Practical Machine Learning - Final Project

Aleksa Lakic

21/05/2020

This document is my final project for the Coursera's "Practical Machine Learning" course. It was produced using R Markdown and the Knit functionality on Rstudio.

I - Overview of the project

1. Introduction

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.

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. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

2. Objective

In this project, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. This is the "classe" variable in the training set.

II - Methodology for the data collection and preparation

1. Preparation of the Environment

Installation/loading of the packages necessary for the project

The packages were installed by using the function install.packages("name of the library"). Example: install.packages("rpart").

The following libraries were used for this project:

```r
library(caret)
```

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

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

```r
library(e1071)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Entrez 'rattle()' pour secouer, faire vibrer, et faire défiler vos données.
```

```r
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

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

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```r
library(lattice)
library(ggplot2)
library(rmarkdown)
```

2. Data preparation

Download the data

```r
# Upload the date from the internet
Training_URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Testing_URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
# Read the data on RStudio
Train_Raw_Data <- read.csv(url(Training_URL), header = TRUE)
Valid_Raw_Data <- read.csv(url(Testing_URL), header = TRUE)
```

Note: The datasets used in this project are available thanks to W.Ugilino, D. Cardador, K. Vega, E. Velloso, R. Milidiu, H. Fuks of their document called "Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements".

```
#Show the data
str(Train_Raw_Data)
```

```
## '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 4844
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/201
##  $ 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]
```

```r
str(Valid_Raw_Data)
```

```
## '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 1
##  $ raw_timestamp_part_2    : int  868349 778725 342967 560311 814776 510661 766645 54671 916313 38428
##  $ cvtd_timestamp          : chr  "05/12/2011 14:23" "30/11/2011 17:11" "30/11/2011 17:11" "02/12/201
##  $ 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]
```

```r
#The dimension of the two dataset
dim(Train_Raw_Data)
```

```
## [1] 19622   160
```

```r
dim(Valid_Raw_Data)
```

```
## [1]  20 160
```

- Both datasets have 160 variables
- Training dataset: 19622 Observations
- Validation dataset: 20 observations

The "Train_Raw_Data" dataset will be used as an input to create the dataset for the prediction models.

The dataset "Valid_Raw_Data" will be used to test the prediction model on the 20 test cases.

This partiioning will work for determining the out-of-sample errors.

3. Data Partitioning

We partitioned the traning dataset called "pml-training.csv" into two datasets: * 70% for the training dataset * 30% for the testing dataset

```r
set.seed(28765)
TrainingSample <- createDataPartition(Train_Raw_Data$classe, p = 0.7, list = FALSE)
Train_Set <- Train_Raw_Data[TrainingSample, ]
Test_Set <- Train_Raw_Data[-TrainingSample, ]
dim(Train_Set)
```

```
## [1] 13737   160
```

```r
dim(Test_Set)
```

```
## [1] 5885  160
```

Both datasets have 160 variables. They includes many missing values and the first 7 columns have little impact on the variable "classe". Some columns have values close to 0. Thus, we need to clean and prepare the data.

3. Clean and prepare the data

Remove variables that contains missing values

```r
#remove variables that contains missing values
Train_Set <- Train_Set[, colSums(is.na(Train_Set)) == 0]
Test_Set <- Test_Set[, colSums(is.na(Test_Set)) == 0]
#Dimension of both sets of data
dim(Train_Set)
```

```
## [1] 13737    93
```

```
dim(Test_Set)
```

```
## [1] 5885   93
```

Remove variables that would have little impact on the variable "classe"

```
#remove variables in the uploaded data of the training dataset
Train_Set <- Train_Set[, -c(1:7)]
dim(Train_Set)
```

```
## [1] 13737    86
```

```
#remove variables in the validation dataset
Test_Set <- Test_Set[, -c(1:7)]
dim(Test_Set)
```

```
## [1] 5885   86
```

Cleaning by removing the variables that have a near zero variance

```
AZV <- nearZeroVar(Train_Set)
Train_Set <- Train_Set[, -AZV]
Test_Set <- Test_Set[, -AZV]
dim(Train_Set)
```

```
## [1] 13737    53
```

```
dim(Test_Set)
```

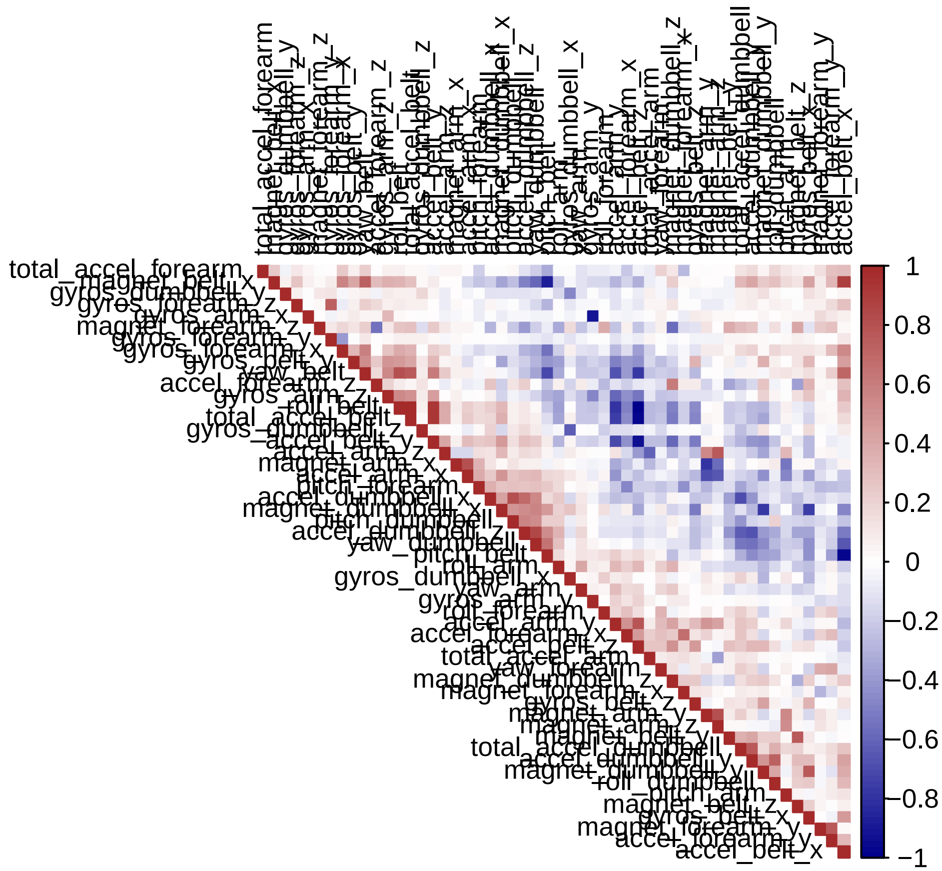
```
## [1] 5885   53
```

By doing these manipulations, we have 53 variables now. How are these variables correlated between them?

4. Correlation between variables

Compute the correlation between the different variables

```
corMatrix <- cor(Train_Set[,-53])
colors<- colorRampPalette(c("darkblue", "white","brown"))(100) #Colors for the matrix
corrplot(corMatrix, order = "AOE", method = "color",
         type = "upper", tl.cex = 0.8, tl.col = "black", col = colors) #Plot the correlation matrix
```



In this graph, we observe the correlation between the different variables. Variables which have dark blue or dark brown intersections with others are the most correlated between them.

Find the names of the variables that are highly correlated

```
highCor = findCorrelation(corMatrix, cutoff = 0.8) # The 20% most correlated variables
names(Train_Set)[highCor] # Show the names of the highly correlated variables
```

```
##  [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"     "accel_dumbbell_z"
##  [5] "accel_belt_x"     "pitch_belt"       "accel_dumbbell_x" "accel_arm_x"
##  [9] "magnet_arm_y"     "gyros_arm_x"
```

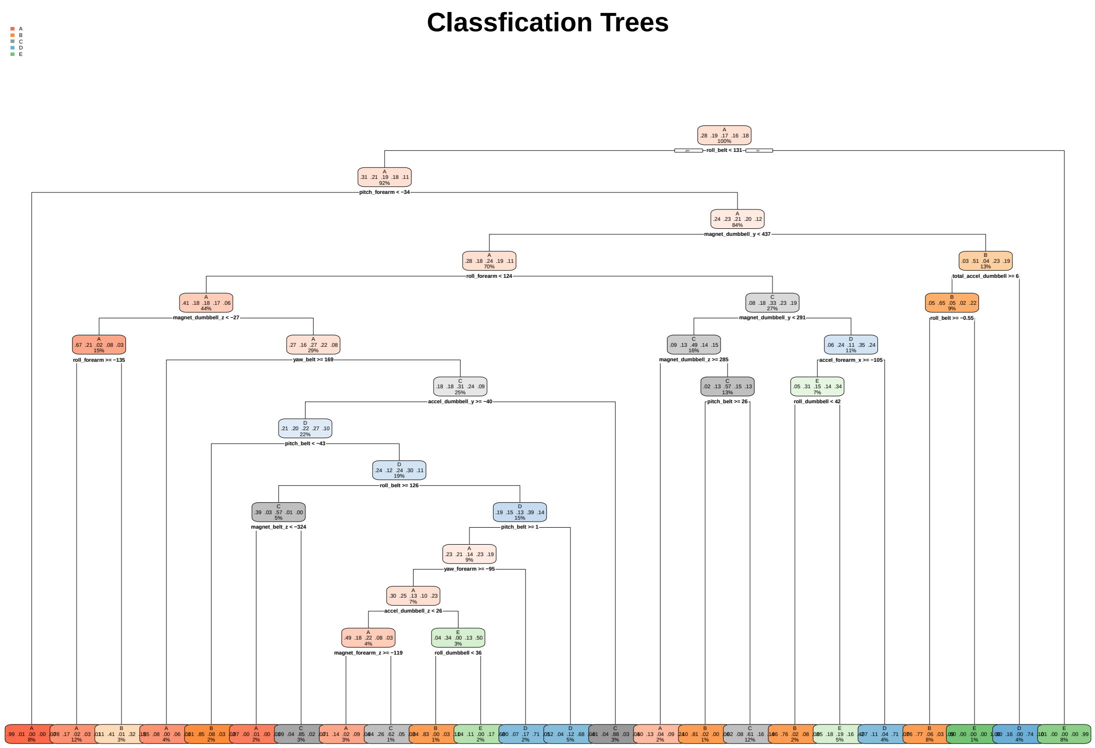
III - Prediction Model Building

1. Classification trees' model (class)

Set the model"'

```
set.seed(28765)
ModelTree <- rpart(classe ~ ., data=Train_Set, method="class") #use of the rpart package
rpart.plot(ModelTree, main="Classfication Trees")  #Plot the classification trees as a dendogram
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Classification Trees' Prediction Model

We use the model on the Test_Data and determine the accuracy of the model.

```
PredictClassTree <- predict(ModelTree, Test_Set, type = "class") #Predict function for this model
CMtree <- confusionMatrix(PredictClassTree, as.factor(Test_Set$classe)) #Create the matrix of the predi
CMtree # Show the matrix
```

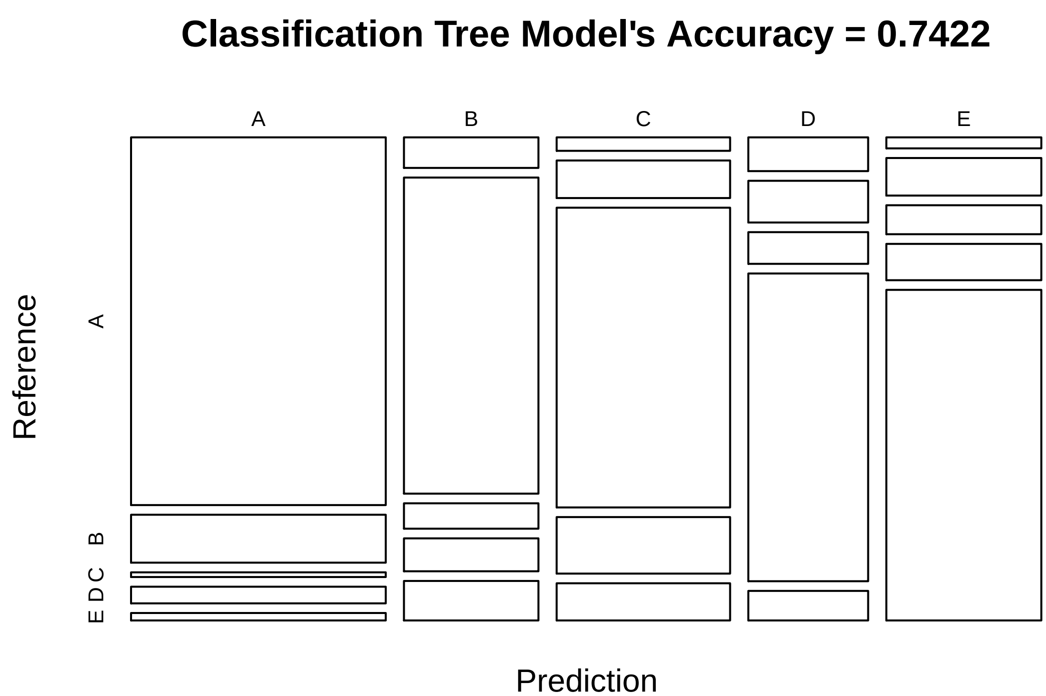
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1481  193   19   67   30
##          B   65  672   54   70   84
##          C   37  103  822  155  102
##          D   64   79   60  583   56
##          E   27   92   71   89  810
##
## Overall Statistics
##
##                Accuracy : 0.7422
##                  95% CI : (0.7308, 0.7534)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6732
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8847   0.5900   0.8012  0.60477   0.7486
## Specificity            0.9266   0.9425   0.9183  0.94737   0.9419
## Pos Pred Value         0.8274   0.7111   0.6743  0.69240   0.7438
## Neg Pred Value         0.9529   0.9055   0.9563  0.92445   0.9433
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2517   0.1142   0.1397  0.09907   0.1376
## Detection Prevalence   0.3042   0.1606   0.2071  0.14308   0.1850
## Balanced Accuracy      0.9057   0.7662   0.8597  0.77607   0.8453
```

- The accuracy rate of the model is low: 74.22%
- The out-of-sample error is high (around 25.78%)

Plot the confusion matrix

```
plot(CMtree$table, col = CMtree$byClass,
     main = paste("Classification Tree Model's Accuracy =",
                  round(CMtree$overall["Accuracy"], 4)))
```



2. Gradient Boosting Model (gbm)

Set the model

```r
set.seed(28765)
control_gbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFit_gbm  <- train(classe ~ ., data=Train_Set, method = "gbm",
                    trControl = control_gbm, verbose = FALSE)
modFit_gbm$finalModel
```

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

Gradient Boosting Model prediction

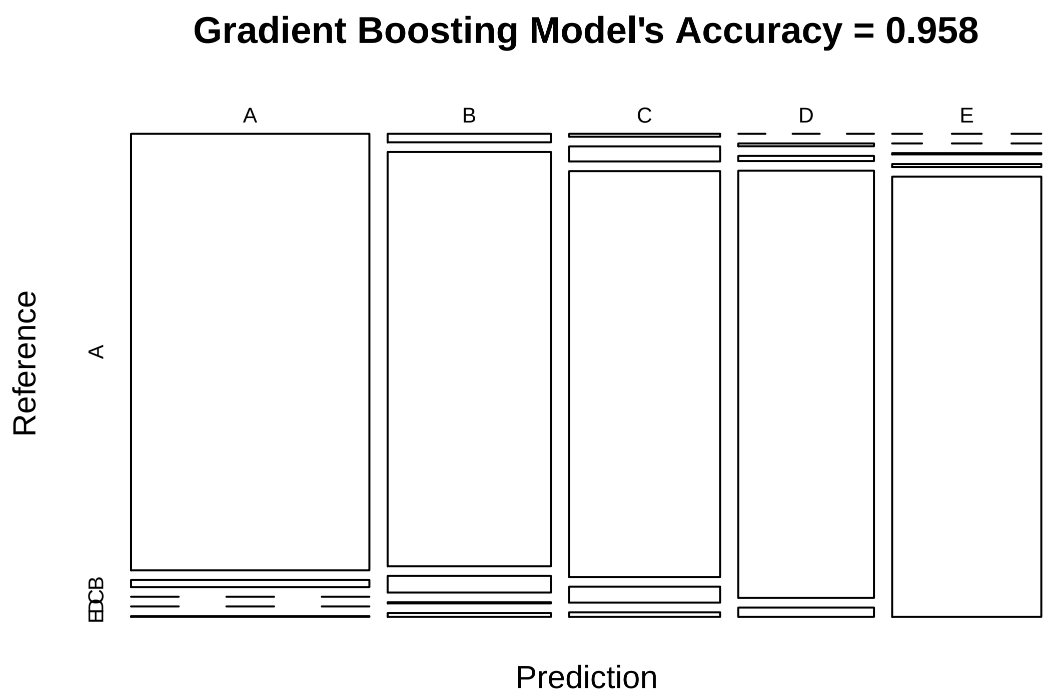
```r
predict_gbm <- predict(modFit_gbm, Test_Set) #Predict function for the gbm model
confMatrix_gbm <- confusionMatrix(predict_gbm, as.factor(Test_Set$classe))#Create the matrix of the pre
confMatrix_gbm #Show the matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1645   27    0    0    3
##          B   22 1070   43    3   10
##          C    7   36  969   38   11
##          D    0    6   11  916   20
##          E    0    0    3    7 1038
##
## Overall Statistics
##
##                Accuracy : 0.958
##                  95% CI : (0.9526, 0.963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9469
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9827   0.9394   0.9444   0.9502   0.9593
## Specificity            0.9929   0.9836   0.9811   0.9925   0.9979
## Pos Pred Value         0.9821   0.9321   0.9133   0.9612   0.9905
## Neg Pred Value         0.9931   0.9854   0.9882   0.9903   0.9909
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2795   0.1818   0.1647   0.1556   0.1764
## Detection Prevalence   0.2846   0.1951   0.1803   0.1619   0.1781
## Balanced Accuracy      0.9878   0.9615   0.9628   0.9713   0.9786
```

- The accuracy rate of the model is high: 95.80%
- The out-of-sample error is low (around 4.20%)

Plot the confusion matrix

```
plot(confMatrix_gbm$table, col = confMatrix_gbm$byClass,
     main = paste("Gradient Boosting Model's Accuracy =", round(confMatrix_gbm$overall["Accuracy"], 4)))
```



3. Random Forest Model (rf)

Set the model

```
set.seed(28765)
CRF <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
RF_Model <- train(classe ~ ., data = Train_Set, method = "rf", trControl = CRF)
RF_Model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.82%
## Confusion matrix:
##      A    B    C    D    E  class.error
```

```
## A 3903    3    0    0    0 0.0007680492
## B   17 2633    8    0    0 0.0094055681
## C    0   24 2371    1    0 0.0104340568
## D    0    0   48 2201    3 0.0226465364
## E    0    0    2    6 2517 0.0031683168
```

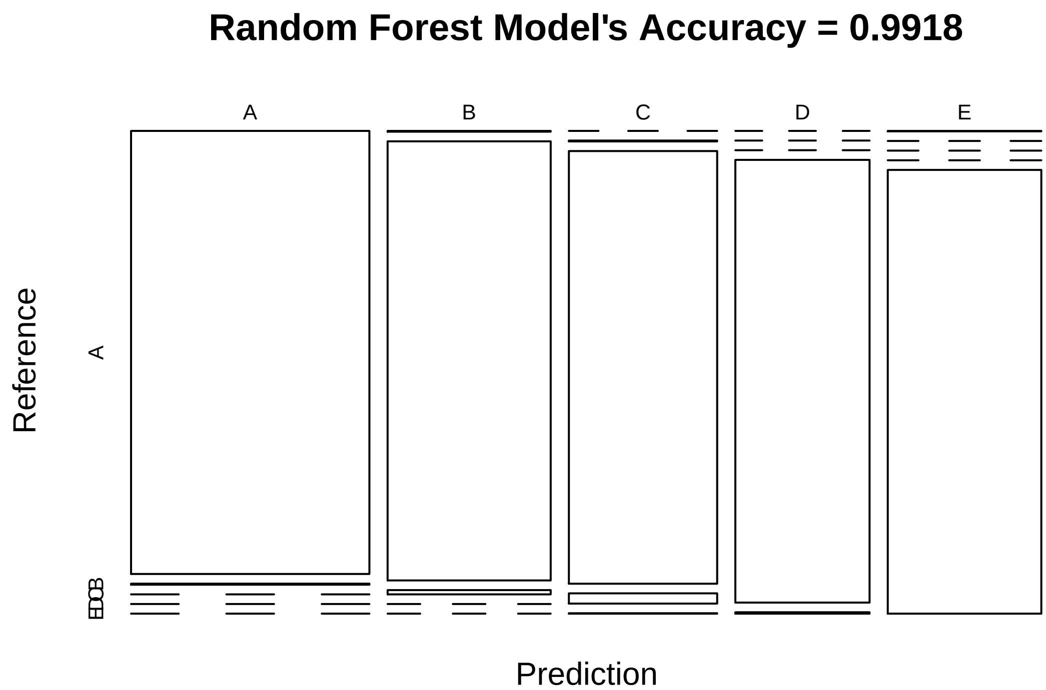
Random Forest Model prediction

```
predictRF <- predict(RF_Model, newdata = Test_Set) #Predict function for the Random Forest model
CMRF <- confusionMatrix(predictRF, as.factor(Test_Set$classe)) #Create the matrix of the prediction res
CMRF #Show the matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    4    0    0    0
##          B    2 1133   11    0    0
##          C    0    2 1015   24    1
##          D    0    0    0  940    3
##          E    1    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9918
##                  95% CI : (0.9892, 0.994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9897
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9947   0.9893   0.9751   0.9963
## Specificity            0.9991   0.9973   0.9944   0.9994   0.9998
## Pos Pred Value         0.9976   0.9887   0.9741   0.9968   0.9991
## Neg Pred Value         0.9993   0.9987   0.9977   0.9951   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1925   0.1725   0.1597   0.1832
## Detection Prevalence   0.2846   0.1947   0.1771   0.1602   0.1833
## Balanced Accuracy      0.9986   0.9960   0.9919   0.9872   0.9980
```

Plot the confusion matrix

```
plot(CMRF$table, col = CMRF$byClass,
     main = paste("Random Forest Model's Accuracy =",
                  round(CMRF$overall['Accuracy'], 4)))
```



- The accuracy rate of the model is very high: 99.18%
- The out-of-sample error is very low (around 0.82%)

IV - Conclusion and final results

1. Choose the most accurate prediction model for the validation dataset

From the previous computations, we obtain the following results:

- Classification Tree Model's Accuracy = 74.22%
- G model's accuracy = 95.80%
- Random Forest Model's Accuracy = 99.18%

As the Random Forest model is the most accurate for the prediction in-sample. Thus, we will use it to predict 20 different test cases.

2. Results

```
# The dataset "Valid_Raw_Data" contains the 20 test cases
predict_test <- predict(RF_Model , newdata=Valid_Raw_Data)
predict_test
```

```
##  [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
```

```
summary(predict_test)
```

```
## A B C D E
## 7 8 1 1 3
```

These results will be used for the Final Quiz.

3. Out-of-sample errors estimate

Note: The rf model was optimized for the initial dataset. We have the following relation verified: in-sample error < out-of-sample error. Thus, we think that the out-of-sample error is higher than 0.8% due to overfitting and lower than 6%. In fact, data have two parts: noise and signal. Reducing the noise by preparing and cleaning the data could have lead to overfitting. Hence, predictors might not work as well in new sample.