Statistical Computing with R and Python (STAT 654)

Project: Predictive Modeling for Human Activity Recognition with Smartphone Sensors

Dataset: Human Activity Recognition Dataset

Group 6
Jayesh Patil (230002375)
Soham Deshpande (829009731)
Sowmya Kolluru (230006706)
Vaidehi Natu (830000028)
Vaishnavi Patki (430000405)

`

INDEX

1. INTRODUCTION

Problem Statement:

Predictive Modeling for Human Activity Recognition with Smartphone Sensors.
Human activity recognition is the problem of predicting activities performed by a person by tracing their movements with the help of sensors. The Human Activity Recognition Dataset (https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones) is used for the same.

Project background and objective:

Human Activity Recognition (HAR) from sensor data can be very useful in developing assistive technology for applications like daily activity monitoring elderly and physically impaired people. The basic six activities (walking, walking upstairs, walking downstairs, sitting, standing, laying) can create a constructive feedback for numerous disease monitoring systems.

The objective is to classify the activities performed in one of the six activities based on reading obtained from the accelerometer and gyroscope.

Description:

The Human Activity Recognition Dataset is built based on experiments carried over 30 participants within an age bracket of 19-48 years where the performed daily activities (walking, walking upstairs, walking downstairs, sitting, standing, laying) while wearing a waist-mounted smartphone (Samsung Galaxy S II).

By using the embedded accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity at a rate of 50Hz were captured. The experiments have been video-recorded to label the data manually. The obtained dataset has then been randomly partitioned into two parts; data from 70% of the people was selected for generating the training data and the remaining 30% was used for the test data.

Data description:

The dataset contains 561 predictors (columns) with time and frequency domain variables and their corresponding activity labels. The acquired data was preprocessed for noise. The sensor acceleration signal consisted of gravitational and body motion components which were separated using a Butterworth low-pass filter.

For each record in the dataset the following is provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables.
- Its activity labels.
- An identifier of the subject who carried out the experiment
- The given dataset is clean and scaled.

2. EXECUTIVE SUMMARY

Our objective is to develop and select the best predictive model which recognizes the human activity with maximum accuracy. Accuracy in this case is defined as the correctness in the prediction of the class of a given activity.

Human activity recognition (HAR) aims at determining the state of user by continuously monitoring several physiological signals from sensor data. HAR has gained significance over recent years due to its applications in various fields such as health, security and surveillance environments. The sensor data of human activity recorded using smart phones attached to the subject body can be used to develop an efficient approach for remote patient activity monitoring for elderly and physically impaired people. These models are developed on a data set collected from the readings obtained from accelerometers and gyroscopes sensors. The activities monitored include; walking, walking upstairs, walking downstairs, sitting, standing, laying. These activities were monitored for 30 persons in the age group of 19-48 years.

Exploratory data analysis was carried out using principle component analysis and hierarchal clustering and K means clustering. Principle Components Analysis (PCA) was performed to select the smaller number of predictor variables among 561 for further model development. The number of PCs explaining 95% variance of the data were considered.

Several classification models like Logistic Regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbors (KNN), Random Forests and Bagging were developed and tested for accuracy and computational time of models are noted. These models are then compared with respect to their test accuracy and computational times and the model with low misclassification rate and less computational time is suggested as the best model.

Maximum accuracy – 92.94% is achieved for LDA with model building time of 1 second. Similar level of accuracy is achieved with QDA (92.12%) with almost similar model building time. We conclude, that for this data set LDA and QDA performed equally well. However, for broader applications of human activity recognition, QDA can perform better as it gives scope for considering non-linear boundary.

3. LITERATURE REVIEW

1. G. Chetty, M. White, F. Akther, Smart Phone Based Data Mining For Human Activity Recognition, Elsevier Procedia Computer Science - ICICT, 46 (2015) 1181-1187[3]

The paper presents data base of human activity with smartphone inertial sensors. The paper also cited the advantages of using smartphone -based sensor recognition over other special hardware set ups and body sensor networks. Chetty et al.[3] used information theory-based ranking of features where features are ranked using information gain. This preprocessing step was performed to deal with high dimensionality of features. They used Naïve Bayes Classifier, K-Means clustering, random forest, random committee and Lazy IBk classifier as machine learning models for classification. They found Lazy IBk classifier (accuracy – 97.89%) and Random Forest (accuracy – 96.30%) to be most accurate with lowest model building times.

2. "Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine" by Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz [4]

Support Vector Machines (SVM) have confirmed successful application in heterogeneous types of recognitions. However, it is computationally expensive. The paper proposed a novel hardware-friendly approach of SVM for multiclass classification. The method uses standard SVM along with fixed point arithmetic for cost reduction. Comparing this method to the traditional SVM, it showed a significant improvement with respect to computational costs. Hardware-Friendly SVM showed an accuracy of 89.0% which is comparable to accuracy of traditional SVM (89.3%). The paper suggests using fixed point calculations in Activity Recognition as it has less power consumption, memory and processing time.

3. Unsupervised learning for human activity recognition using smartphone sensors by Yongjin Kwon, Kyuchang Kang, Changseok Bae [5]

This paper discusses unsupervised learning methods for human activity recognition when some activities are unknown. Mixture of Gaussian Method (GMM), K-Means clustering and Hierarchical clustering with average linkage were used. Kwon et al. reported that GMM performed with an accuracy of 100% when number of classes (K) were known. When K was unknown, K-Means clustering showed 71.98% accuracy and Hierarchical Clustering showed 79.98% accuracy.

4. MOTIVATION AND PROJECT SCOPE

Motivation:

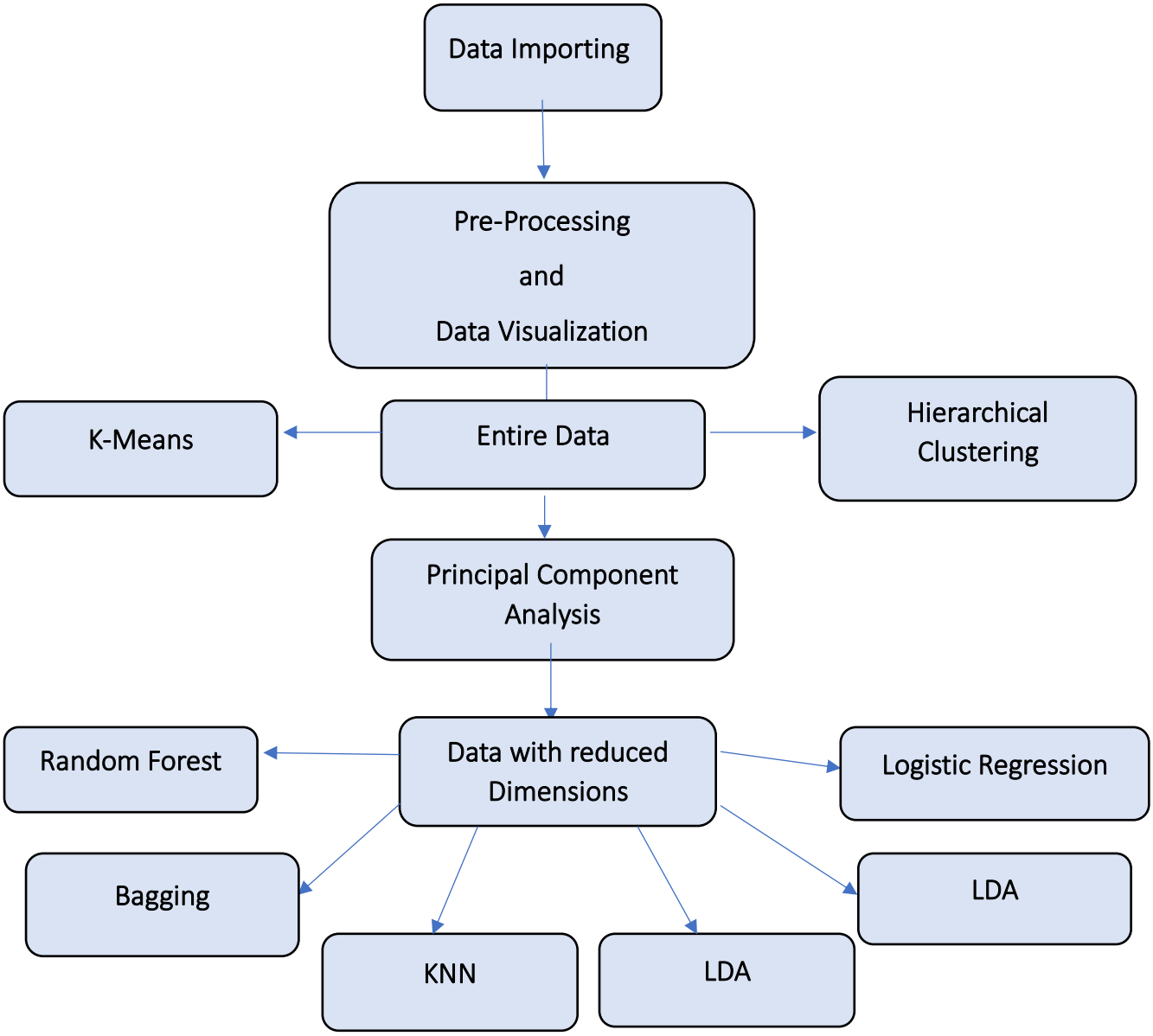
1. Literature review confirmed the need for dimension reduction technique as number of features is very large i.e. 561 features. Principle Component Analysis stands as one of the powerful techniques for this purpose.
2. Unsupervised learning models had inferior performance than traditional supervised learning methods. Hence, we scope usage of these techniques to data visualization.
3. LDA and QDA can potentially perform well as they are based on Naïve Bayes Classifier. Assuming the normality of data, we aim at building LDA and QDA models and check for accuracies and model building times.
4. Taking inspiration from Lazy Learning techniques, we plan to use KNN algorithm and check its effectiveness.
5. Random Forest is reported to have good testing accuracy and less model building time. We aim at checking its accuracy after using PCA as dimension reduction technique.
6. Traditional SVM can be a good model but given the difficulty of computational cost and tuning of various parameters like cost and gamma, development of hardware friendly SVM is out of scope of this project.

Scope:

The scope of the project includes:

1. Pre-processing: Data compilation and cleaning to include all the relevant data
2. Exploratory Data Analysis
3. Dimension Reduction: Dimension reduction techniques like PCA were used
4. Classification Models: Development of classification models like Logistic Regression, LDA, QDA, KNN, Random Forests and Bagging
5. Model comparison: Selecting the best model by comparison based on the classification accuracy and the computational time

5. PROJECT FLOW



Data Importing: Data has been imported from two different files available on Kaggle in form of train data and test data.

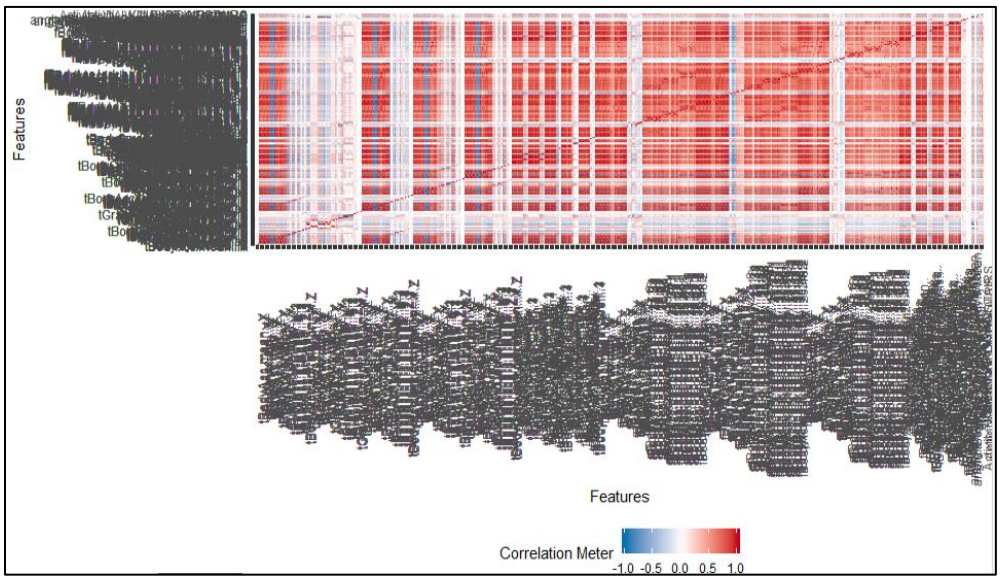
Preprocessing: Basic data cleaning and preprocessing activities were carried out to check the data but was not so much significant as data was very clean and optimized.

Dimension Reduction: Unsupervised learning method Principal Component Analysis was leveraged to reduce dimensions to explain about 95% of variance and reduce computation time.

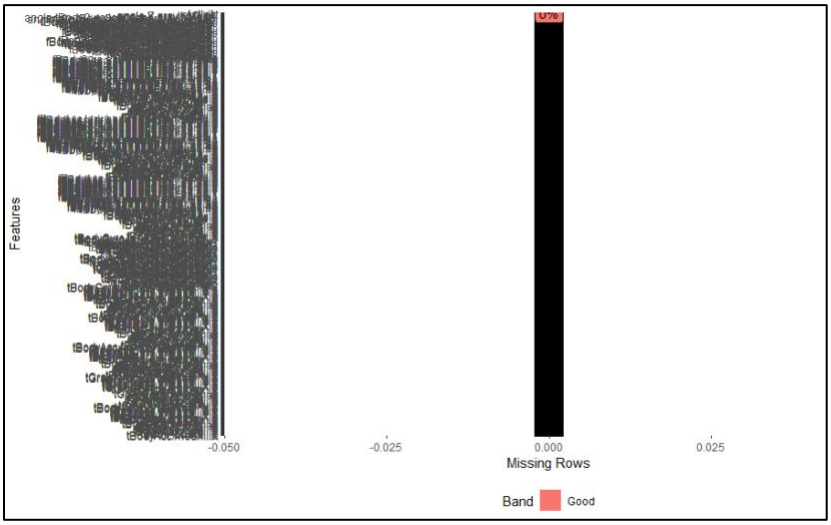
Model Fitting: Classic Machine learning models ranging from LDA, QDA to tree-based methods like Random forest and Bagging have been fitted to compare accuracy and computational time.

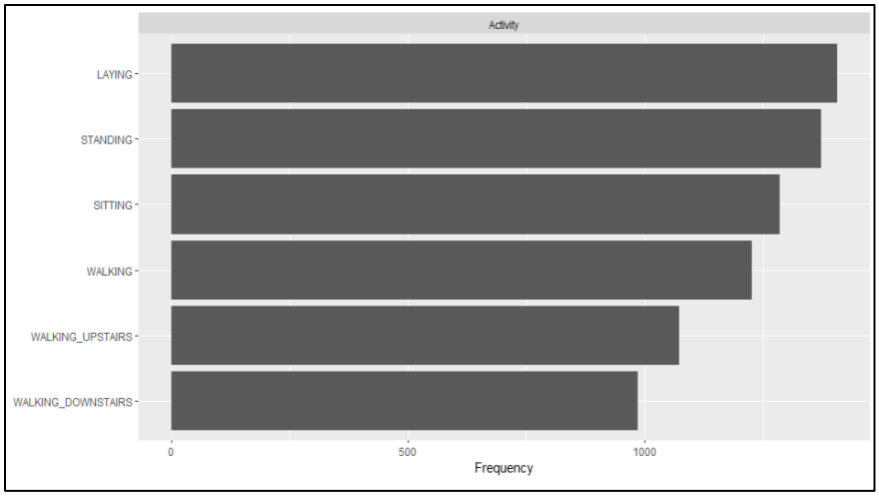
6. PREPROCESSING:

Correlation was observed for the data sets. The technique aided in predicting the highly correlated predictors easily.



The data set consisted of many variables making it essential to check for any missing data. As observed from the plot, no missing data was found.

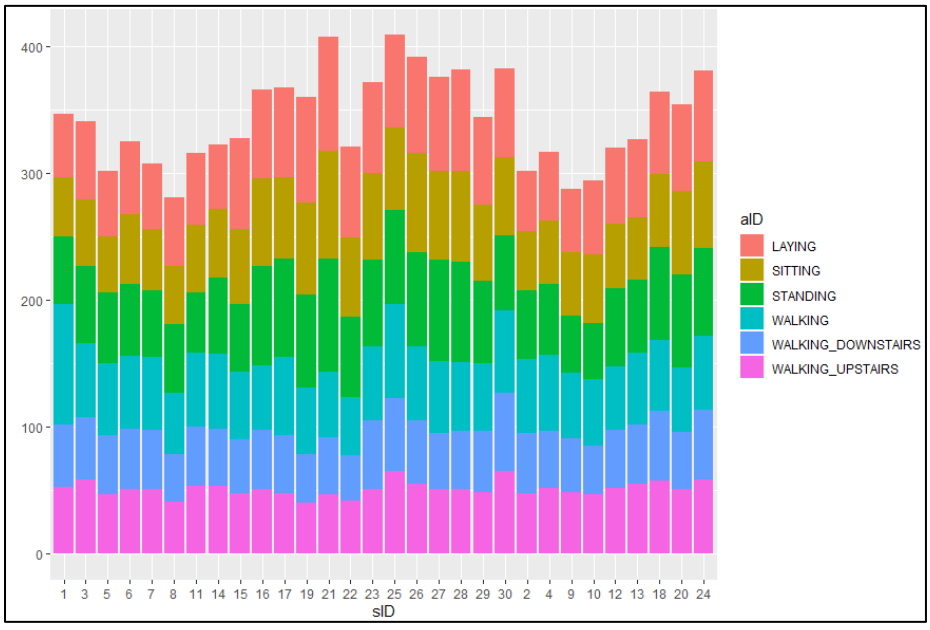




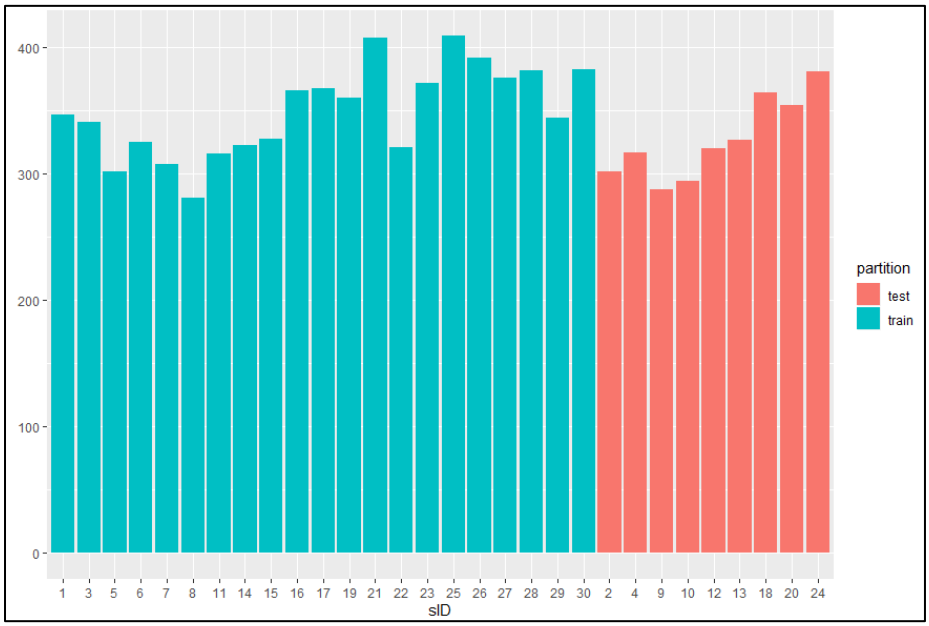
Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach for analyzing data sets mainly by visual methods (mostly graphical) to perform initial investigations on data and summarizing main characteristics to discover patterns, spot anomalies.

The plot helps to visualize that the 30 participants performed six activities each (walking, walking upstairs, walking downstairs, sitting, standing, laying) for a certain amount of time.

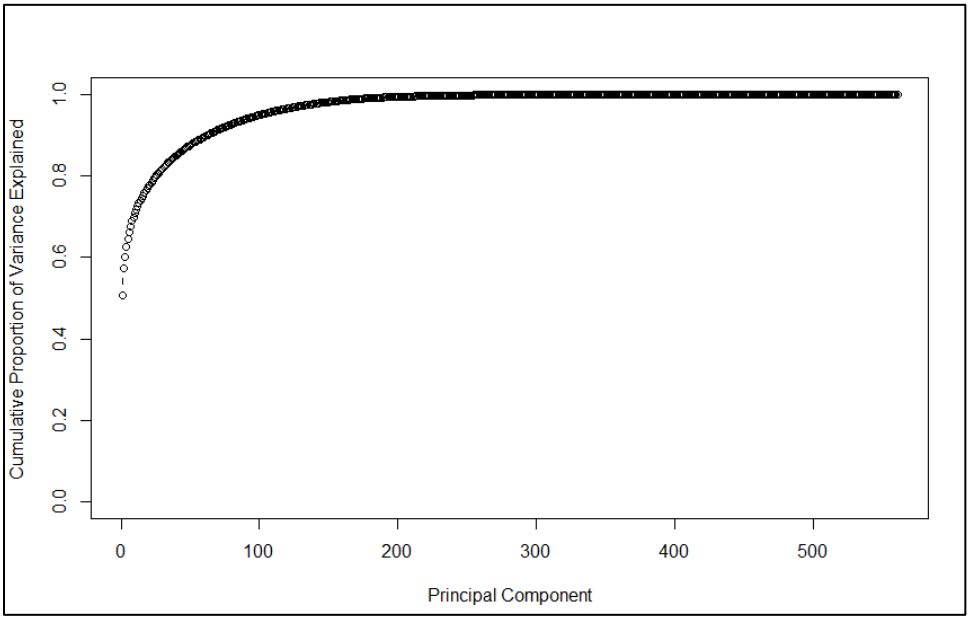


The plot depicts the split between the training and the test data. 70% of the data was used as training data and 30% was used as test data.



1. Principle Component Analysis (PCA):

Given the large number of predictors in the data set, dimensional reduction is a viable approach. PCA was used to reduce the dataset feature space to a smaller set explaining majority of the variance in the model. Firstly, PCA was applied on the training data set.



[1]	50.78117	57.36185	60.16829	62.67224	64.56053	66.28453	67.65554	68.85462	69.85048	70.81557	71.67562
[12]	72.47590	73.23990	73.88523	74.51755	75.11727	75.70402	76.27943	76.84735	77.37465	77.87501	78.36342
[23]	78.84162	79.31019	79.75948	80.18050	80.59848	81.00405	81.39258	81.77960	82.14555	82.50011	82.84805
[34]	83.18524	83.51491	83.84313	84.16366	84.45927	84.74600	85.03107	85.29984	85.56546	85.82886	86.08771
[45]	86.33677	86.58372	86.82440	87.06052	87.29080	87.51836	87.73853	87.95200	88.15970	88.36220	88.56198
[56]	88.75973	88.95400	89.14424	89.33230	89.51852	89.69998	89.87737	90.05345	90.22673	90.39747	90.56602
[67]	90.73164	90.89474	91.05318	91.21067	91.36370	91.51590	91.66404	91.81011	91.95478	92.09754	92.23619
[78]	92.37178	92.50649	92.63957	92.77038	92.90062	93.02885	93.15385	93.27682	93.39838	93.51755	93.63498
[89]	93.75027	93.86341	93.97401	94.08333	94.19125	94.29724	94.39995	94.50204	94.60276	94.70119	94.79766
[100]	94.89150	94.98384	95.07558	95.16547	95.25497	95.34249	95.42954	95.51539	95.59886	95.68162	95.76426
[111]	95.84541	95.92524	96.00252	96.07834	96.15339	96.22620	96.29802	96.36902	96.43880	96.50667	96.57311
[122]	96.63814	96.70212	96.76520	96.82735	96.88888	96.94913	97.00883	97.06712	97.12468	97.18113	97.23688
[133]	97.29144	97.34537	97.39831	97.45088	97.50239	97.55291	97.60313	97.65202	97.70051	97.74798	97.79494
[144]	97.84105	97.88708	97.93211	97.97638	98.02014	98.06311	98.10537	98.14699	98.18790	98.22862	98.26843
[155]	98.30721	98.34579	98.38377	98.42067	98.45695	98.49239	98.52716	98.56185	98.59536	98.62863	98.66124

The number of PCAs to be used can be determined from the PVE (Percentage of Variance Explained) plot. It can be observed that around 95% of the variance is explained by the first 102 PCs. Hence first 102 PCs are selected for analysis as there is no significant increase in the variance explained by increasing the number of components.

Applying PCA on Test Data:

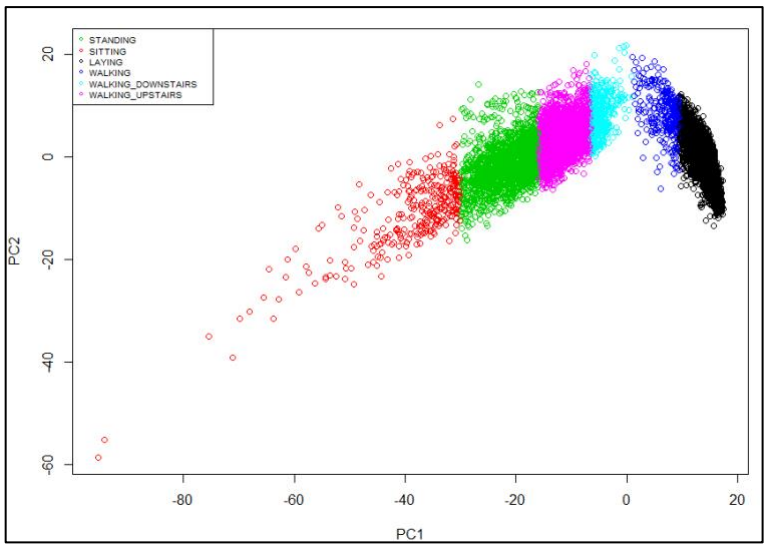
To ensure that the direction of variance is same in both the train and the test data, predict function was used on the test data. Then, a data frame consisting of the test and train data frames was formed for further analysis.

2. K- means Clustering:

It is an algorithm which is used for partitioning a dataset in K distinct, non-overlapping clusters. We must first specify the desired number of clusters K; then the K-means approach will assign each observation to exactly one of the K clusters by the following algorithm:

1. Randomly assigning a cluster for each observation.
2. Iterate until convergence is obtained:
    - Computing the cluster centroid for each k clusters.
    - Assigning each observation to the cluster whose centroid is the closest

But for K- means, we need to specify the to specify the value of number of clusters, k and many times the algorithm is stuck at the local optimum and does not give the global optimum.



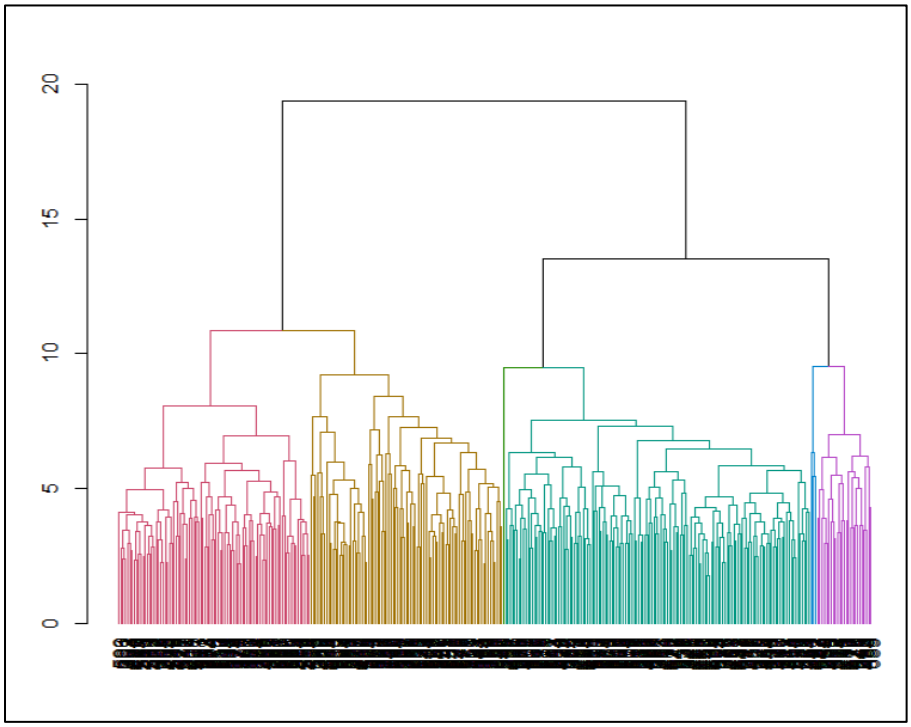
Observation: 6 distinct, non-overlapping clusters

3. Hierarchical clustering:

It is an alternative approach which does not need to us to specify the value of K and gives a better visual representation of the data; a dendogram. Hierarchical clustering can be performed in 2 ways:

1. Agglomerative clustering or 'Bottom up' approach
2. Divisive clustering or 'Top bottom' approach

The hierarchical clustering dendogram is obtained by an algorithm using a dissimilarity measure (like Euclidean distance).



- Bottom up approach is used commonly in practice.
- Here we used bottom up approach as it gives higher accuracy at lower levels.
- We observed 6 distinct classes when Hierarchical clustering was used on subject 3 indicating the 6 activities performed by the individual.

We decided to move forward with Principal Component Analysis (PCA) as it is the most viable option to reduce the number of predictors.

7.1 Model 1 : Logistic Regression

Logistic regression is a statistical model used to predict probability of a binary response when the outcome should be categorical. Logistic Regression uses Logistic Function Maximum Likelihood function to fit the model.

This can be extended to more than two classes: Multinomial Logistic Regression (One Vs All).

Multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes. When using multinomial logistic regression, separate odds are determined for all independent variables of each category of dependent variables.

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p,$$

[2]

R-Workspace

```
> pred.mlr = predict(fit.mlr,test.pca)
> CM = table(pred.mlr,test.pca$test.y)
> CM

pred.mlr              LAYING SITTING STANDING WALKING WALKING_DOWNSTAIRS
  LAYING                 529       0        0       0                   0
  SITTING                  0     420       33       0                   0
  STANDING                 8      68      499       0                   1
  WALKING                  0       0        0     484                  21
  WALKING_DOWNSTAIRS       0       0        0       5                 369
  WALKING_UPSTAIRS         0       3        0       7                  29

pred.mlr              WALKING_UPSTAIRS
  LAYING                             1
  SITTING                            5
  STANDING                           0
  WALKING                           64
  WALKING_DOWNSTAIRS                 5
  WALKING_UPSTAIRS                 396
```

```
> acc = (sum(diag(CM)))/sum(CM)
> acc
[1] 0.915168
```

Model outcome: Accuracy – 91.51%

7.2 Model 2: LDA-Linear Discriminant Analysis

Discriminant analysis is popular when classification is to be done in more than 2 classes.
LDA classifier is based on Bayes Rule.
Density function of predictors: Multivariate Gaussian Distribution with class specific mean vector (μk) an d covariance matrix (∑) which is common to all K classes. LDA classifies observation to a class where δk (x) is largest. LDA explicitly attempts to model the difference between the class of the data.

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

[2]

R-Workspace

```
> pred<-predict(lda1,test.pca)
> CM = table(pred$class,test.pca$test.y)
> CM

                    LAYING SITTING STANDING WALKING WALKING_DOWNSTAIRS
  LAYING               537       0        0       0                   0
  SITTING                0     420       45       0                   0
  STANDING               0      70      487       0                   0
  WALKING                0       0        0     491                  15
  WALKING_DOWNSTAIRS     0       0        0       5                 366
  WALKING_UPSTAIRS       0       1        0       0                  39

                    WALKING_UPSTAIRS
  LAYING                           0
  SITTING                          0
  STANDING                         0
  WALKING                         28
  WALKING_DOWNSTAIRS               5
  WALKING_UPSTAIRS               438
```

```
> acc = (sum(diag(CM)))/sum(CM)
> acc
[1] 0.9294197
```

Model outcome: Accuracy – 92.9%.

7.3 Model 3: QDA- Quadratic Discriminant Analysis

The fundamental idea behind QDA is to find the posterior probability of the observation belonging to kth class using Bayes' theorem. It assumes the observations from each class follows Normal distribution. QDA assumes quadratic decision boundary. Hence, this method of classification lies in between nonparametric methods and linear methods.

Density function of predictors: Multivariate Gaussian Distribution with class specific mean vector (μk) and class specific covariance matrix (∑). QDA classifies observation to a class where δk(x) is largest.

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)+\log\pi_k \\ &= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + \log\pi_k\end{aligned}$$ [2]

R-Workspace

```
> pred<-predict(qda1,test.data)
> CM = table(pred$class,test.pca$test.y)
> CM

                    LAYING SITTING STANDING WALKING WALKING_DOWNSTAIRS
  LAYING               537       5        1       0                   0
  SITTING                0     376       24       0                   0
  STANDING               0     108      501       0                   0
  WALKING                0       0        0     451                   0
  WALKING_DOWNSTAIRS     0       0        0      42                 395
  WALKING_UPSTAIRS       0       2        6       3                  25

                    WALKING_UPSTAIRS
  LAYING                         0
  SITTING                        0
  STANDING                       0
  WALKING                        6
  WALKING_DOWNSTAIRS            10
  WALKING_UPSTAIRS             455
```

```
> acc = (sum(diag(CM)))/sum(CM)
> acc
[1] 0.9212759
```

Model outcome: Accuracy – 92.12%.

7.4 Model 4: K-Nearest Neighbors

KNN applies Bayes rule and classifies the test observation to a class with the largest probability i.e. KNN classifies the observations based on the maximum number of neighbors belonging to a specific class.

The choice of K has a drastic effect on the classifier obtained and hence it is the tuning parameter.

When the value of 'K' is smaller the decision boundary is overly flexible resulting in high variance and low bias. But as value of 'K' increases, the boundary becomes less flexible (almost linear) reducing the variance at the expense of high bias.

Hence it is essential to take into consideration the bias-variance tradeoff and select the optimal value of 'K'. We considered K= 5, 10, 25, 50, 100 and calculated the corresponding accuracies.

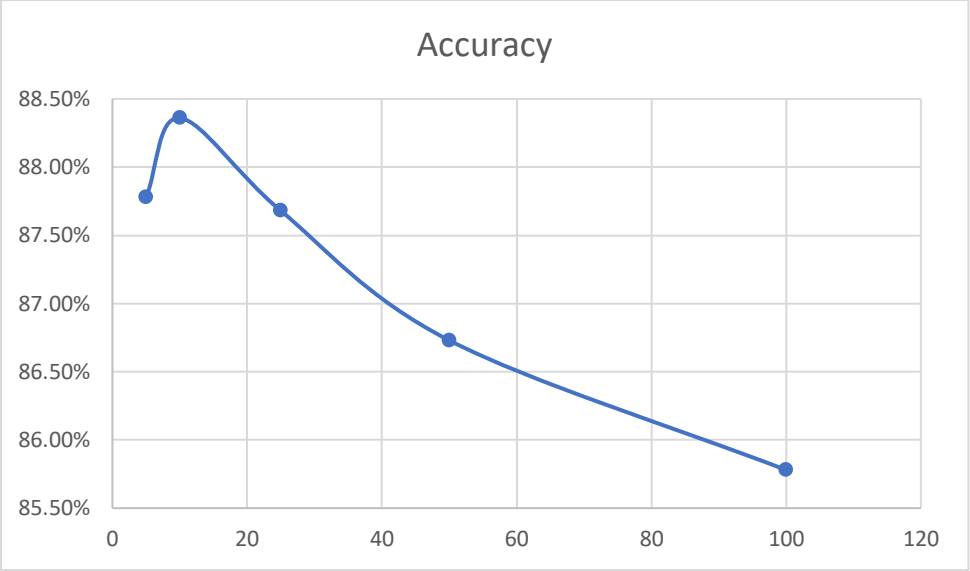
R-Workspace

```
> knn.fit5=knn(train.data,test.data,train.y,k=5)
> summary(knn.fit5)
            LAYING             SITTING            STANDING             WALKING
               505                 423                 631                 577
WALKING_DOWNSTAIRS   WALKING_UPSTAIRS
               339                 472
> CM = table(knn.fit5, test.y)
> acc = (sum(diag(CM)))/sum(CM)
> acc
[1] 0.8778419
```

```
> knn.fit10=knn(train.data,test.data,train.y,k=10)
> summary(knn.fit10)
            LAYING             SITTING            STANDING             WALKING
               509                 421                 629                 574
WALKING_DOWNSTAIRS   WALKING_UPSTAIRS
               335                 479
> CM = table(knn.fit10, test.y)
> acc = (sum(diag(CM)))/sum(CM)
> acc
[1] 0.8836105
```

```
> set.seed(1)
> knn.fit50=knn(train.data,test.data,train.y,k=50)
> summary(knn.fit10)
            LAYING             SITTING            STANDING             WALKING
               509                 421                 629                 574
WALKING_DOWNSTAIRS   WALKING_UPSTAIRS
               335                 479
> CM = table(knn.fit50, test.y)
> acc = (sum(diag(CM)))/sum(CM)
> acc
[1] 0.8673227
```

```
> set.seed(1)
> knn.fit100=knn(train.data,test.data,train.y,k=100)
> summary(knn.fit100)
            LAYING             SITTING            STANDING             WALKING
               497                 342                 718                 609
WALKING_DOWNSTAIRS   WALKING_UPSTAIRS
               301                 480
> CM = table(knn.fit100, test.y)
> acc = (sum(diag(CM)))/sum(CM)
```



Model outcome: Accuracy – 88.36% for k=10

Tree Based Methods:

Tree based algorithms involve arranging or segmenting the predictor space into a number of distinct regions.

A classification tree is used to predict a qualitative response. The best prediction is the most commonly occurring class of training observations in the region to which it belongs. Decision trees generally have high variance, so in order to have models with low variance, tree improving methods like Bagging, Random Forest and Boosting are implemented.

7.5 Model 5:Bagging

Bagging is also known as Bootstrap Aggregating. Bagging involves generating different bootstrapped samples and constructing trees using these training sets. The resulting predictions are averaged and combined into an aggregated prediction. Bagging model is considered as a great tool based on two things:

- Bootstrapping results in plenty of training datasets
- Averaging the predictions reduces variance

R-Workspace

```
> bag.fit.tree

Call:
 randomForest(formula = factor(train.y) ~ ., data = train.pca,      mtry = 10, ntree = 500, imp
ortance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 10

        OOB estimate of  error rate: 5.77%
Confusion matrix:
                   LAYING SITTING STANDING WALKING WALKING_DOWNSTAIRS WALKING_UPSTAIRS
LAYING               1406       1        0       0                  0                0
SITTING                30    1075      180       0                  0                1
STANDING                0      99     1275       0                  0                0
WALKING                 0       0        0    1198                 18               10
WALKING_DOWNSTAIRS      0       0        0      11                954               21
WALKING_UPSTAIRS        0       0        0      31                 22             1020
                    class.error
LAYING              0.0007107321
SITTING             0.1640746501
STANDING            0.0720524017
WALKING             0.0228384992
WALKING_DOWNSTAIRS 0.0324543611
WALKING_UPSTAIRS    0.0493942218
```

```
> pred.bag <- predict(bag.fit.tree, newdata=test.data)
> Accuracy <- mean(pred.bag == test.y)
> Accuracy
[1] 0.889379
```

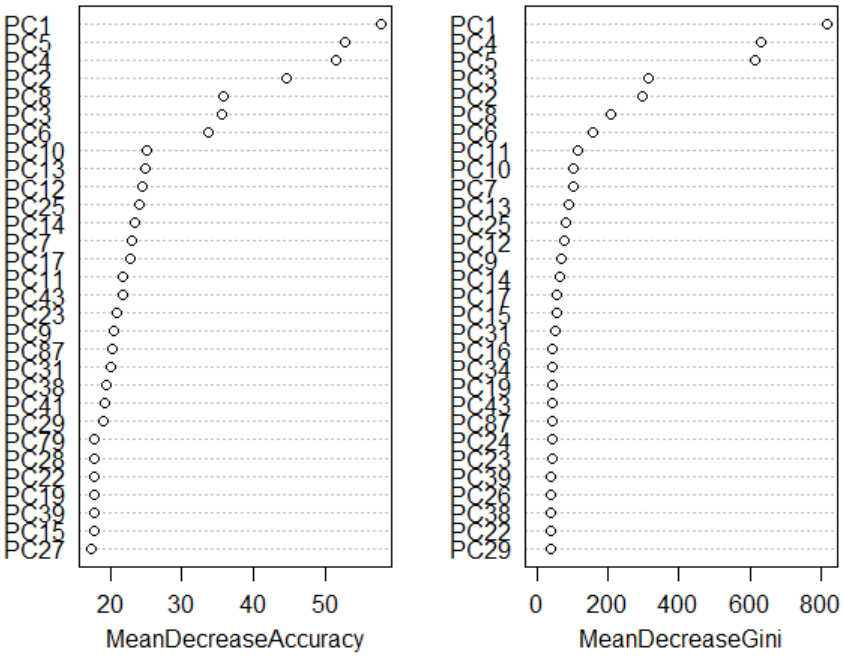
Model outcome: Accuracy – 88.93%

One of the simple approaches to estimate the bagged model is to check for the Out-Of-Bag error in the model eliminating the need to perform cross-validation. OOB score is computed as the number of correctly predicted rows from the out of bag sample. In the above model, the OOB error is found to be 5.77%, which means the OOB accuracy is 94.23%. The model accuracy is 88.93% on test data for number of trees=300.

From the graph of variable importance, it can be inferred that when PC1, PC5 and PC4 are omitted from training sample, the accuracy of the model is found to be decreased by an average of 53%.

This implies that these variables are significant to the model for better performance.

```
> varImpPlot(bag.fit.tree)
```



7.6 Model 6:Random Forest

Random Forest is built on the idea of bagging, but it provides an improvement because it de-correlates the trees which leads to more reduction in variance. Hence, this classification algorithm averages the predictions from all the uncorrelated classification trees which are generated on different bootstrapped training data sets.

For each tree, at each split, the number of predictors to be sampled are chosen using square root of total predictors under consideration. In the project, 10 predictor PCs are considered among the 102 principal components and the number of trees considered are 300.

R-Workspace

```
> fit.rf

Call:
 randomForest(formula = factor(train.y) ~ ., data = train.pca,      mtry = rf, ntree =
300, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 10

        OOB estimate of  error rate: 6.13%
Confusion matrix:
                  LAYING SITTING STANDING WALKING WALKING_DOWNSTAIRS WALKING_UPSTAIRS
LAYING              1403       2        1       0                  0                1
SITTING               23    1074      188       0                  0                1
STANDING               0     111     1263       0                  0                0
WALKING                0       0        0    1200                 14               12
WALKING_DOWNSTAIRS     0       0        0      11                948               27
WALKING_UPSTAIRS       0       1        0      28                 31             1013
                  class.error
LAYING            0.002842928
SITTING           0.164852255
STANDING          0.080786026
WALKING           0.021207178
WALKING_DOWNSTAIRS 0.038539554
WALKING_UPSTAIRS   0.055917987
```

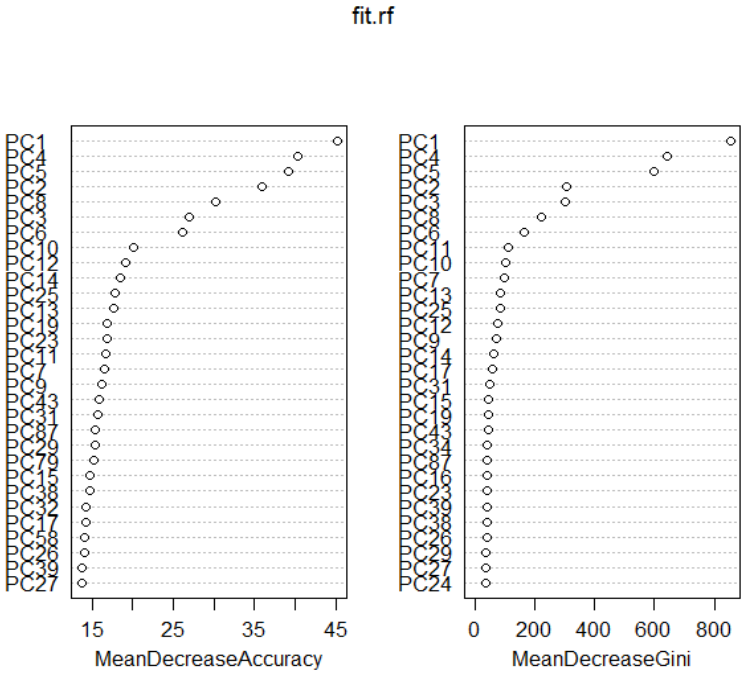
```
> fitrf.pred <-predict(fit.rf, newdata=test.data)
> Accuracy <- mean(fitrf.pred == test.y)
> Accuracy
[1] 0.8910757
```

Model outcome: Accuracy – 89.10%

The error in prediction is calculated in terms of Out-Of-Bag error. It is 6.13%, this implies that the Out-Of-Bag accuracy is 93.87%. Model reported testing accuracy of 89.1%.

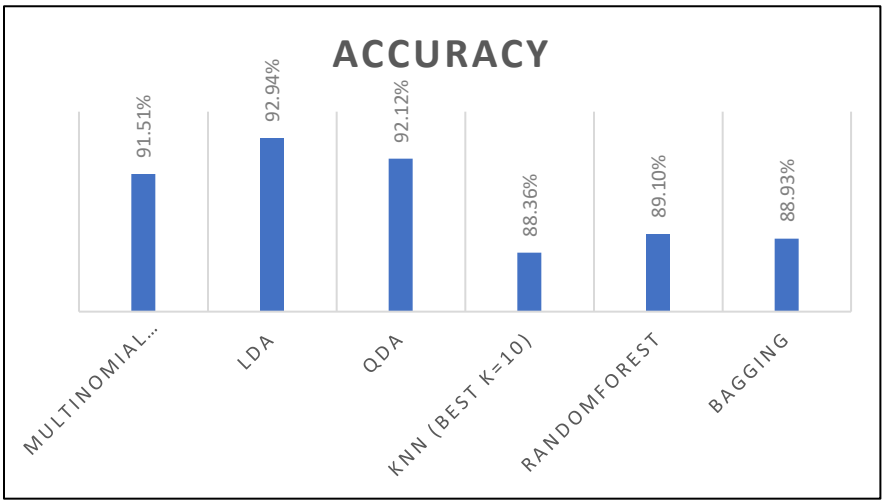
Across all the trees considered in the random forest, the important PCs are identified from the following graph of Importance. When PC1, PC4 and PC5 are omitted from training sample, the accuracy of the model is found to be decreased by an average of 45%. Hence, PC1, PC4 and PC5 are the most important PCs to consider for better accuracy of the model.

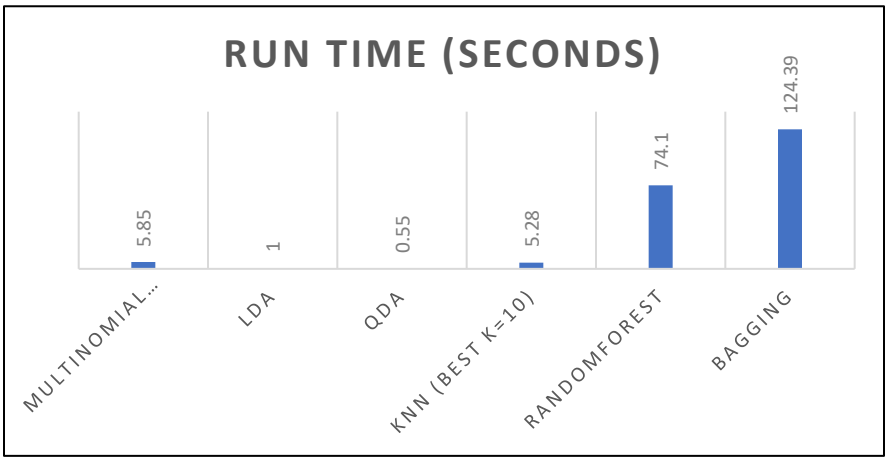
```
> importance(fit.rf)
```



8. MODEL COMPARISON

The models are compared based on two parameters, accuracy and the processing times.





For finding out the model building times, we tried using microbenchmark library. However estimating time using microbenchmark is computationally expensive for Logistic Regression and tree based methods using Microbenchmark. We then computed the model building time using Sys.time() function. Above comparison is based on this method.

From the graphs, it is seen that the classification accuracy and the computational time is optimal for the Linear Discriminant Analysis (LDA) model and Quadratic Discriminant Analysis (QDA) model. The highest obtained accuracy is 92.94% for LDA followed by 92.19% for QDA. Some loss in accuracy may be explained by the number of PCs considered; the PCs considered explain 95% variance in the data leading to loss of 5% data.

We can conclude that LDA is the best model for the given data set for its high accuracy and low computational time.

9. REFERENCES:

1. Data Set: https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

2. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An Introduction to Statistical Learning: With Applications in R. Corrected edition New York: Springer, 2013.

3. G. Chetty, M. White, F. Akther, Smart Phone Based Data Mining For Human Activity Recognition, Elsevier Procedia Computer Science - ICICT, 46 (2015) 1181-1187

4. 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: Bravo J., Hervás R., Rodríguez M. (eds) Ambient Assisted Living and Home Care. IWAAL 2012. Lecture Notes in Computer Science, vol 7657. Springer, Berlin, Heidelberg

5. Unsupervised learning for human activity recognition using smartphone sensors by Yongjin Kwon, Kyuchang Kang, Changseok Bae

10. APPENDIX:

R-CODE:

```
############################ libraries ################################

library(ggplot2)
library(dendextend)
library(caret)
library(MASS)
library(DataExplorer)
library(class)
library(ISLR)
library(caret)
library(lattice)
library(nnet)
library(randomForest)
library(microbenchmark)
install.packages('e1071')
library(e1071)

############################ Data extraction ################################


train=read.csv("/Users/vaishnavipatki/Downloads/Project 2/Trial/train.csv",header = TRUE,sep =",")
test=read.csv("/Users/vaishnavipatki/Downloads/Project 2/Trial/test.csv",header = TRUE,sep=",")

summary(train)
attach(train)
head(train)
sum(is.null(train))
sum(is.null(test))

############################ Data combining ################################

mydata=rbind(train,test)
train.par=train
train.par$partition="train"
train.par=transform(train.par,sID=factor(subject),aID=factor(Activity))

test.par=test
test.par$partition="test"
test.par=transform(test.par,sID=factor(subject),aID=factor(Activity))

mydata.partition=rbind(train.par,test.par)

########################## Data Vizualization ############################

qplot(data = mydata.partition,x=sID,fill=aID)
```

```
qplot(data = mydata.partition,x=sID,fill=partition )
plot_missing(train)
plot_bar(train)
plot_correlation(train)

###################### Splitting the data ########################
set.seed(1)
train.x <- train[, -c(562:563)]
train.y <- train["Activity"]
test.x <- test[, -c(562:563)]
test.y <- test["Activity"]

###################### Applying PCA on train ##############################
set.seed(1)
pc.x <- prcomp (train.x , scale =TRUE)
pr.var =pc.x$sdev ^2
pve=pr.var/sum(pr.var)
plot(pve , xlab="Principal Component", ylab="Proportion of Variance Explained", ylim=c(0,1) ,type="b")
plot(cumsum (pve), xlab=" Principal Component ", ylab ="Cumulative Proportion of Variance Explained ",
 ylim=c(0,1) , type="b")


cumsum (pve)*100

train.data <- data.frame(pc.x$x[, 1:102])
train.pca = data.frame(pc.x$x[, 1:102], train.y)


################ PCA on test #################
set.seed(1)
test.data <- predict(pc.x, newdata=test.x)
test.data <- as.data.frame(test.data)
test.data <- test.data[,1:102]

test.pca= data.frame(test.data, test.y)

######### Combining test and train data frames after PCA#####

my.data = rbind(train.data,test.data)

########################## Hierarchical cluster ######################

#we perform hierachical clustering on subject 3 for vizualization

subject.3=subset(mydata,subject==3)
distances=dist(subject.3[,-c(562:563)])
hcluster=hclust(distances,method = "complete")
dendr=as.dendrogram(hcluster)
```

```
dendr=color_branches(dendr,k=6)
plot(dendr)


######################## K means ##########################

k.out= kmeans(c(my.data[,1],my.data[,2]), 6,nstart=25)
kcluster=k.out$cluster
plot(my.data[,1],my.data[,2],col=kcluster,xlab='PC1',ylab='PC2')
legend("topleft",legend=unique(train.pca$train.y),col=unique(train.pca$train.y),cex=0.6,pch=1)


#################### Logistics regression ######################
t1<-Sys.time()
set.seed(1)
fit.mlr = multinom(train.y~., data =train.pca)
fit.mlr
pred.mlr = predict(fit.mlr,test.pca)
CM = table(pred.mlr,test.pca$test.y)
CM
acc = (sum(diag(CM)))/sum(CM)
acc
t2<-Sys.time()
print(t2-t1)

Logistic_function<-function(train.y,train.pca,test.pca){
  set.seed(1)
  fit.mlr = multinom(train.y~., data =train.pca)
  fit.mlr
  pred.mlr = predict(fit.mlr,test.pca)
  CM = table(pred.mlr,test.pca$test.y)
  CM
  acc = (sum(diag(CM)))/sum(CM)
  acc
}

############### LDA ##############
LDA_function<-function(train.y,train.pca,test.pca){
  set.seed(1)
  lda1<-lda(train.y~., data = train.pca)
  lda1
  pred<-predict(lda1,test.pca)
  CM = table(pred$class,test.pca$test.y)
  CM
  acc = (sum(diag(CM)))/sum(CM)
  acc
}
t1<-Sys.time()
```

```
set.seed(1)
lda1<-lda(train.y~., data = train.pca)
lda1
pred<-predict(lda1,test.pca)
CM = table(pred$class,test.pca$test.y)
CM
acc = (sum(diag(CM)))/sum(CM)
acc
t2<-Sys.time()
print(t2-t1)
############ QDA ##############
QDA_function<-function(train.y,train.pca,test.pca){
 set.seed(1)
 qda1<-qda(train.y~., data =train.pca)
 qda1
 pred<-predict(qda1,test.data)
 CM = table(pred$class,test.pca$test.y)
 CM
 acc = (sum(diag(CM)))/sum(CM)
 acc

}
t1<-Sys.time()
set.seed(1)
qda1<-qda(train.y~., data =train.pca)
qda1
pred<-predict(qda1,test.data)
CM = table(pred$class,test.pca$test.y)
CM
acc = (sum(diag(CM)))/sum(CM)
acc
t2<-Sys.time()
print(t2-t1)
#################### KNN ########################

## K=5 #########
set.seed(1)
knn.fit5=knn(train.data,test.data,train.y,k=5)
summary(knn.fit5)
CM = table(knn.fit5, test.y)
acc = (sum(diag(CM)))/sum(CM)
acc
### K=10 ########
t1<-Sys.time()
set.seed(1)
knn.fit10=knn(train.data,test.data,train.y,k=10)
summary(knn.fit10)
CM = table(knn.fit10, test.y)
```

```
acc = (sum(diag(CM)))/sum(CM)
acc
t2<-Sys.time()
print(t2-t1)
### K=50##########
set.seed(1)
knn.fit50=knn(train.data,test.data,train.y,k=50)
summary(knn.fit10)
CM = table(knn.fit50, test.y)
acc = (sum(diag(CM)))/sum(CM)
acc

##### K=100 #########
set.seed(1)
knn.fit100=knn(train.data,test.data,train.y,k=100)
summary(knn.fit100)
CM = table(knn.fit100, test.y)
acc = (sum(diag(CM)))/sum(CM)
acc

##### K=25 #########
set.seed(1)
knn.fit100=knn(train.data,test.data,train.y,k=25)
summary(knn.fit100)
CM = table(knn.fit100, test.y)
acc = (sum(diag(CM)))/sum(CM)
acc

KNN_function<-function(train.data,test.data,train.y){
 set.seed(1)
 knn.fit10=knn(train.data,test.data,train.y,k=10)
 summary(knn.fit10)
 CM = table(knn.fit10, test.y)
 acc = (sum(diag(CM)))/sum(CM)
 acc

}

#################### Random Forests ######################
t1<-Sys.time()
set.seed(1)
rf <- round((102)^0.5)
fit.rf = randomForest(factor(train.y) ~ ., data=train.pca, mtry = rf, ntree=300, importance = TRUE)
fitrf.pred <-predict(fit.rf, newdata=test.data)
Accuracy <- mean(fitrf.pred == test.y)
Accuracy
t2<-Sys.time()
print(t2-t1)
```

```
importance(fit.rf)
varImpPlot(fit.rf)

RF_function<-function(train.y,test.y,train.pca,test.data){
 rf <- round((102)^0.5)
 fit.rf = randomForest(factor(train.y) ~ ., data=train.pca, mtry = rf, ntree=300, importance = TRUE)
 fitrf.pred <-predict(fit.rf, newdata=test.data)
 Accuracy <- mean(fitrf.pred == test.y)
 Accuracy
}

################### Bagging ###################
t1<-Sys.time()
bag.fit.tree <- randomForest(factor(train.y) ~., data=train.pca , mtry = 10, ntree = 500, importance =
 TRUE)
pred.bag <- predict(bag.fit.tree, newdata=test.data)
Accuracy <- mean(pred.bag == test.y)
Accuracy
t2<-Sys.time()
print(t2-t1)
importance(bag.fit.tree)
varImpPlot(bag.fit.tree)

Bagging_function<-function(train.y,test.y,train.pca,test.data){
 bag.fit.tree <- randomForest(factor(train.y) ~., data=train.pca , mtry = 10, ntree = 500, importance =
 TRUE)
 pred.bag <- predict(bag.fit.tree, newdata=test.data)
 Accuracy <- mean(pred.bag == test.y)
 Accuracy
}

#########Compare the run times#############
out<-microbenchmark(LDA_function(train.y,train.pca,test.pca),unit="ms")
summary(out)
out<-microbenchmark(QDA_function(train.y,train.pca,test.pca),unit="ms")
summary(out)
out<-microbenchmark(KNN_function(train.data,test.data,train.y),unit="ms")
summary(out)
#out<-microbenchmark(RF_function(train.y,test.y,train.pca,test.data),unit="ms")
#sbestummary(out)
#out<-microbenchmark(Bagging_function(train.y,test.y,train.pca,test.data),unit="ms")
#summary(out)
```