Practical Machine Learning Course Project

Oluwadare, Margaret

9/13/2020

EXECUTIVE SUMMARY

The goal of this data analysis is to predict the manner in which the user did the exercise. The training data set contains the target variable `classe`, of which other variables will be used for prediction. The best model that fits the data will be elicited using cross validation. Data preparation and cleaning will be done via removing columns that were not related to accelerometer reading and readings that were dominated by NA values reducing the variables from 160 to 53. The training data is partitioned into `Atrain` and `Atest` dataset which is used for training and validation respectively. The `cleantest` data set is used for the final prediction.

The decision tree model will be used first, followed by the Random forest modela and finally Generalised boosted regression model will be used. The best model will be finally used to predict the `cleantest` data set.

BACKGROUND

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, our 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

DATA SOURCES

The training data for this project is available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
The test data is available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
The data for this project comes from this original source: http://groupware.les.inf.puc-rio.br/har.

DATA LOADING AND PROCESSING

The following libraries are needed for the course of this project

```
library(rattle)
library(rsample)
library(caret)
library(rpart)
library(rpart.plot)
library(corrplot)
library(randomForest)
library(RColorBrewer)
```

```r
library(scales)
set.seed(3030)
```

DATA READING, CLEANING AND PARTITIONING

After downloading the data from the data source, we can read the two csv files into two data frames and check their structures. From the result below, our training set is a data frame of 19622 observations of 160 varaibles. The test data is a data frame of 20 observations of 160 varaibles. The first seven columns contains some informations that we might not need in the course of our analysis; also there are some features that have NA's and missing observations.

```r
train <- read.csv("./pml-training.csv", header = TRUE)

test <- read.csv("./pml-testing.csv", header = TRUE)

str(train, 5)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/20
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ 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      : chr  "" "" "" "" ...
##  $ kurtosis_picth_belt     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_belt       : chr  "" "" "" "" ...
##  $ skewness_roll_belt      : chr  "" "" "" "" ...
##  $ skewness_roll_belt.1    : chr  "" "" "" "" ...
##  $ skewness_yaw_belt       : chr  "" "" "" "" ...
##  $ 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            : chr  "" "" "" "" ...
##  $ 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            : chr  "" "" "" "" ...
##  $ 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      : chr  "" "" "" "" ...
##  $ 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      : chr  "" "" "" "" ...
##  $ kurtosis_picth_arm     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_arm       : chr  "" "" "" "" ...
##  $ skewness_roll_arm      : chr  "" "" "" "" ...
##  $ skewness_pitch_arm     : chr  "" "" "" "" ...
##  $ skewness_yaw_arm       : chr  "" "" "" "" ...
##  $ 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 : chr  "" "" "" "" ...
##  $ kurtosis_picth_dumbbell: chr  "" "" "" "" ...
##  $ kurtosis_yaw_dumbbell  : chr  "" "" "" "" ...
##  $ skewness_roll_dumbbell : chr  "" "" "" "" ...
```

```
##  $ skewness_pitch_dumbbell : chr  "" "" "" "" ...
##  $ skewness_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ 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        : chr  "" "" "" "" ...
##  $ 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        : chr  "" "" "" "" ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##    [list output truncated]
```

```
str(test, 5)
```

```
## 'data.frame':    20 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "pedro" "jeremy" "jeremy" "adelmo" ...
##  $ raw_timestamp_part_1    : int  1323095002 1322673067 1322673075 1322832789 1322489635 1322673149 
##  $ raw_timestamp_part_2    : int  868349 778725 342967 560311 814776 510661 766645 54671 916313 3842
##  $ cvtd_timestamp          : chr  "05/12/2011 14:23" "30/11/2011 17:11" "30/11/2011 17:11" "02/12/20
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt               : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt              : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
##  $ yaw_belt                : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1 ...
##  $ total_accel_belt        : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ kurtosis_roll_belt      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ max_picth_belt          : logi  NA NA NA NA NA NA ...
##  $ max_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ min_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ min_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ min_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : logi  NA NA NA NA NA NA ...
##  $ avg_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : logi  NA NA NA NA NA NA ...
##  $ var_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : logi  NA NA NA NA NA NA ...
##  $ var_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : logi  NA NA NA NA NA NA ...
##  $ var_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...
##  $ gyros_belt_y            : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...
##  $ gyros_belt_z            : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...
##  $ accel_belt_x            : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y            : int  69 11 -1 45 4 -16 2 -2 1 63 ...
```

```
##  $ accel_belt_z           : int  -179 39 49 -156 27 38 35 42 32 -158 ...
##  $ magnet_belt_x          : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y          : int  581 636 631 608 566 638 622 635 600 601 ...
##  $ magnet_belt_z          : int  -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...
##  $ roll_arm               : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm              : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm                : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm        : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ var_accel_arm          : logi  NA NA NA NA NA NA ...
##  $ avg_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_arm        : logi  NA NA NA NA NA NA ...
##  $ var_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_arm       : logi  NA NA NA NA NA NA ...
##  $ var_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_arm         : logi  NA NA NA NA NA NA ...
##  $ var_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ gyros_arm_x            : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...
##  $ gyros_arm_y            : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...
##  $ gyros_arm_z            : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...
##  $ accel_arm_x            : int  16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...
##  $ accel_arm_y            : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z            : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x           : int  -326 -325 -264 -173 -170 396 702 535 -367 -420 ...
##  $ magnet_arm_y           : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z           : int  481 434 413 633 617 516 217 385 520 493 ...
##  $ kurtosis_roll_arm      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_arm      : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_arm     : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_arm       : logi  NA NA NA NA NA NA ...
##  $ max_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ max_picth_arm          : logi  NA NA NA NA NA NA ...
##  $ max_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ min_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ min_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ min_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_arm     : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm    : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm      : logi  NA NA NA NA NA NA ...
##  $ roll_dumbbell          : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell         : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell           : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ kurtosis_roll_dumbbell : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell: logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell: logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ max_picth_dumbbell     : logi  NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell       : logi  NA NA NA NA NA NA ...
```

```
##  $ min_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : logi  NA NA NA NA NA NA ...
##   [list output truncated]
```

The full training dataset will be split into a training dataset (`Atrain`) and a testing? validation dataset (`Atest`). The testing data will be used to cross validate our models. To clean the data we get rid of NA's and near zero variables and blank spaces. Here we get the indexes of the columns having at least 70% of NA or blank values on the training dataset

```
indColToRemove <- which(colSums(is.na(train) | train == "") > 0.7*dim(train)[1])

cleantrain <- train[,-indColToRemove]
cleantrain <- cleantrain[,-c(1:7)]
dim(cleantrain)
```

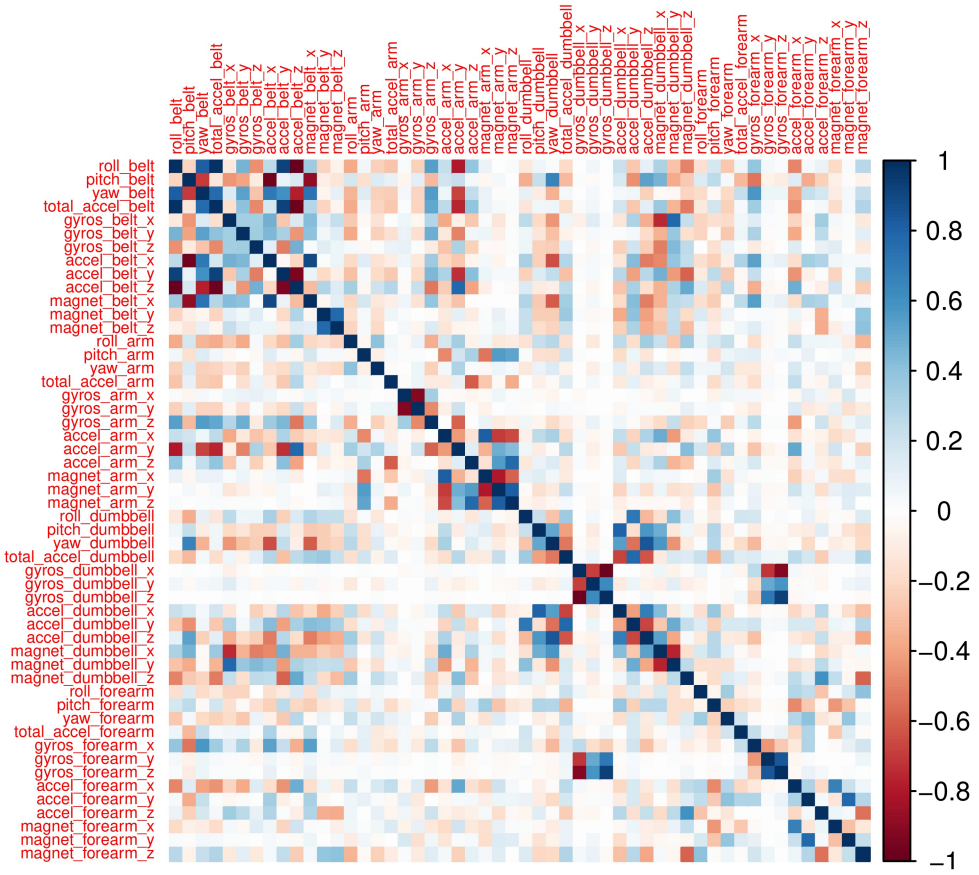
```
## [1] 19622    53
```

```
# We do the same for the test set
indColToRemove <- which(colSums(is.na(test) | test == "") >0.7*dim(test)[1])
cleantest <- test[,-indColToRemove]
cleantest <- cleantest[,-c(1:7)]
dim(cleantest)
```

```
## [1] 20 53
```

After cleaning our training data set is reduced to a data frame of 19622 observations with 53 features; while our test data set is reduced to 20 observations of 53 features. To proceed , we will explore our taining data set by plotting the correlation matrix graphs of our training data set.

```
corrplot(cor(cleantrain[, -length(names(cleantrain))]), method = "color", tl.cex = 0.5)
```



DATA PREPARATION FOR PREDICTION ———————- Preparing the training data for prediction by splitting the training data `cleantrain` into 70% as train data `Atrain` and 30% as test data `Atest`. This splitting will serve also to compute the out-of-sample errors. The split training data is renamed `Atrain` for the part that will be used for training and the cross validation data renamed as `Atest` (validate data) will stay as it is and will be used later to test the prediction algorithm on the 20 cases. The `Atrain` is a dataframe of 13737 observations of 53 variables, while `Atest` is a dataframe of 5885 obs. of 53 variables. Hence the spliting is accurately done.

```
set.seed(123)
cleantrain_split <- createDataPartition(y = cleantrain$classe, p = 0.70, list = FALSE)
Atrain <- cleantrain[cleantrain_split, ]
Atest <- cleantrain[-cleantrain_split, ]

str(Atrain, 5)
```

```
## 'data.frame':    13737 obs. of  53 variables:
##  $ roll_belt            : num  1.41 1.41 1.48 1.42 1.43 1.45 1.43 1.42 1.42 1.48 ...
##  $ pitch_belt           : num  8.07 8.07 8.07 8.09 8.16 8.17 8.18 8.2 8.21 8.15 ...
##  $ 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 ...
##  $ gyros_belt_x         : num  0 0.02 0.02 0.02 0.02 0.03 0.02 0.02 0.02 0 ...
##  $ gyros_belt_y         : num  0 0 0.02 0 0 0 0 0 0 0 ...
##  $ gyros_belt_z         : num  -0.02 -0.02 -0.02 -0.02 -0.02 0 -0.02 0 -0.02 0 ...
##  $ accel_belt_x         : int  -21 -22 -21 -22 -20 -21 -22 -22 -22 -21 ...
##  $ accel_belt_y         : int  4 4 2 3 2 4 2 4 4 4 ...
##  $ accel_belt_z         : int  22 22 24 21 24 22 23 21 21 23 ...
##  $ magnet_belt_x        : int  -3 -7 -6 -4 1 -3 -2 -3 -8 0 ...
```

```
##  $ magnet_belt_y       : int  599 608 600 599 602 609 602 606 598 592 ...
##  $ magnet_belt_z       : int  -313 -311 -302 -311 -312 -308 -319 -309 -310 -305 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -129 ...
##  $ pitch_arm           : num  22.5 22.5 22.1 21.9 21.7 21.6 21.5 21.4 21.4 21.3 ...
##  $ 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 ...
##  $ gyros_arm_x         : num  0 0.02 0 0 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.03 -0.03 -0.03 -0.03 -0.03 -0.02 0 0 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 0 0 -0.02 -0.02 0 -0.02 -0.03 -0.03 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -288 -288 -288 -287 -288 -289 ...
##  $ accel_arm_y         : int  109 110 111 111 109 110 111 111 111 109 ...
##  $ accel_arm_z         : int  -123 -125 -123 -125 -122 -124 -123 -124 -124 -121 ...
##  $ magnet_arm_x        : int  -368 -369 -374 -373 -369 -376 -363 -372 -371 -367 ...
##  $ magnet_arm_y        : int  337 337 337 336 341 334 343 338 331 340 ...
##  $ magnet_arm_z        : int  516 513 506 509 518 516 520 509 523 509 ...
##  $ roll_dumbbell       : num  13.1 13.1 13.4 13.1 13.2 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.4 -70.2 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -84.9 -85.1 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0.02 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 0 0 0 0 -0.02 -0.02 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -233 -232 -232 -235 -233 -234 -234 -233 ...
##  $ accel_dumbbell_y    : int  47 47 48 47 47 48 47 48 48 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -270 -269 -270 -270 -269 -268 -271 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -554 -551 -549 -558 -554 -552 -554 -554 ...
##  $ magnet_dumbbell_y   : int  293 296 292 295 292 291 291 302 295 297 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -68 -70 -65 -69 -65 -69 -68 -73 ...
##  $ roll_forearm        : num  28.4 28.3 28 27.9 27.7 27.7 27.5 27.2 27.2 27.1 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 -63.9 -63.9 -64 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -151 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.02 0.02 0.03 0.02 0.02 0 0 0.02 ...
##  $ gyros_forearm_y     : num  0 0 0 0 0 0 0.02 0 -0.02 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.03 -0.03 -0.03 0 ...
##  $ accel_forearm_x     : int  192 192 189 195 193 190 191 193 193 194 ...
##  $ accel_forearm_y     : int  203 203 206 205 204 205 203 205 202 204 ...
##  $ accel_forearm_z     : int  -215 -216 -214 -215 -214 -215 -215 -215 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -17 -18 -16 -22 -11 -15 -14 -13 ...
##  $ magnet_forearm_y    : num  654 661 655 659 653 656 657 655 659 656 ...
##  $ magnet_forearm_z    : num  476 473 473 470 476 473 478 472 478 471 ...
##  $ classe              : chr  "A" "A" "A" "A" ...
```

```
str(Atest, 5)
```

```
## 'data.frame':    5885 obs. of  53 variables:
##  $ roll_belt           : num  1.42 1.48 1.45 1.42 1.45 1.45 1.57 1.56 1.51 1.43 ...
##  $ pitch_belt          : num  8.07 8.05 8.06 8.13 8.18 8.2 8.09 8.1 8.1 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.3 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0.02 0.02 0.03 0 0.02 0.02 0.02 0 ...
##  $ gyros_belt_y        : num  0 0 0 0 0 0 0.02 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.03 -0.02 -0.02 -0.02 0 -0.02 -0.02 -0.02 -0.03 ...
##  $ accel_belt_x        : int  -20 -22 -21 -22 -21 -21 -21 -21 -20 -22 ...
##  $ accel_belt_y        : int  5 3 4 4 2 2 3 4 4 4 ...
```

```
##  $ accel_belt_z        : int  23 21 21 21 23 22 21 21 22 19 ...
##  $ magnet_belt_x       : int  -2 -6 0 -2 -5 -1 -2 -4 -3 4 ...
##  $ magnet_belt_y       : int  600 604 603 603 596 597 604 606 601 602 ...
##  $ magnet_belt_z       : int  -305 -310 -312 -313 -317 -310 -313 -311 -318 -316 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -129 -129 -129 -129 -129 ...
##  $ pitch_arm           : num  22.5 22.1 22 21.8 21.5 21.4 20.8 20.7 20.7 20.5 ...
##  $ 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 ...
##  $ gyros_arm_x         : num  0.02 0.02 0.02 0.02 0.02 0.02 0.03 0.02 -0.02 0.03 ...
##  $ gyros_arm_y         : num  -0.02 -0.03 -0.03 -0.02 -0.03 0 -0.02 -0.02 0 -0.02 ...
##  $ gyros_arm_z         : num  -0.02 0.02 0 0 0 -0.03 -0.02 -0.02 -0.02 0 ...
##  $ accel_arm_x         : int  -289 -289 -289 -289 -290 -289 -289 -290 -289 -290 ...
##  $ accel_arm_y         : int  110 111 111 111 110 111 111 110 110 110 ...
##  $ accel_arm_z         : int  -126 -123 -122 -124 -123 -124 -123 -123 -125 -126 ...
##  $ magnet_arm_x        : int  -368 -372 -369 -372 -366 -374 -372 -373 -374 -375 ...
##  $ magnet_arm_y        : int  344 344 342 338 339 342 338 333 350 339 ...
##  $ magnet_arm_z        : int  513 512 513 510 509 510 510 509 516 508 ...
##  $ roll_dumbbell       : num  12.9 13.4 13.4 12.8 13.1 ...
##  $ pitch_dumbbell      : num  -70.3 -70.4 -70.8 -70.3 -70.6 ...
##  $ yaw_dumbbell        : num  -85.1 -84.9 -84.5 -85.1 -84.7 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 -0.02 0 0 0 0 0 0 0 -0.02 ...
##  $ accel_dumbbell_x    : int  -232 -232 -234 -234 -233 -234 -233 -234 -235 -234 ...
##  $ accel_dumbbell_y    : int  46 48 48 46 47 47 48 48 47 48 ...
##  $ accel_dumbbell_z    : int  -270 -269 -269 -272 -269 -270 -270 -270 -271 -272 ...
##  $ magnet_dumbbell_x   : int  -561 -552 -558 -555 -564 -554 -554 -557 -558 -556 ...
##  $ magnet_dumbbell_y   : int  298 303 294 300 299 294 301 294 291 298 ...
##  $ magnet_dumbbell_z   : num  -63 -60 -66 -74 -64 -63 -65 -69 -71 -62 ...
##  $ roll_forearm        : num  28.3 28.1 27.9 27.8 27.6 27.2 27 26.9 27.1 26.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.8 -63.8 -63.9 -63.9 -63.8 -63.7 -63.7 ...
##  $ yaw_forearm         : num  -152 -152 -152 -152 -152 -151 -151 -151 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.02 0.02 0.02 0 0.02 0.02 0.03 0 ...
##  $ gyros_forearm_y     : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.03 -0.02 -0.03 -0.02 ...
##  $ gyros_forearm_z     : num  0 0 -0.03 0 -0.02 -0.02 -0.02 -0.02 0 -0.02 ...
##  $ accel_forearm_x     : int  196 189 193 193 193 192 191 194 193 196 ...
##  $ accel_forearm_y     : int  204 206 203 205 205 201 206 206 203 207 ...
##  $ accel_forearm_z     : int  -213 -214 -215 -213 -214 -214 -213 -214 -213 -216 ...
##  $ magnet_forearm_x    : int  -18 -16 -9 -9 -17 -16 -17 -10 -11 -15 ...
##  $ magnet_forearm_y    : num  658 658 660 660 657 656 654 653 661 650 ...
##  $ magnet_forearm_z    : num  469 469 478 474 465 472 478 467 470 473 ...
##  $ classe              : chr  "A" "A" "A" "A" ...
```

The Dataset now consists of 53 variables with the observations divided as following: 1. Training Data (Atrain): 13737 observations. 2. Validation Data (Atest): 5885 observations. 3. Testing Data: 20 observations.

DATA MODELLING AND MACHINE LEARNING

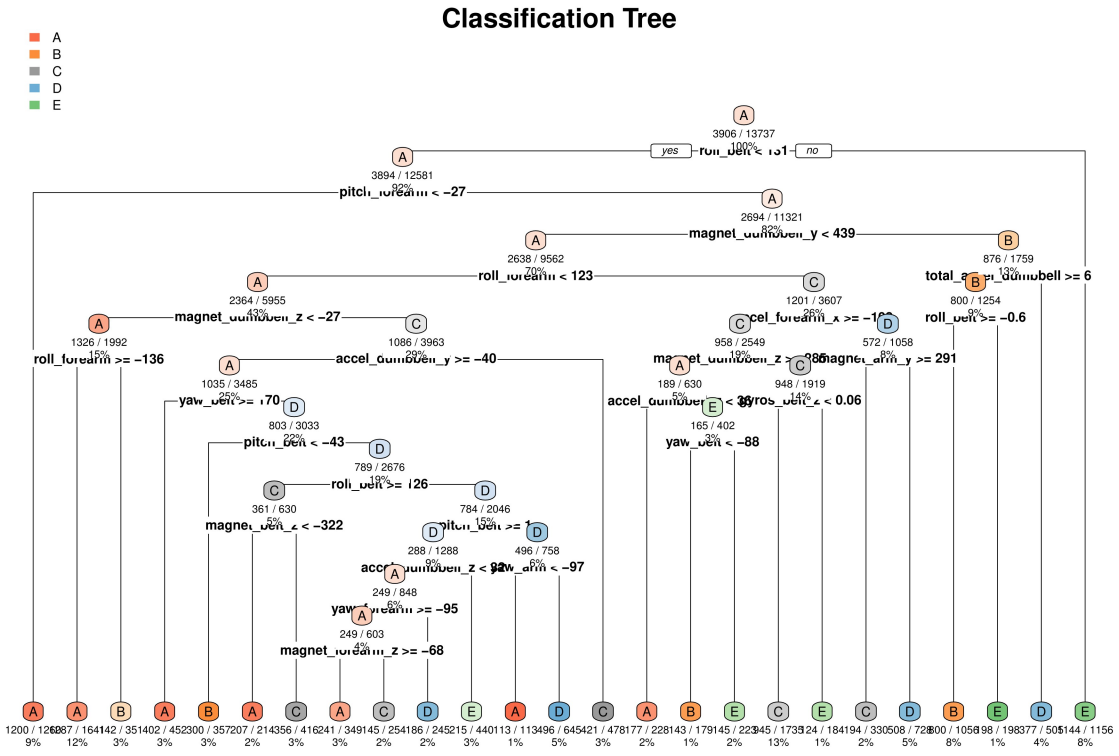
1. DECISION TREE

We fit a predictive model for activity recognition using Decision Tree algorithm and we estimate the performance of the model on the validation data set.

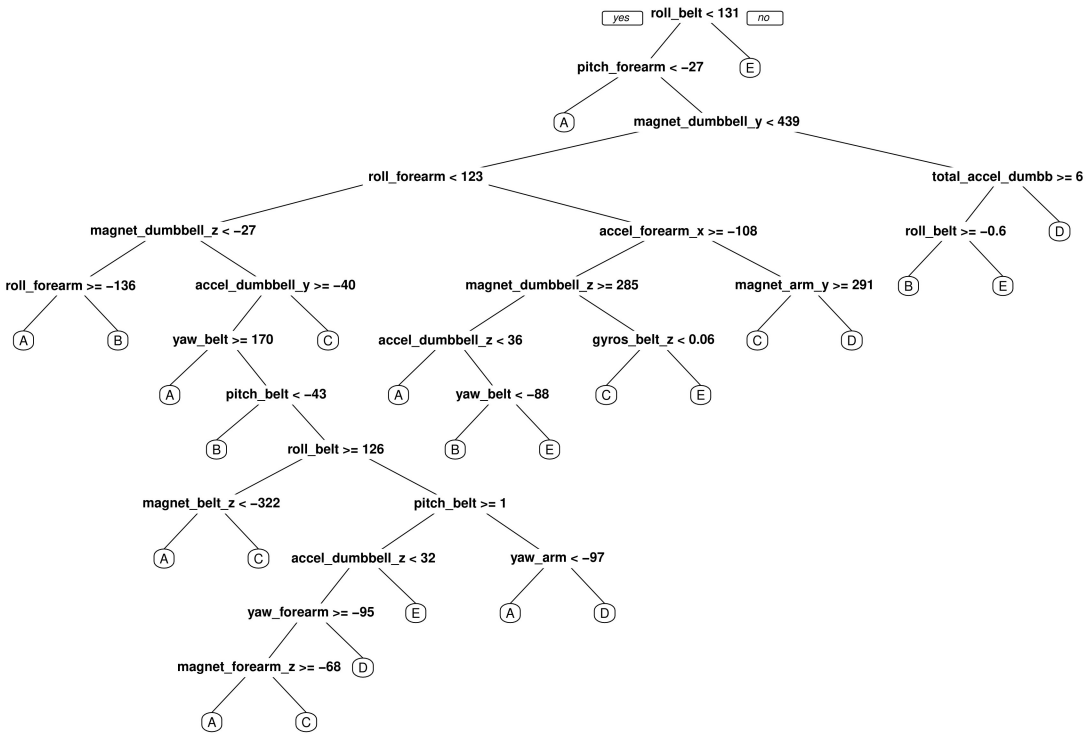
```
decitree <- rpart(classe ~ ., data = Atrain, method = "class")

# Validate the decison tre  model with the validation data set
predictTree <- predict(decitree, Atest, type = "class")

# Plot out our model
rpart.plot(decitree, main = "Classification Tree", extra = 102, under = TRUE, faclen = 0, cex = .4)
```




```
prp(decitree)
```




```
# calcaulate the confusion matrix
confusionMatrix(factor(Atest$classe), predictTree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1552   48   39   24   11
##          B  174  588  220   83   74
##          C   18   43  888   75    2
##          D   60   63  100  651   90
##          E    6   64  148   86  778
##
## Overall Statistics
##
##               Accuracy : 0.7573
##                 95% CI : (0.7462, 0.7683)
##    No Information Rate : 0.3076
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.6926
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8575  0.72953   0.6366   0.7084   0.8147
## Specificity            0.9701  0.89151   0.9693   0.9370   0.9383
## Pos Pred Value         0.9271  0.51624   0.8655   0.6753   0.7190
## Neg Pred Value         0.9387  0.95407   0.8957   0.9455   0.9631
## Prevalence             0.3076  0.13696   0.2370   0.1562   0.1623
## Detection Rate         0.2637  0.09992   0.1509   0.1106   0.1322
## Detection Prevalence   0.2845  0.19354   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9138  0.81052   0.8029   0.8227   0.8765
```

```
# check the accuracy and standard error
deciAccu <- postResample(predictTree, factor(Atest$classe))
decioutSE <- 1 - as.numeric(confusionMatrix(factor(Atest$classe),predictTree)$overall[1])
deciAccu
```

```
##  Accuracy     Kappa
## 0.7573492 0.6925606
```

```
decioutSE
```

```
## [1] 0.2426508
```

```
percent(deciAccu)
```

```
## Accuracy    Kappa
##  "75.7%"  "69.3%"
```

```
percent(decioutSE)
```

```
## [1] "24%"
```

The estimated out of sample error with the cross validation dataset for this model is 24%. Accuracy for the decision tree is 75.7% with kappa as 69.3% which is an indication of substantial (in other words problematic and not good enough), hence we will try some other model- Random forest

2. RANDOM FOREST

Set the random forest model with training and cross validation data

```
RFpml <- randomForest(factor(classe)~.,data = Atrain, ntree = 250, keep.forest = T, xtest = Atest[-53],

RFpml
```

```
##
## Call:
##  randomForest(formula = factor(classe) ~ ., data = Atrain, ntree = 250,      keep.forest = T, xtest =
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.52%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3899    4    1    1    1 0.001792115
## B   10 2645    3    0    0 0.004890895
## C    0   15 2379    2    0 0.007095159
## D    0    0   23 2227    2 0.011101243
## E    0    0    2    7 2516 0.003564356
##                 Test set error rate: 0.44%
```

```
## Confusion matrix:
##      A    B    C   D    E class.error
## A 1674    0    0   0    0 0.000000000
## B    4 1132    3   0    0 0.006145742
## C    0    4 1022   0    0 0.003898635
## D    0    0    7 957    0 0.007261411
## E    0    0    4   4 1074 0.007393715
```

```
# validate model with the validation test data set
rfpred <- predict(RFpml, Atest, type = "class")

# calculate the confusion matrix
confusionMatrix(rfpred, factor(Atest$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1132    4    0    0
##          C    0    3 1022    7    4
##          D    0    0    0  957    4
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9939   0.9961   0.9927   0.9926
## Specificity            0.9991   0.9992   0.9971   0.9992   1.0000
## Pos Pred Value         0.9976   0.9965   0.9865   0.9958   1.0000
## Neg Pred Value         1.0000   0.9985   0.9992   0.9986   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1924   0.1737   0.1626   0.1825
## Detection Prevalence   0.2851   0.1930   0.1760   0.1633   0.1825
## Balanced Accuracy      0.9995   0.9965   0.9966   0.9960   0.9963
```

```
# estimate the accuracy and out of sample error
accuracy <- postResample(Atest$classe, rfpred)
ose <- 1 - as.numeric(confusionMatrix(rfpred,factor(Atest$classe))$overall[1])

accuracy
```

```
##  Accuracy     Kappa
## 0.9955820 0.9944114
```

```
ose
```

```
## [1] 0.004418012
```

```
mean(RFpml$err.rate)
```

```
## [1] 0.01046482
```

```
mean(RFpml$test$err.rate)
```

```
## [1] 0.007403291
```

In-Sample Error Rate Is the error rate for the predictions for input data (training data i.e `Atrain`) to the model. `OOB` error rate is 0.016. The `in-sample error rate` is 0.01046482. `Out Of Sample Error Rate` which is the error rate for predictions on the cross validated data provided to the model. It is normally higher than the in-sample error rate and is 0.007403291. A close observation of the accuracy for the `random forest` model shows 99.58% and a 99% `kappa` with an `out-of-sample-error` of about 0.004 which is highly negligible.

we will simply try a third model just to be sure that we are actually gett ing what we desire from the `Random forest` model.

3. PREDICTIONS WITH GENERALIZED BOOSTED RESGRESSION MODEL

```
set.seed(2020)
# set control parameter
gbmcntl <- trainControl(method = "repeatedcv", number = 5, repeats = 1)

#the gbm
gbmmod  <- train(classe ~ ., data = Atrain, method = "gbm", trControl = gbmcntl, verbose = FALSE)

gbmmod$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

```
print(gbmmod)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10989, 10989, 10989, 10990, 10991
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7535853  0.6878119
##   1                  100      0.8194660  0.7715761
##   1                  150      0.8525149  0.8133894
##   2                   50      0.8560101  0.8176176
##   2                  100      0.9057300  0.8807080
```

```
##   2                  150      0.9320822  0.9140652
##   3                   50      0.8948098  0.8668407
##   3                  100      0.9424913  0.9272318
##   3                  150      0.9604727  0.9499917
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
#Validate the GBM model and
gbmpred <- predict(gbmmod, newdata = Atest)
gbmconfusion <- confusionMatrix(factor(Atest$classe), gbmpred)
gbmconfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1648   17    6    1    2
##          B   30 1075   33    1    0
##          C    0   35  973   16    2
##          D    1    6   28  920    9
##          E    3    8   19   18 1034
##
## Overall Statistics
##
##                Accuracy : 0.9601
##                  95% CI : (0.9547, 0.9649)
##     No Information Rate : 0.2858
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9495
##
##  Mcnemar's Test P-Value : 9.35e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9798   0.9422   0.9188   0.9623   0.9876
## Specificity            0.9938   0.9865   0.9890   0.9911   0.9901
## Pos Pred Value         0.9845   0.9438   0.9483   0.9544   0.9556
## Neg Pred Value         0.9919   0.9861   0.9823   0.9927   0.9973
## Prevalence             0.2858   0.1939   0.1799   0.1624   0.1779
## Detection Rate         0.2800   0.1827   0.1653   0.1563   0.1757
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9868   0.9643   0.9539   0.9767   0.9888
```

```
accuracyGBM <- postResample(Atest$classe, gbmpred)
oseGBM <- 1 - as.numeric(confusionMatrix(gbmpred, factor(Atest$classe))$overall[1])

accuracyGBM
```

```
## Accuracy    Kappa
## 0.960068 0.949484
```

```
oseGBM
```

```
## [1] 0.03993203
```

A gradient boosted generalized regression model with multinomial loss function is computed and 150 iterations were performed. There were 53 predictors of which 52 had non-zero influence. Also observing the accuaracy rate and out of sample of the `gbm` the sample is given as `96%` and `0.0366`.

Therefore we conclude that the `Random forest` model is out best choice for this work.

PREDICTION FOR THE TEST VALIDATION DATA

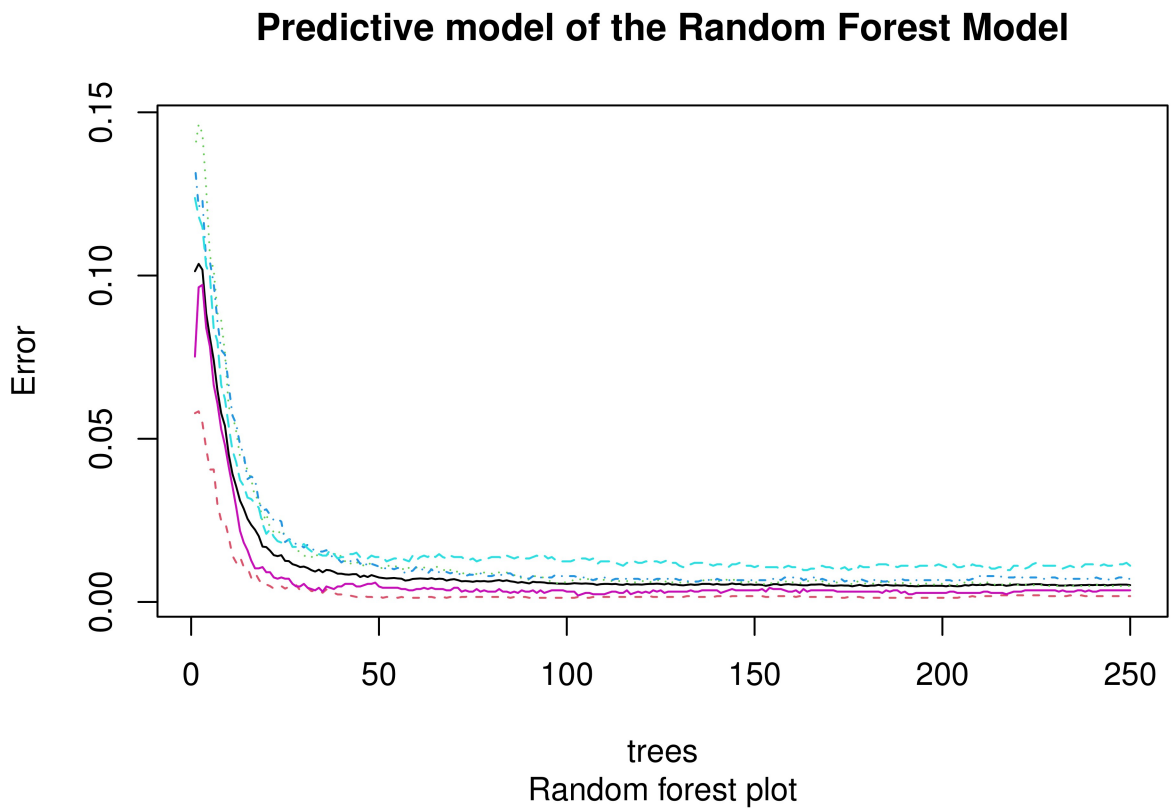
Prediction for the test data is provided by the following line of code:

```
result <- predict(RFpml, cleantest)
result
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

```
# plot out the predicted result

plot(RFpml, main="Predictive model of the Random Forest Model",  sub = "Random forest plot")
```



There is a certain element of randomness to randomforest, hence, hardware, operating system and time of use may effect the final answer. In the predicted plot above shows that increasing the number of trees improves accuracy, so increasing the number of trees, ntree would improve the accuracy of predictions. From the plot, it is ap-

parent that error rates improve slowly for trees beyond about 50. Use the following answer key to check your output. `result=c("B","A","B","A","A","E","D","B","A","A","B","C","B","A","E","E","A","B","B","B")`