Project Title - PML_Project

Creator - James Cooksley

Date - JAN, 2018

Inroduction:

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways

Load in all the packages that are needed in the study:

```
library(dplyr)
```

```
## Warning: Installed Rcpp (0.12.9) different from Rcpp used to build dplyr
(0.12.11).
## Please reinstall dplyr to avoid random crashes or undefined behavior.
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(caret)

## Loading required package: lattice

library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine

library(rpart)
library(rpart.plot)
library(corrplot)
```

Training data

```
data.train<- read.csv("C://Users//u182335//Documents//DataScience//Course 7
Week 4//pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))
```

Testing data

```
data.test<- read.csv("C://Users//u182335//Documents//DataScience//Course 7
Week 4//pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))
```

Take a look at basic distributions of data:

```
dim(data.train)

## [1] 19622   160

str(data.train)

## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2
2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231
1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232
...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328
304277 368296 440390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9
9 9 9 9 9 9 9 9 9 ...
```

```
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1
1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42
1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13
8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -
94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02
0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -
0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21
...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609
...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313
-312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128
```

```
-128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8
21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161
-161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02
...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -
0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -
0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289
-288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110
...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124
-122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372
-369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334
...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516
...
##  $ kurtosis_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
```

```
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Data Transformation: Convert the date and additional variable (Day) for plots

```
data.train$cvtd_timestamp<- as.Date(data.train$cvtd_timestamp, format =
"%m/%d/%Y %H:%M")
data.train$Day<-factor(weekdays(data.train$cvtd_timestamp))
```

Exploratory data analysis for a better understanding of the data

```
table(data.train$classe)

##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607

prop.table(table(data.train$classe))

##
##         A         B         C         D         E
## 0.2843747 0.1935073 0.1743961 0.1638977 0.1838243

prop.table(table(data.train$user_name))

##
##    adelmo  carlitos   charles    eurico    jeremy     pedro
## 0.1983488 0.1585975 0.1802059 0.1564570 0.1733768 0.1330140

prop.table(table(data.train$user_name,data.train$classe),1)

##
##                   A         B         C         D         E
##   adelmo   0.2993320 0.1993834 0.1927030 0.1323227 0.1762590
##   carlitos 0.2679949 0.2217224 0.1584190 0.1561697 0.1956941
##   charles  0.2542421 0.2106900 0.1524321 0.1815611 0.2010747
##   eurico   0.2817590 0.1928339 0.1592834 0.1895765 0.1765472
```

```
##   jeremy   0.3459730 0.1437390 0.1916520 0.1534392 0.1651969
##   pedro    0.2452107 0.1934866 0.1911877 0.1796935 0.1904215
```

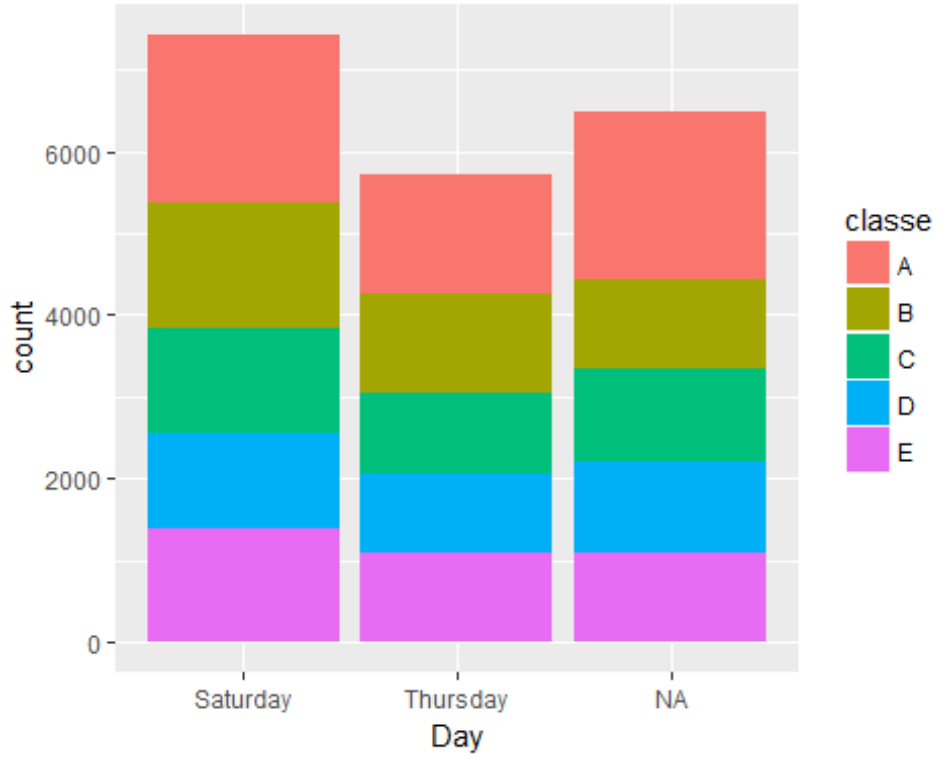
```
prop.table(table(data.train$user_name,data.train$classe),2)
```

```
##
##                    A         B         C         D         E
##   adelmo   0.2087814 0.2043719 0.2191701 0.1601368 0.1901857
##   carlitos 0.1494624 0.1817224 0.1440678 0.1511194 0.1688384
##   charles  0.1611111 0.1962075 0.1575102 0.1996269 0.1971167
##   eurico   0.1550179 0.1559126 0.1428989 0.1809701 0.1502634
##   jeremy   0.2109319 0.1287859 0.1905319 0.1623134 0.1558082
##   pedro    0.1146953 0.1329997 0.1458212 0.1458333 0.1377876
```

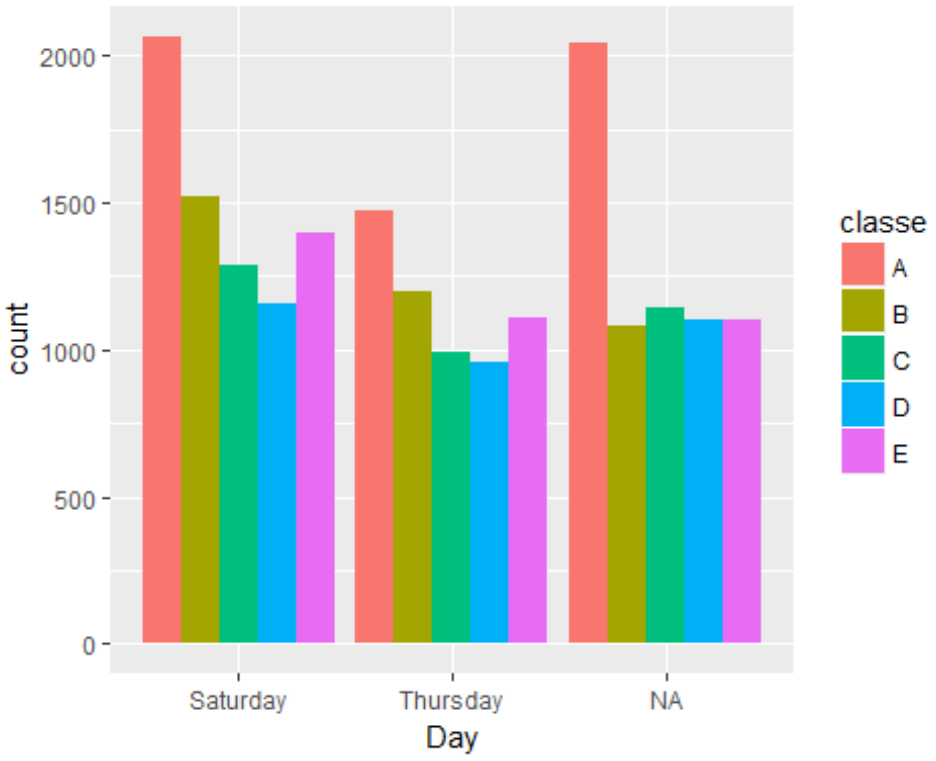
```
prop.table(table(data.train$classe, data.train$Day),1)
```

```
##
##      Saturday  Thursday
##   A 0.5833804 0.4166196
##   B 0.5600147 0.4399853
##   C 0.5651030 0.4348970
##   D 0.5478220 0.4521780
##   E 0.5581302 0.4418698
```

```
qplot(x=Day, fill=classe, data = data.train)
```




```
ggplot(data.train, aes(x=Day, fill=classe))+geom_bar(position="dodge")
```



Initial key findings from the exploratory data analysis: ** - A. The most frequently used activity is Class-A (28.4%) which is most frequently used by Jeremy - B. The most frequent user across acitivities is Adelmo (19.8%) and they are the most frequent user of Class-C - C. Most activities are held on a Saturday and Classes A and B are the most frequently used**

Data Cleansing: Remove columns with NA or missing values

```
data.train <- data.train[, colSums(is.na(data.train)) == 0]
data.test <- data.test[, colSums(is.na(data.test)) == 0]
```

Remove any of the columns that are no longer relevant to accelerometer measurements

```
classe<- data.train$classe
trainRemove<- grepl("^X|timestamp|window", names(data.train))
data.train<- data.train[, !trainRemove]
trainCleaned<- data.train[, sapply(data.train, is.numeric)]
trainCleaned$classe<- classe
testRemove<- grepl("^X|timestamp|window", names(data.test))
data.test<- data.test[, !testRemove]
testCleaned<- data.test[, sapply(data.test, is.numeric)]
```

Cleansed data contains 19622 observations and 53 variables in both the training and testing datasets Create Training and Testing data sets using a randomly assigned seed to get a random sample:

```
set.seed(68464)
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Data Modelling: Indetifying significant variables: By using Random Forest algorithm we will fit a predictive model to identify important variables and removes multicollinearity/outliers. 5-fold cross validation will be included when applying the algorithm

```
controlRf <- trainControl(method="cv", 5)
rfmod<- train(classe ~., data=trainData, method="rf", trControl=controlRf,
importance=TRUE, ntree=100)
rfmod
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10990, 10991, 10987
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9899547  0.9872923
##   27    0.9902455  0.9876597
##   52    0.9827474  0.9781727
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Understand the accuracy of the model on Validation data:

```
predictRfmod<- predict(rfmod, testData)
confusionMatrix(testData$classe, predictRfmod)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    1    1    0    1
##          B    4 1132    3    0    0
##          C    0    6 1016    4    0
##          D    0    1   18  943    2
##          E    0    2    0    4 1076
##
## Overall Statistics
##
```

```
##                Accuracy : 0.992
##                  95% CI : (0.9894, 0.9941)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9899
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9912   0.9788   0.9916   0.9972
## Specificity            0.9993   0.9985   0.9979   0.9957   0.9988
## Pos Pred Value         0.9982   0.9939   0.9903   0.9782   0.9945
## Neg Pred Value         0.9991   0.9979   0.9955   0.9984   0.9994
## Prevalence             0.2846   0.1941   0.1764   0.1616   0.1833
## Detection Rate         0.2839   0.1924   0.1726   0.1602   0.1828
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9984   0.9949   0.9884   0.9937   0.9980
```

```
accuracy <- postResample(predictRfmod, testData$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9920136 0.9898977
```

```
Error <- 1 - as.numeric(confusionMatrix(testData$classe,
predictRfmod)$overall[1])
Error
```

```
## [1] 0.007986406
```

Results: The estimated accuracy of the model is 99.2% and the estimated out-of-sample error is 0.8%

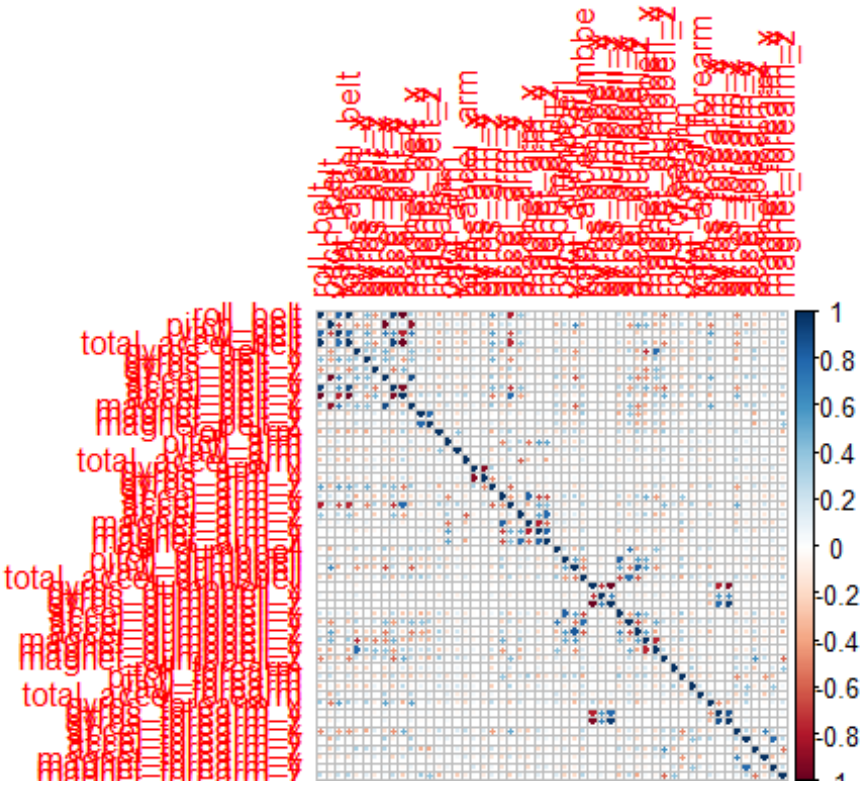
Predicting the test data

```
result <- predict(rfmod, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

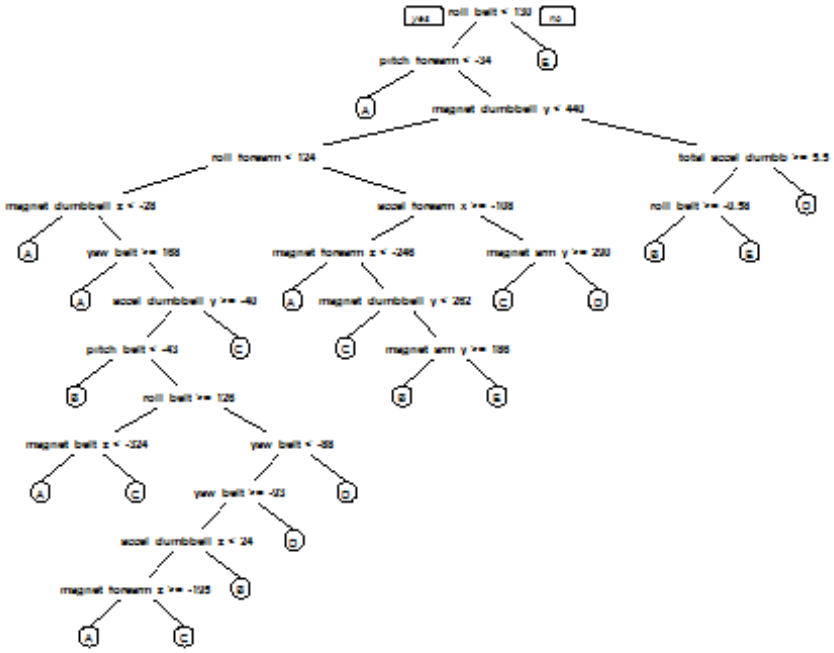
Appendix Create the correlation matrix

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="circle")
```



Plot the decision tree to help visualise the end result

```
rtree<- rpart(classe ~ ., data=trainData, method="class")
prp(rtree)
```


yes
roll belt < 130
no
pitch forearm < -34
E
A
magnet dumbbell y < 440
roll forearm < 124
total accel dumbb >= 5.5
magnet dumbbell z < -28
accel forearm x >= -108
roll belt >= -0.58
D
A
yaw belt >= 169
magnet forearm z < -246
magnet arm y >= 200
B
E
A
accel dumbbell y >= -40
A
magnet dumbbell y < 262
C
D
pitch belt < -43
C
C
magnet arm y >= 186
B
roll belt >= 126
B
E
magnet belt z < -224
yaw belt < -88
A
C
yaw belt >= -93
D
accel dumbbell z < 24
D
magnet forearm z >= -105
B
A
C