London School of Economics and Political Science

Department of Mathematics

Project Report
Human Activity Recognition

MA429 Data Mining

Mock Project

Submitted by

Akeel Ahamed (201316007)

Rudraksh Garg (201760430)

Lucas Straub (201761371)

March 15, 2018

Executive Summary

The goal of this project was to predict human posture by applying data mining techniques on an accelerometer data set. Multiple approaches were tested and two of them turned out to work very well: radial Support Vector Machines (SVM) as well as k-Nearest Neighbours (KNN). Both can predict human posture with an accuracy of over 99%. Whereas radial SVM had comparably huge computational time, KNN turned out to be much quicker. Therefore, this report suggests that KNN can be used for reliable Human Activity Recognition with accelerometers.

Contents

Introduction

Data mining enables Human Activity Recognition by applying certain methods on huge data sets. Accelerometers help to track human movements and provide the data needed. One exemplary data set which was published by Ugulino et al.[1] is used in this work. Multiple data mining techniques were applied on the data to predict the type of human movements.

The whole approach is explained in this report and was divided into the following sections: First, a sound understanding of the data set is given. Next, preliminary analysis results are provided including pre-processing steps. The successive section presents the actual application of the classification methods used and compares them according to performance measures. Finally, the results are interpreted, and an outlook is given in the conclusion.

Accelerometer Dataset

The Dataset is a collection of 165,633 observations recording 4 individuals (two men and two women) wearing four accelerometers which are positioned at waist, left thigh, right ankle and right arm. Each accelerometer simply measures the movement of the different parts of the body. The data is a collection of 8 hours of activity, 2 hours with each one of the 4 subjects. The other features include subject name (which was excluded from the beginning), gender, age, height, weight and Body Mass Index (BMI).

Finally, each instance also records a class which represents the posture of the subject in different positions that are:

- Sitting down
- Standing up
- Walking
- Standing
- Sitting

Data Pre-processing & Analysis

The data was imported from a CSV (Comma Separated Value) file into R. The initial steps involved checking whether there were any missing values in the dataset. There were no missing predictor values for any of the instances.

Hence, the next step involved looking for evidence of correlation between the different predictors and to get a general overview of how the different variables interact with each other.

[1] Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012.

This would allow us to see which variables influence the response independently. In this case, they are the different positions that the subject can take (as mentioned above). Thereafter, we may be able to eliminate the variables which aren't strongly correlated (and thus will not play a vital role in the classification of the subjects), in order to give us a greater prediction accuracy.

Feature Selection[2]

It is important that the features used in the classification analysis are not heavily correlated with each other as this can lead to the problem of multicollinearity (which will reduce the accuracy of the predictions). To determine the correlations, we found out the correlation matrix using the 'Caret Package' (specifically using the cor() and the findCorrelation() functions) and shortlisted the candidates which can be considered for elimination on the basis of their high correlation. The features: weight-BMI, height-BMI, x_2-y_2, y_2-z_2 were highly correlated (absolute correlation >0.9).

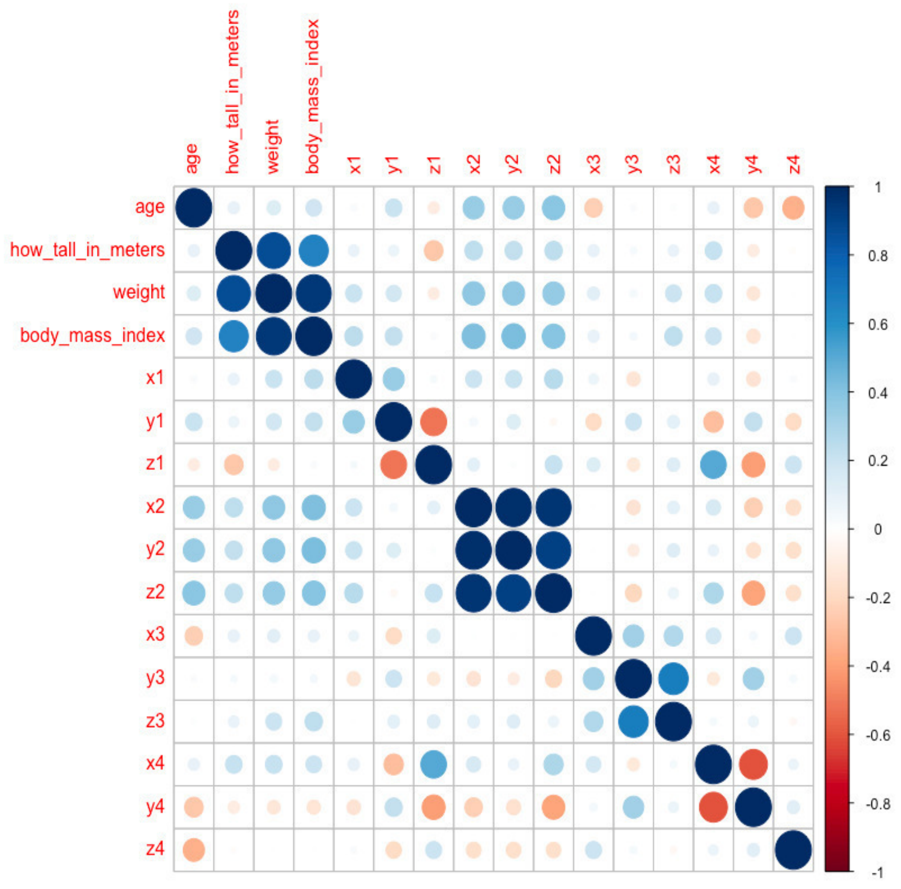


Figure 1: Correlation plot of predictors

[2] Feature Selection with the Caret R Package (Article) - https://machinelearningmastery.com/feature-selection-with-the-caret-r-package/

Some of these features were dropped when their relevance to the model was taken into consideration along with the rank of importance calculated using the Learning Vector Quantisation (LVQ) model. The varImp() function determines the importance of each variable.

Based on our findings, it was decided to do the entire analysis on the basis of data obtained from the four accelerometers (i.e. the x_i, y_i and z_i measures from accelerometer i) and specifically chosen predictors which were not highly correlated with each other. These specific predictors varied among the different methods discussed in the Application section.

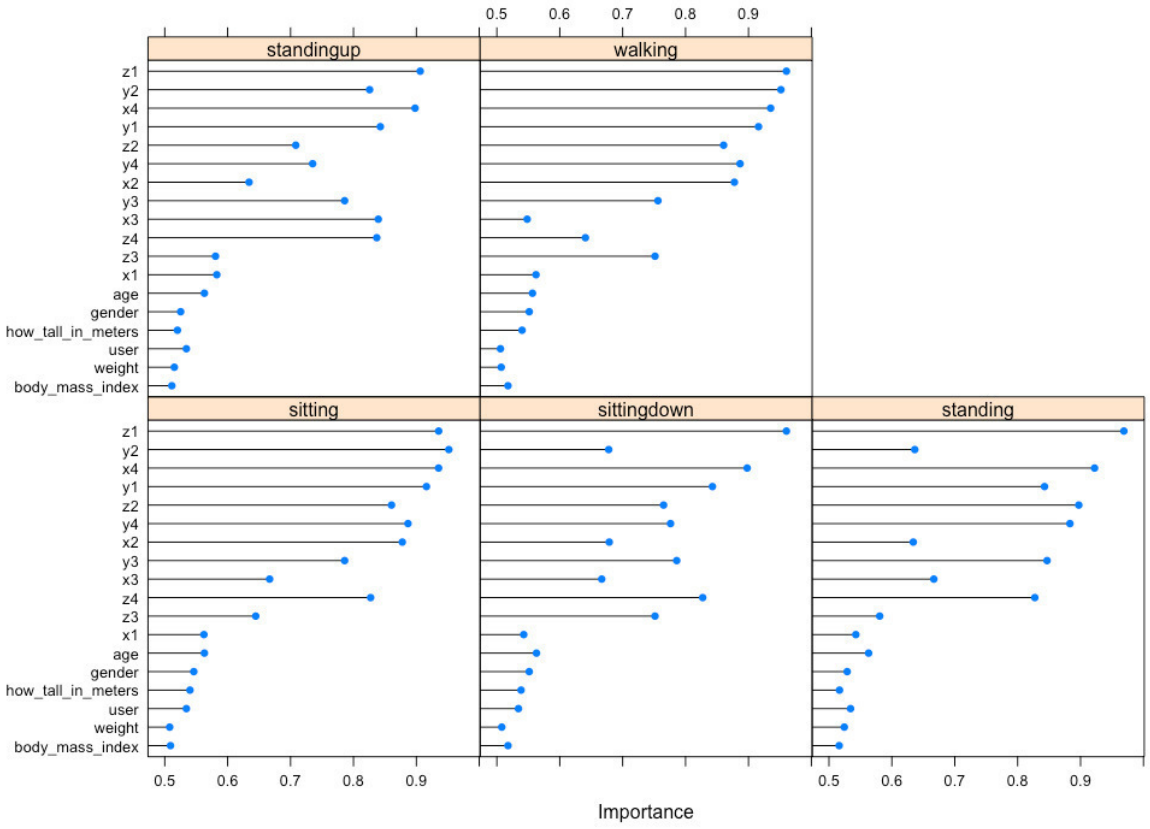


Figure 2: Ranking of predictors by level of importance

Data Splits and Subsets

At this point, the data was divided in training and testing sets. A random split was created at different points dividing training and testing data in ratios of 3:1 (for SVM) or 9:1 (for KNN) depending how well the model performed with different levels of training data. It was also ensured that the models were not over trained and were flexible enough to take into account the variance-bias trade-off.

For methods which are quite complex and take a lot of computational effort like Support Vector Machines, it made more sense to train and test on smaller subset first to determine the efficiency of the different methods. If a particular method performed well on the given data set, it was later trained and tested on the split of the entire dataset. This approach saved time which would otherwise be lost on training methods which couldn't perform well on the testing split.

Application and Results of Data Mining Techniques

Since the dependent variable is nominal consisting of five distinct positions, classification methods were considered. The performance of a Data Mining method relies heavily on whether the method is appropriate according to the underlying shape of the data. The more complex it is, the more flexible the method should be. If there are at most two (three??) predictors, the shape of the data can be estimated by plotting the data. In the accelerometer data set, there are 18 predictors which were cut down to 12-13 (depending on analysis method) relevant features. Therefore, a graphical analysis was not possible, and it was unclear whether the data has a linear or a more complex shape. Instead, a different approach was implemented which is presented below. We applied different techniques including linear methods (LDA, linear SVM) as well as more flexible ones (QDA, polynomial and radial SVM, KNN). The outcomes were then compared among them using performance measures. A brief description of the application of each method is given below.

Linear/Quadratic Discriminant Analysis (LDA/QDA)

One of the more concerning facts is that LDA only takes into account the numeric predictors. Hence, even if nominal variables, like gender, have some role in the classification process they were not taken in to account. In addition, only age and BMI were included as predictors apart from the accelerometer data.

LDA as well as QDA assume that the predictor features have a normal distribution. In case of violation of this assumption, the methods will definitely perform poorly on the given data set. Our results and the distribution graphs are consistent with this theory. Some of the features are not normally distributed (see Appendix 1 for examples) and this is also reflected in the results obtained as there is a huge number of misclassifications. Therefore, both methods were discarded for evaluation of the data set and are not handled any further in this report. In addition, the poor fit may also be attributed to a nonlinear shape of the decision boundaries which separates the different classes.

Support Vector Machines (SVM)

The predictors were changed compared to LDA/QDA analysis. Nominal features could be included and, hence, all features except height and weight (which are highly correlated with BMI) were included. Support Vector Machines can be used with different choice of kernels depending on the linearity/non-linearity of the decision (or classification) boundaries. Hence, we used different kernels like linear, polynomial and radial and assessed their performance on the dataset. The different parameters were tuned using 10-fold cross validation to select best parameter in each case i.e. cost for linear kernel, cost & degree for polynomial kernel and cost & $\boldsymbol{\gamma}$ for radial kernel. The linear kernel is expected to perform well if the classes are separable by a hyperplane, but given the results from LDA analysis, it would probably perform poorly since there is an indication of a non-linear decision boundary. Since SVM requires huge computational effort, the best kernel was chosen by training and testing the SVM models on relatively small subset. It was found that the linear and polynomial kernel underperformed in terms

of accuracy (see Table 1 below) compared to the radial kernel. Due to their underperformance, linear and polynomial kernels were dropped as a method for classification analysis.
To be further sure about radial kernel's performance it was trained and tested on a subset on 5000 instances and it was observed that the performance of the kernel improved drastically. Hence, it was decided to train and test the SVM model using radial kernel on the entire dataset.
The parameters used were cost=10 and **γ**=0.5.

Sample\Kernel	Linear	Polynomial	Radial
1000	83.79	91.7	92.89
5000	84.88	96.88	96.96
All Training Data	-	-	99.38

Table 1: Performance measure - Accuracy of different kernels

k-Nearest Neighbours (KNN)

In case of KNN, the smallest number of predictors was used. That is, the accelerometer data only as the method was not performing well with more predictors included (backward selection). In this particular case, since all the predictors had the same unit, there was no need for standardization or scaling of predictors.
KNN is a non-parametric classification method which works best in case of highly non-linear boundaries or boundaries which do not particularly assume some formal shapes or patterns. The choice of K has drastic effect on the classifier obtained. As K grows, the method becomes less flexible and produce a decision boundary that is closer to linear. From the previous analysis of SVM, it is apparent that the data is non-linear and hence smaller value of K would probably perform better. The value of K was chosen on the accuracy performance measure. For K=1, the classifier performs the best (as reflected in Table 2 below).

Value of K	Accuracy
1	99.47
3	99.43
5	99.34

Table 2: Performance measure - Accuracy of different K values

Performance measures

In order to rank the performance of the different data mining techniques applied on the given data set we used the following measures:

1. Accuracy - Proportion of total number of correct classifications to total number of test instances
2. Precision - Proportion of the true positive to the total number of classifications in a particular class
3. Sensitivity - Proportion of positives/true classification that are correctly identified

Comparison of radial SVM and KNN

We decided to concentrate on the two most promising classifiers according to our experimental analysis: radial SVM (c=10, γ=0.5) and KNN (K=1). They were trained on the training data as mentioned in the section on data splits. The comparison between the two methods is based on the previously described performance measures and the computation complexity which was measured in execution time.

	KNN	Radial SVM
Sitting	99.98	99.85
Standing	99.87	99.67
Sitting down	99.00	98.55
Standing up	97.62	97.01
Walking	99.10	99.43

Table 3: Comparison of Sensitivity for all classes

	KNN	Radial SVM
Sitting	99.96	99.97
Standing	99.35	99.40
Sitting down	97.53	98.76
Standing up	98.55	98.80
Walking	99.94	99.01

Table 4: Comparison of Precision for all classes

As it can be seen in Tables 1 and 2, KNN leads to slightly higher accuracy than radial SVM. When it comes to sensitivity and precision, both methods perform equally well which is displayed in Tables 3 and 4. But one method can be chosen one over the other when computational effort is used as a performance evaluation criterion: While KNN only needed 55 seconds for the computations, radial SVM had an execution time of 10 minutes and 2 seconds.

Conclusion

The goal of this project was to predict the human posture. We were able to come up with two data mining algorithms which performed very well with over 99% accuracy. Interestingly, KNN performed exceptionally well with k=1, whereas in most data sets k=1 would tend to overfit the data. This implies a highly non-linear decision boundary.

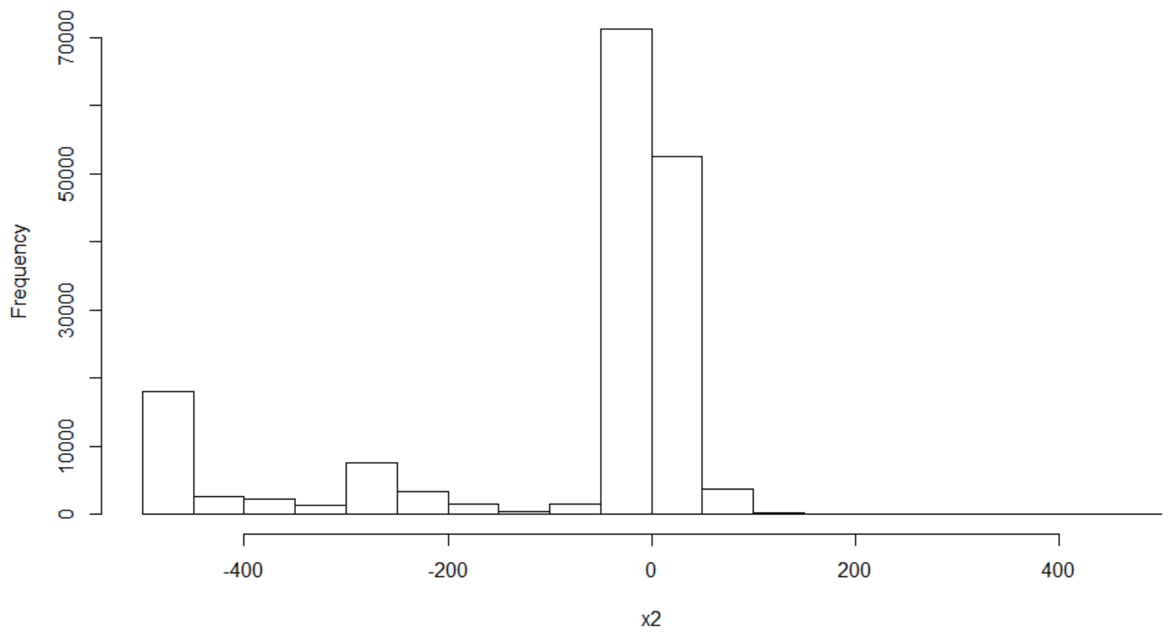
Yet, there is one weakness in the fact that SVM in general takes a long time to run. This is because the running time scales as square of the number of training instances.

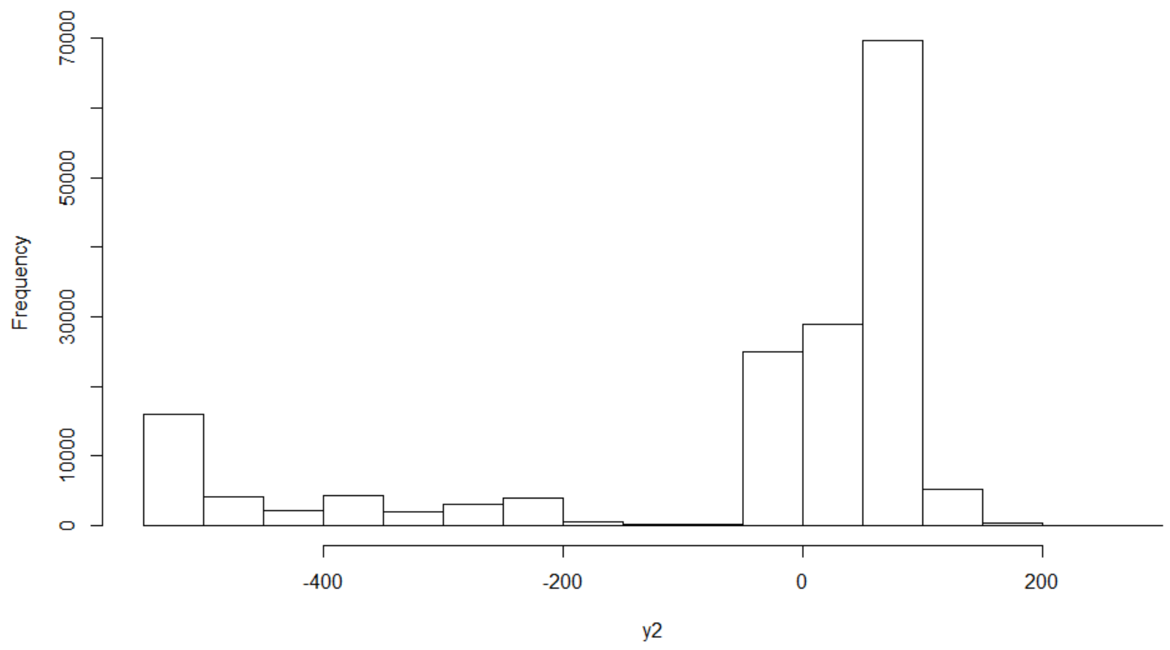
We were able to show that there exist data mining techniques which can predict human posture. These can be applied to practical applications in Human Activity Recognition.

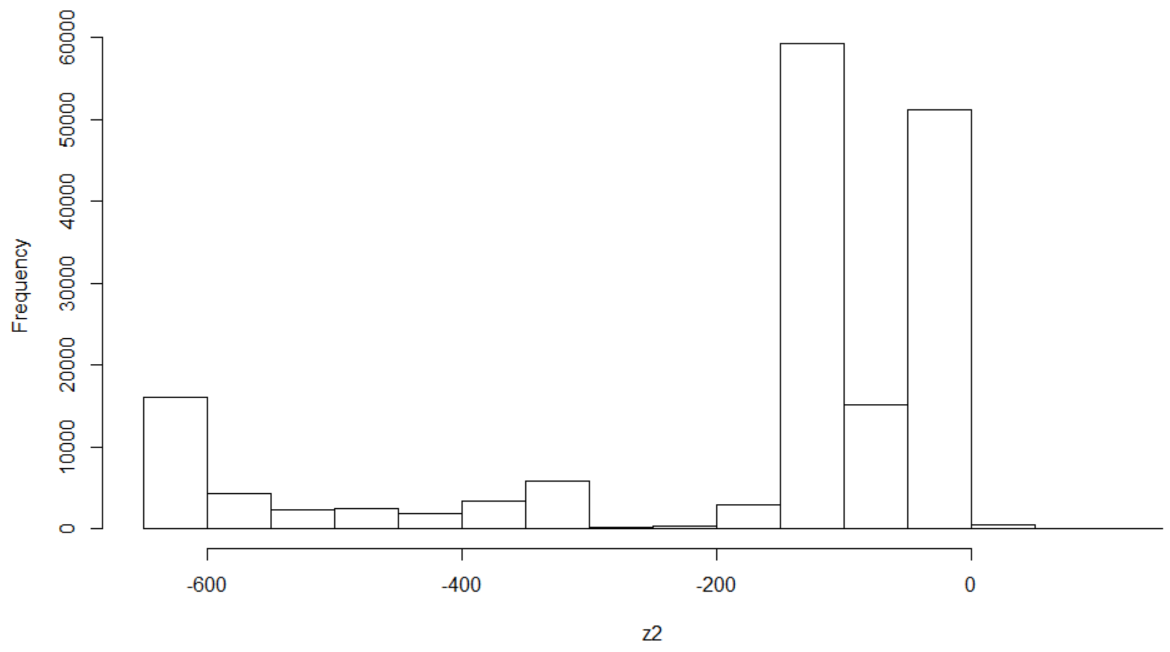
As there are many more machine learning algorithms, future works might involve tree-based methods and neural networks.

Appendix 1

Histogram of features x2, y2, z2







Appendix 2

R code and output

Accelerometer Prediction

```r
setwd("~/Desktop/MA429 Mock Project/")
#Load Relevant Libraries
library(e1071)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(class)
#Read in Data
accelerometer_data <- read.table("accelerometer.csv", sep = ";", header = TRUE, dec = ",")
head(accelerometer_data)
```

```
##     user gender age how_tall_in_meters weight body_mass_index x1 y1  z1
## 1 debora  Woman  46               1.62     75            28.6 -3 92 -63
## 2 debora  Woman  46               1.62     75            28.6 -3 94 -64
## 3 debora  Woman  46               1.62     75            28.6 -1 97 -61
## 4 debora  Woman  46               1.62     75            28.6 -2 96 -57
## 5 debora  Woman  46               1.62     75            28.6 -1 96 -61
## 6 debora  Woman  46               1.62     75            28.6 -2 95 -62
##    x2 y2  z2  x3  y3  z3   x4   y4   z4   class
## 1 -23 18 -19   5 104 -92 -150 -103 -147 sitting
## 2 -21 18 -18 -14 104 -90 -149 -104 -145 sitting
## 3 -12 20 -15 -13 104 -90 -151 -104 -144 sitting
## 4 -15 21 -16 -13 104 -89 -153 -103 -142 sitting
## 5 -13 20 -15 -13 104 -89 -153 -104 -143 sitting
## 6 -14 19 -16 -13 104 -89 -153 -104 -142 sitting
```

```r
summary(accelerometer_data)
```

```
##           user          gender            age        how_tall_in_meters
##  debora     :51577   Man  : 64259   Min.   :28.00   Min.   :1.58
##  jose_carlos:13161   Woman:101374   1st Qu.:28.00   1st Qu.:1.58
##  katia      :49797                  Median :31.00   Median :1.62
##  wallace    :51098                  Mean   :38.27   Mean   :1.64
##                                     3rd Qu.:46.00   3rd Qu.:1.71
##                                     Max.   :75.00   Max.   :1.71
##
##      weight      body_mass_index       x1                 y1
##  Min.   :55.00   Min.   :22.00   Min.   :-306.000   Min.   :-271.00
##  1st Qu.:55.00   1st Qu.:22.00   1st Qu.: -12.000   1st Qu.:  78.00
##  Median :75.00   Median :28.40   Median :  -6.000   Median :  94.00
##  Mean   :70.82   Mean   :26.19   Mean   :  -6.649   Mean   :  88.29
##  3rd Qu.:83.00   3rd Qu.:28.60   3rd Qu.:   0.000   3rd Qu.: 101.00
##  Max.   :83.00   Max.   :28.60   Max.   : 509.000   Max.   : 533.00
##
##        z1                 x2                 y2                 z2
##  Min.   :-603.00   Min.   :-494.00   Min.   :-517.00   Min.   :-617.0
##  1st Qu.:-120.00   1st Qu.: -35.00   1st Qu.: -29.00   1st Qu.:-141.0
##  Median : -98.00   Median :  -9.00   Median :  27.00   Median :-118.0
##  Mean   : -93.16   Mean   : -87.83   Mean   : -52.06   Mean   :-175.1
##  3rd Qu.: -64.00   3rd Qu.:   4.00   3rd Qu.:  86.00   3rd Qu.: -29.0
##  Max.   : 411.00   Max.   : 473.00   Max.   : 295.00   Max.   : 122.0
##
##        x3                y3               z3                 x4
##  Min.   :-499.00   Min.   :-506.0   Min.   :-613.00   Min.   :-702.0
##  1st Qu.:   9.00   1st Qu.:  95.0   1st Qu.:-103.00   1st Qu.:-190.0
##  Median :  22.00   Median : 107.0   Median : -90.00   Median :-168.0
##  Mean   :  17.42   Mean   : 104.5   Mean   : -93.88   Mean   :-167.6
##  3rd Qu.:  34.00   3rd Qu.: 120.0   3rd Qu.: -80.00   3rd Qu.:-153.0
##  Max.   : 507.00   Max.   : 517.0   Max.   : 410.00   Max.   : -13.0
##
##        y4                z4                class
##  Min.   :-526.00   -162   :  6859   sitting    :50631
##  1st Qu.:-103.00   -158   :  6770   sittingdown:11827
##  Median : -91.00   -163   :  6762   standing   :47370
##  Mean   : -92.63   -159   :  6641   standingup :12415
##  3rd Qu.: -80.00   -161   :  6402   walking    :43390
##  Max.   :  86.00   -160   :  6114
##                    (Other):126085
```

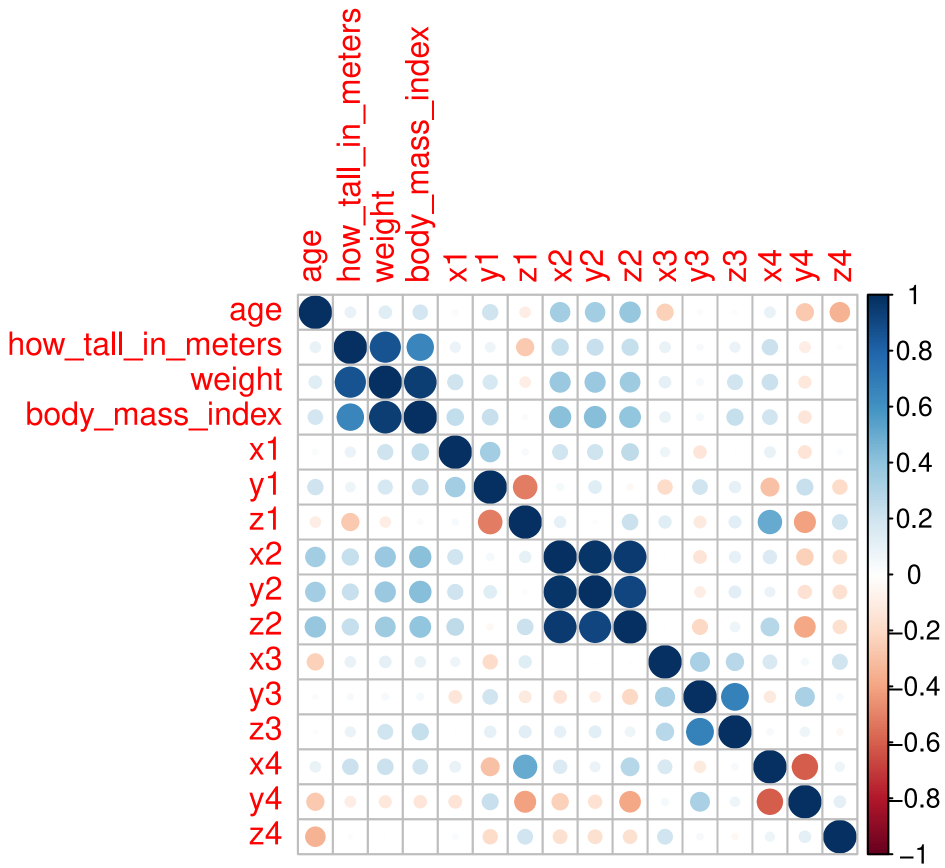
```
dim(accelerometer_data)
```

```
## [1] 165633     19
```

```
#Check for any Missing Values
anyNA(accelerometer_data)
```

```
## [1] FALSE
```

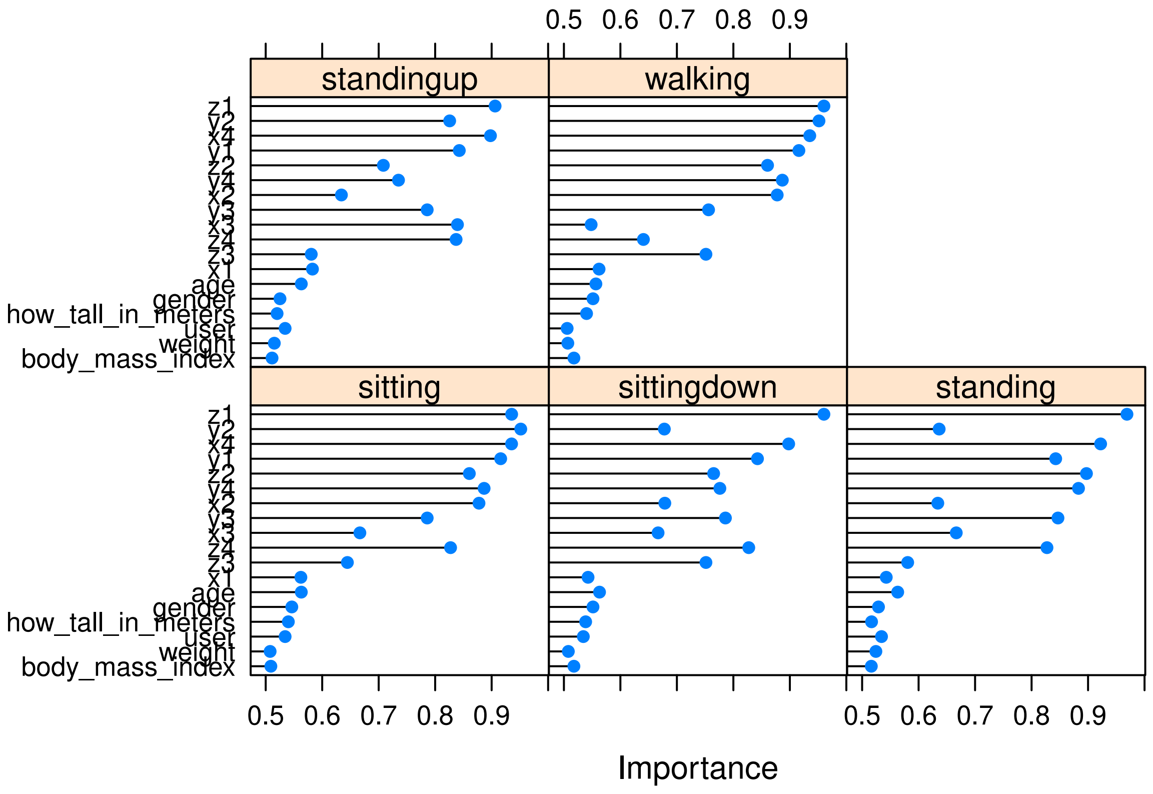
```
# Convert all numerical variables to class "numeric" to enable correlation computation
accelerometer_data[,3:18] <- sapply(accelerometer_data[,3:18],as.numeric)

#Create a subset to work on before trying on full dataset
set.seed(201316007)
subset1 <- sample(165633, 1000)
accelerometer_subset <- accelerometer_data[subset1,]

# Feature Selection
correlations <- cor(accelerometer_data[,3:18])
corrplot(correlations, method = "circle")
```



```
#Rank features by importance
#ensure results are repeatable
set.seed(201316007)
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model <- train(class~., data=accelerometer_subset, method="lvq", preProcess="scale", trControl=control)
# estimate variable importance
importance <- varImp(model, scale=FALSE)
# summarize importance
print(importance)
```

```
## ROC curve variable importance
##
##   variables are sorted by maximum importance across the classes
##                        sitting sittingdown standing standingup walking
## z1                      0.9354      0.9603   0.9689     0.9061  0.9603
## y2                      0.9516      0.6778   0.6364     0.8257  0.9516
## x4                      0.9353      0.8979   0.9223     0.8979  0.9353
## y1                      0.9160      0.8427   0.8427     0.8427  0.9160
## z2                      0.8605      0.7650   0.8972     0.7083  0.8605
## y4                      0.8867      0.7761   0.8831     0.7351  0.8867
## x2                      0.8776      0.6787   0.6340     0.6340  0.8776
## y3                      0.7859      0.7859   0.8467     0.7859  0.7561
## x3                      0.6667      0.6667   0.6667     0.8395  0.5482
## z4                      0.8273      0.8273   0.8273     0.8371  0.6408
## z3                      0.6446      0.7514   0.5807     0.5807  0.7514
## x1                   0.5623    0.5427  0.5427    0.5828  0.5623
## age                  0.5630    0.5630  0.5630    0.5630  0.5566
## gender               0.5462    0.5515  0.5290    0.5253  0.5515
## how_tall_in_meters   0.5401    0.5385  0.5167    0.5202  0.5401
## user                 0.5344    0.5344  0.5344    0.5344  0.5058
## weight               0.5078    0.5078  0.5244    0.5152  0.5070
## body_mass_index      0.5092    0.5177  0.5163    0.5112  0.5177
```

```
# plot importance
plot(importance)
```



```
#Create a training and testing set with a 0.75:0.25 ratio by random sampling
set.seed(201316007)
split <- sample(seq_len(nrow(accelerometer_subset)), size = floor(0.75*nrow(accelerometer_subset)))
#train_set<- accelerometer_subset[split,]

#####Creating training and testing set from subset, having features "weight" and "BMI" removed
train_set<- accelerometer_subset[split,-c(4,5)]
#test_set <- accelerometer_subset[-split,]
test_set <- accelerometer_subset[-split,-c(4,5)]
dim(train_set)
```

```
## [1] 750  17
```

```
dim(test_set)
```

```
## [1] 250  17
```

```
############Method 1) : Support Vector Machines ##################
#Perform 10 fold CV on the model, varying the cost, to determine which cost is best for a linear kernel
set.seed(201316007)
```

```r
tune.out = tune(svm, class~., data = train_set,kernel = "linear", ranges =
                  list(cost = c(0.0001, 0.01, 0.1,1,5,10,20)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.1586667
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-04 0.7253333 0.04836078
## 2 1e-02 0.2813333 0.06423721
## 3 1e-01 0.2186667 0.06416029
## 4 1e+00 0.1706667 0.04523737
## 5 5e+00 0.1640000 0.04402020
## 6 1e+01 0.1586667 0.04190524
## 7 2e+01 0.1626667 0.04063690
```

```r
bestmod = tune.out$best.model
#bestmod says that a cost of 5 gives the lowest error.

classpred <- predict(bestmod, test_set[,-17])
confusionMatrix(table(predict = classpred, truth = test_set$class))
```

```
## Confusion Matrix and Statistics
##
##                truth
## predict       sitting sittingdown standing standingup walking
##   sitting          79           2        0          0       1
##   sittingdown       0          12        0          1       0
##   standing          0           4       65          1       9
##   standingup        0           1        0         12       3
##   walking           0           3        2          0      55
##
## Overall Statistics
##
##                Accuracy : 0.892
##                  95% CI : (0.8468, 0.9276)
##     No Information Rate : 0.316
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8537
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                0.0000          0.5455          0.9701
## Specificity                0.9825          0.9956          0.9235
## Pos Pred Value             0.9634          0.9231          0.8228
## Neg Pred Value             1.0000          0.9578          0.9883
## Prevalence                 0.3160          0.0880          0.2680
## Detection Rate             0.3160          0.0480          0.2600
## Detection Prevalence       0.3280          0.0520          0.3160
## Balanced Accuracy          0.9912          0.7705          0.9468
##                      Class: standingup Class: walking
## Sensitivity                   0.8571         0.8088
## Specificity                   0.9831         0.9725
## Pos Pred Value                0.7500         0.9167
## Neg Pred Value                0.9915         0.9316
## Prevalence                    0.0560         0.2720
## Detection Rate                0.0480         0.2200
## Detection Prevalence          0.0640         0.2400
## Balanced Accuracy             0.9201         0.8907
```

```
# Trying with a Polynomial kernel, tuning cost and degree
set.seed(201316007)
tune.out = tune(svm, class~., data = train_set,kernel = "polynomial", ranges =
                  list(cost = c(0.0001, 0.01, 0.1,1,5,10,20),degree = c(1,2,3,4,5)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    20      2
##
## - best performance: 0.08933333
##
## - Detailed performance results:
##     cost degree      error dispersion
## 1  1e-04      1 0.72533333 0.04836078
## 2  1e-02      1 0.54533333 0.10525923
## 3  1e-01      1 0.31200000 0.06477311
## 4  1e+00      1 0.22933333 0.06734470
## 5  5e+00      1 0.19066667 0.05588348
## 6  1e+01      1 0.17333333 0.04868645
## 7  2e+01      1 0.16933333 0.04705657
## 8  1e-04      2 0.72533333 0.04836078
## 9  1e-02      2 0.71466667 0.05190685
## 10 1e-01      2 0.37733333 0.07595970
## 11 1e+00      2 0.19466667 0.05936287
## 12 5e+00      2 0.11600000 0.03208015
## 13 1e+01      2 0.10266667 0.03445520
## 14 2e+01      2 0.08933333 0.03877125
## 15 1e-04      3 0.72133333 0.05123656
## 16 1e-02      3 0.69333333 0.05258738
## 17 1e-01      3 0.39600000 0.05864309
## 18 1e+00      3 0.23466667 0.04584461
## 19 5e+00      3 0.16533333 0.03621609
## 20 1e+01      3 0.14133333 0.02529822
## 21 2e+01      3 0.10800000 0.03742317
## 22 1e-04      4 0.72000000 0.05106278
## 23 1e-02      4 0.66533333 0.05810166
## 24 1e-01      4 0.40533333 0.07004760
## 25 1e+00      4 0.31600000 0.06223809
## 26 5e+00      4 0.19866667 0.04508428
## 27 1e+01      4 0.17200000 0.04375013
## 28 2e+01      4 0.15466667 0.03833575
## 29 1e-04      5 0.72133333 0.05123656
## 30 1e-02      5 0.63333333 0.09701724
## 31 1e-01      5 0.42666667 0.06769576
## 32 1e+00      5 0.34266667 0.06349492
## 33 5e+00      5 0.26533333 0.05084957
## 34 1e+01      5 0.24800000 0.05190685
## 35 2e+01      5 0.18266667 0.04075825
```

```
bestmod = tune.out$best.model
#bestmod says that a cost of 20 and degree of 1 gives the lowest error.

classpred <- predict(bestmod, test_set[,-17])
confusionMatrix(table(predict = classpred, truth = test_set$class))
```

```
## Confusion Matrix and Statistics
##
##              truth
## predict       sitting sittingdown standing standingup walking
##   sitting          78           1        2          0       1
##   sittingdown       0          15        0          0       0
##   standing          0           3       65          0      12
##   standingup        0           1        0         12       1
##   walking           1           2        0          2      54
##
## Overall Statistics
##
##                Accuracy : 0.896
##                  95% CI : (0.8513, 0.9309)
##     No Information Rate : 0.316
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8591
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                  0.9873             0.6818          0.9701
## Specificity                  0.9766             1.0000          0.9180
## Pos Pred Value               0.9512             1.0000          0.8125
## Neg Pred Value               0.9940             0.9702          0.9882
## Prevalence                   0.3160             0.0880          0.2680
## Detection Rate               0.3120             0.0600          0.2600
## Detection Prevalence         0.3280             0.0600          0.3200
## Balanced Accuracy            0.9820             0.8409          0.9441
##                      Class: standingup Class: walking
## Sensitivity                     0.8571         0.7941
## Specificity                     0.9915         0.9725
## Pos Pred Value                  0.8571         0.9153
## Neg Pred Value                  0.9915         0.9267
## Prevalence                      0.0560         0.2720
## Detection Rate                  0.0480         0.2160
## Detection Prevalence            0.0560         0.2360
## Balanced Accuracy               0.9243         0.8833
```

```
#Changing kernel to be radial, tuning cost and gamma
set.seed(201316007)
tune.out = tune(svm, class~., data = train_set,kernel = "radial", ranges =
                  list(cost = c(0.0001, 0.01, 0.1,1,5,10,20), gamma = c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     5   0.5
##
## - best performance: 0.08133333
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1  1e-04   0.5 0.72533333 0.04836078
## 2  1e-02   0.5 0.72533333 0.04836078
## 3  1e-01   0.5 0.23466667 0.06891039
## 4  1e+00   0.5 0.08933333 0.03613418
## 5  5e+00   0.5 0.08133333 0.02978109
## 6  1e+01   0.5 0.08133333 0.03232550
## 7  2e+01   0.5 0.08266667 0.03192584
## 8  1e-04   1.0 0.72533333 0.04836078
## 9  1e-02   1.0 0.72533333 0.04836078
## 10 1e-01   1.0 0.25600000 0.07378313
## 11 1e+00   1.0 0.12800000 0.04131182
## 12 5e+00   1.0 0.11466667 0.03510830
## 13 1e+01   1.0 0.11600000 0.03502380
## 14 2e+01   1.0 0.11866667 0.03468376
## 15 1e-04   2.0 0.72533333 0.04836078
## 16 1e-02   2.0 0.72533333 0.04836078
## 17 1e-01   2.0 0.32533333 0.08507368
## 18 1e+00   2.0 0.16400000 0.05588348
## 19 5e+00   2.0 0.15733333 0.04819713
## 20 1e+01   2.0 0.15866667 0.05200190
## 21 2e+01   2.0 0.15866667 0.05200190
## 22 1e-04   3.0 0.72533333 0.04836078
## 23 1e-02   3.0 0.72533333 0.04836078
## 24 1e-01   3.0 0.42266667 0.05260615
## 25 1e+00   3.0 0.19066667 0.05931294
## 26 5e+00   3.0 0.18133333 0.05869360
## 27 1e+01   3.0 0.18133333 0.05869360
## 28 2e+01   3.0 0.18133333 0.05869360
## 29 1e-04   4.0 0.72533333 0.04836078
## 30 1e-02   4.0 0.72533333 0.04836078
## 31 1e-01   4.0 0.42400000 0.05213468
## 32 1e+00   4.0 0.20800000 0.05414202
## 33 5e+00   4.0 0.20133333 0.05531503
## 34 1e+01   4.0 0.20133333 0.05531503
## 35 2e+01   4.0 0.20133333 0.05531503
```

```
bestmod = tune.out$best.model
#bestmod says that a cost of 5 and gamma of 0.5 gives the lowest error.

classpred <- predict(bestmod, test_set[,-17])
confusionMatrix(table(predict = classpred, truth = test_set$class))
```

```
## Confusion Matrix and Statistics
##
##                truth
## predict         sitting sittingdown standing standingup walking
##   sitting            77           0        0          0       0
##   sittingdown         0          21        0          0       0
##   standing            0           0       66          0       4
##   standingup          0           0        0          9       0
##   walking             2           1        1          5      64
##
## Overall Statistics
##
##                  Accuracy : 0.948
##                    95% CI : (0.9127, 0.972)
##       No Information Rate : 0.316
##       P-Value [Acc > NIR] : < 2.2e-16
##
##                     Kappa : 0.9296
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                  0.9747             0.9545          0.9851
## Specificity                  1.0000             1.0000          0.9781
## Pos Pred Value               1.0000             1.0000          0.9429
## Neg Pred Value               0.9884             0.9956          0.9944
## Prevalence                   0.3160             0.0880          0.2680
## Detection Rate               0.3080             0.0840          0.2640
## Detection Prevalence         0.3080             0.0840          0.2800
## Balanced Accuracy            0.9873             0.9773          0.9816
##                      Class: standingup Class: walking
## Sensitivity                     0.6429         0.9412
## Specificity                     1.0000         0.9505
## Pos Pred Value                  1.0000         0.8767
## Neg Pred Value                  0.9793         0.9774
## Prevalence                      0.0560         0.2720
## Detection Rate                  0.0360         0.2560
## Detection Prevalence            0.0360         0.2920
## Balanced Accuracy               0.8214        0.9459
```

```
#Now Use Radial Kernel with cost = 10, gamma = 0.5 for entire dataset:
set.seed(201316007)
split <- sample(seq_len(nrow(accelerometer_data)), size = floor(0.75*nrow(accelerometer_data)))
# Remove 2 least important features:
train_set_full<- accelerometer_data[split,-c(4,5)]
test_set_full <- accelerometer_data[-split,-c(4,5)]

dim(train_set_full)
```

```
## [1] 124224     17
```

```
dim(test_set_full)
```

```
## [1] 41409    17
```

```
set.seed(201316007)
start.time <- Sys.time()
svm_fit = svm(class~., data = train_set_full,kernel = "radial", cost = 10, gamma = 0.5)
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 10.46888 mins
```

```
summary(svm_fit)
```

```
##
## Call:
## svm(formula = class ~ ., data = train_set_full, kernel = "radial",
##     cost = 10, gamma = 0.5)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
##
## Number of Support Vectors:  9782
##
##  ( 5063 1040 413 2112 1154 )
##
##
## Number of Classes:  5
##
## Levels:
##  sitting sittingdown standing standingup walking
```

```
classpred <- predict(svm_fit, test_set_full[,-17])
confusionMatrix(table(predict = classpred, truth = test_set_full$class))
```

```
## Confusion Matrix and Statistics
##
##              truth
## predict       sitting sittingdown standing standingup walking
##   sitting       12636           1        0          3       0
##   sittingdown       1        2868        0         27       8
##   standing          0           3     11876         21       48
##   standingup        2          16        13       3048        6
##   walking          16          22        26         43    10725
##
## Overall Statistics
##
##                Accuracy : 0.9938
##                  95% CI : (0.993, 0.9946)
##     No Information Rate : 0.3056
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9917
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                  0.9985            0.98557          0.9967
## Specificity                  0.9999            0.99906          0.9976
## Pos Pred Value               0.9997            0.98760          0.9940
## Neg Pred Value               0.9993            0.99891          0.9987
## Prevalence                   0.3056            0.07027          0.2877
## Detection Rate               0.3052            0.06926          0.2868
## Detection Prevalence         0.3052            0.07013          0.2885
## Balanced Accuracy            0.9992            0.99232          0.9971
##                      Class: standingup Class: walking
## Sensitivity                    0.97008         0.9943
## Specificity                    0.99903         0.9965
## Pos Pred Value                 0.98801         0.9901
## Neg Pred Value                 0.99755         0.9980
## Prevalence                     0.07588         0.2605
## Detection Rate                 0.07361         0.2590
## Detection Prevalence           0.07450         0.2616
## Balanced Accuracy              0.98456         0.9954
```

```
# [1] 0.9938178%


##################Method 2) KNN ##########################:
# K = 1
set.seed(201316007)
split <- sample(seq_len(nrow(accelerometer_data)), size = floor(0.9*nrow(accelerometer_data)))
# Remove 6 least important features:
train_set_full<- accelerometer_data[split,-c(1:6)]
test_set_full <- accelerometer_data[-split,-c(1:6)]

standardized.train.X = train_set_full[,-c(13)]
standardized.test.X = test_set_full[,-c(13)]
train.Y = train_set_full[,13]
test.Y = test_set_full[,13]

set.seed(201316007)
start.time <- Sys.time()
knn.pred = knn(standardized.train.X, standardized.test.X,train.Y,k=1)
```

```r
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 50.83761 secs
```

```r
confusionMatrix(table(knn.pred, truth = test.Y))
```

```
## Confusion Matrix and Statistics
##
##              truth
## knn.pred      sitting sittingdown standing standingup walking
##   sitting        5048           0        0          2       0
##   sittingdown       0        1188        0         20      10
##   standing          0           1     4736          7      23
##   standingup        1           9        2       1231       6
##   walking           0           2        4          1    4273
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9935, 0.9957)
##     No Information Rate : 0.3048
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                  0.9998            0.99000          0.9987
## Specificity                  0.9998            0.99805          0.9974
## Pos Pred Value               0.9996            0.97537          0.9935
## Neg Pred Value               0.9999            0.99922          0.9995
## Prevalence                   0.3048            0.07245          0.2863
## Detection Rate               0.3048            0.07172          0.2859
## Detection Prevalence         0.3049            0.07353          0.2878
## Balanced Accuracy            0.9998            0.99402          0.9981
##                      Class: standingup Class: walking
## Sensitivity                    0.97621         0.9910
## Specificity                    0.99882         0.9994
## Pos Pred Value                 0.98559         0.9984
## Neg Pred Value                 0.99804         0.9968
## Prevalence                     0.07613         0.2603
## Detection Rate                 0.07432         0.2580
## Detection Prevalence           0.07540         0.2584
## Balanced Accuracy              0.98752         0.9952
```

```r
# 0.9%

#Trying with k = 3
set.seed(201316007)
split <- sample(seq_len(nrow(accelerometer_data)), size = floor(0.9*nrow(accelerometer_data)))
# Remove 6 least important features:
train_set_full<- accelerometer_data[split,-c(1:6)]
```

```r
test_set_full <- accelerometer_data[-split,-c(1:6)]

standardized.train.X = train_set_full[,-c(13)]
standardized.test.X = test_set_full[,-c(13)]
train.Y = train_set_full[,13]
test.Y = test_set_full[,13]

set.seed(201316007)
start.time <- Sys.time()
knn.pred = knn(standardized.train.X, standardized.test.X,train.Y,k=3)
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 1.207006 mins
```

```r
confusionMatrix(table(knn.pred, truth = test.Y))
```

```
## Confusion Matrix and Statistics
##
##              truth
## knn.pred      sitting sittingdown standing standingup walking
##   sitting        5048           0        0          4       0
##   sittingdown       0        1186        0         19       9
##   standing          0           2     4739         10      27
##   standingup        1          10        1       1225       4
##   walking           0           2        2          3    4272
##
## Overall Statistics
##
##                Accuracy : 0.9943
##                  95% CI : (0.9931, 0.9954)
##     No Information Rate : 0.3048
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9924
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                  0.9998            0.98833          0.9994
## Specificity                  0.9997            0.99818          0.9967
## Pos Pred Value               0.9992            0.97694          0.9918
## Neg Pred Value               0.9999            0.99909          0.9997
## Prevalence                   0.3048            0.07245          0.2863
## Detection Rate               0.3048            0.07160          0.2861
## Detection Prevalence         0.3050            0.07329          0.2885
## Balanced Accuracy            0.9997            0.99326          0.9980
##                      Class: standingup Class: walking
## Sensitivity                    0.97145         0.9907
## Specificity                    0.99895         0.9994
## Pos Pred Value                 0.98711         0.9984
## Neg Pred Value                 0.99765         0.9967
## Prevalence                     0.07613         0.2603
## Detection Rate                 0.07396         0.2579
## Detection Prevalence           0.07492         0.2583
## Balanced Accuracy              0.98520         0.9951
```

```
# K =6
set.seed(201316007)
split <- sample(seq_len(nrow(accelerometer_data)), size = floor(0.9*nrow(accelerometer_data)))
# Remove 2 least important features:
train_set_full<- accelerometer_data[split,-c(1:6)]
test_set_full <- accelerometer_data[-split,-c(1:6)]

standardized.train.X = train_set_full[,-c(13)]
standardized.test.X = test_set_full[,-c(13)]
train.Y = train_set_full[,13]
test.Y = test_set_full[,13]

set.seed(201316007)
start.time <- Sys.time()
knn.pred = knn(standardized.train.X, standardized.test.X,train.Y,k=6)
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 58.71345 secs
```

```
confusionMatrix(table(knn.pred, truth = test.Y))
```

```
## Confusion Matrix and Statistics
##
##              truth
## knn.pred      sitting sittingdown standing standingup walking
##   sitting        5048           0        0          5       0
##   sittingdown       0        1186        0         21      10
##   standing          0           3     4734         12      29
##   standingup        1           8        2       1220       7
##   walking           0           3        6          3    4266
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.992, 0.9945)
##     No Information Rate : 0.3048
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9911
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                  0.9998            0.98833          0.9983
## Specificity                  0.9996            0.99798          0.9963
## Pos Pred Value               0.9990            0.97453          0.9908
## Neg Pred Value               0.9999            0.99909          0.9993
## Prevalence                   0.3048            0.07245          0.2863
## Detection Rate               0.3048            0.07160          0.2858
## Detection Prevalence         0.3051            0.07347          0.2885
## Balanced Accuracy            0.9997            0.99316          0.9973
##                      Class: standingup Class: walking
## Sensitivity                    0.96749         0.9893
## Specificity                    0.99882         0.9990
## Pos Pred Value                 0.98546         0.9972
## Neg Pred Value                 0.99732         0.9963
## Prevalence                     0.07613         0.2603
## Detection Rate                 0.07365         0.2575
## Detection Prevalence           0.07474         0.2583
## Balanced Accuracy              0.98315         0.9942
```

```
############ Method 3) LDA ###################
library(MASS)
set.seed(201316007)
split <- sample(seq_len(nrow(accelerometer_data)), size = floor(0.75*nrow(accelerometer_data)))
# Remove 2 least important features:
start.time <- Sys.time()
lda.fit <- lda(accelerometer_data$class ~ ., data = accelerometer_data[,-c(1:6)], subset =split )
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 1.022799 secs
```

```
lda.pred <- predict(lda.fit, accelerometer_data[-split,])
confusionMatrix(table(lda.pred$class, accelerometer_data[-split,]$class))
```

```
## Confusion Matrix and Statistics
##
##
##               sitting sittingdown standing standingup walking
##   sitting       12606         340        0        472      17
##   sittingdown      25        1469       55        462     138
##   standing          0         697    11113        731    3620
##   standingup       23         288       15       1323     257
##   walking           1         116      732        154    6755
##
## Overall Statistics
##
##                Accuracy : 0.8034
##                  95% CI : (0.7995, 0.8072)
##     No Information Rate : 0.3056
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7316
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                  0.9961            0.50481          0.9327
## Specificity                  0.9712            0.98234          0.8288
## Pos Pred Value               0.9383            0.68357          0.6876
## Neg Pred Value               0.9982            0.96330          0.9682
## Prevalence                   0.3056            0.07027          0.2877
## Detection Rate               0.3044            0.03548          0.2684
## Detection Prevalence         0.3244            0.05190          0.3903
## Balanced Accuracy            0.9836            0.74357          0.8808
##                      Class: standingup Class: walking
## Sensitivity                    0.42107         0.6262
## Specificity                    0.98476         0.9672
## Pos Pred Value                 0.69412         0.8707
## Neg Pred Value                 0.95395         0.8802
## Prevalence                     0.07588         0.2605
## Detection Rate                 0.03195         0.1631
## Detection Prevalence           0.04603         0.1874
## Balanced Accuracy              0.70292         0.7967
```

```
############## Method 4) QDA #################
library(MASS)
set.seed(201316007)
split <- sample(seq_len(nrow(accelerometer_data)), size = floor(0.75*nrow(accelerometer_data)))
# Remove 2 least important features:
start.time <- Sys.time()
qda.fit <- qda(accelerometer_data$class ~ ., data = accelerometer_data[,-c(1:6)], subset =split )
end.time <- Sys.time()
end.time - start.time
```

```
## Time difference of 0.847326 secs
```

```
qda.pred <- predict(qda.fit, accelerometer_data[-split,])
confusionMatrix(table(qda.pred$class, accelerometer_data[-split,]$class))
```

```
## Confusion Matrix and Statistics
##
##
##               sitting sittingdown standing standingup walking
##   sitting       12354         118        0         29       0
##   sittingdown     171        2376      212        786     472
##   standing          0         281    11530        397     727
##   standingup      130          78       55       1672      94
##   walking           0          57      118        258    9494
##
## Overall Statistics
##
##                Accuracy : 0.9038
##                  95% CI : (0.9009, 0.9066)
##     No Information Rate : 0.3056
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8709
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: sitting Class: sittingdown Class: standing
## Sensitivity                  0.9762            0.81649          0.9677
## Specificity                  0.9949            0.95738          0.9524
## Pos Pred Value               0.9882            0.59149          0.8914
## Neg Pred Value               0.9896            0.98572          0.9865
## Prevalence                   0.3056            0.07027          0.2877
## Detection Rate               0.2983            0.05738          0.2784
## Detection Prevalence         0.3019            0.09701          0.3124
## Balanced Accuracy            0.9856            0.88694          0.9600
##                      Class: standingup Class: walking
## Sensitivity                    0.53215         0.8801
## Specificity                    0.99067         0.9859
## Pos Pred Value                 0.82405         0.9564
## Neg Pred Value                 0.96267         0.9589
## Prevalence                     0.07588         0.2605
## Detection Rate                 0.04038         0.2293
## Detection Prevalence           0.04900         0.2397
## Balanced Accuracy              0.76141         0.9330
```