr_s4,
1134                    ann_Uncorr_s5=ann_Uncorr_s5, ann_PCA_s1=ann_PCA_s1,
1135                    ann_PCA_s2=ann_PCA_s2,ann_PCA_s3=ann_PCA_s3,ann_PCA_s4=ann_PCA_s4,
1136                    ann_PCA_s5=ann_PCA_s5)
1137
1138 results_ANN <- resamples(models_ANN)
1139 summary(results_ANN)
1140 dotplot(results_ANN)
1141 bwplot(results_ANN)
1142
1143 sort(results_ANN, decreasing = TRUE, metric = results_ANN$metrics[1])
1144
1145 ann_M1 <- list(ann_Uncorr_s1=ann_Uncorr_s1, ann_Uncorr_s2=ann_Uncorr_s2,
1146                ann_Uncorr_s3=ann_Uncorr_s3, ann_Uncorr_s4=ann_Uncorr_s4,
1147                ann_Uncorr_s5=ann_Uncorr_s5)
1148
1149 ann_M2 <- list(ann_PCA_s1=ann_PCA_s1,ann_PCA_s2=ann_PCA_s2,
1150                ann_PCA_s3=ann_PCA_s3,ann_PCA_s4=ann_PCA_s4,
1151                ann_PCA_s5=ann_PCA_s5)
1152
1153 results_ANN_M1 <- resamples(ann_M1)
1154 bwplot(results_ANN_M1)
1155
1156 results_ANN_M2 <- resamples(ann_M2)
1157 bwplot(results_ANN_M2)
1158
1159
1160 models_merge <- resamples(list(ANN = ann_Uncorr_s1,LR = lr_Uncorr_s2,
1161                                NB = nb_Uncorr_s4,KNN = knn_Uncorr_s1,
1162                                DT = dt_Uncorr_s1))
1163
1164 models_m1 <- resamples(list(ANN = ann_M1, LR = lr_M1))
1165                           NB = results_NB_M1, KNN = results_KNN_M1,
1166                           DT = results_DT_M1))
1167 bwplot(models_merge)
1168 bwplot(models_m1)
```