Practical Machine Learning Report_Course Project

Pengfei LI

7/10/2020

Practical Machine Learning: Prediction Assignment Writeup

Introduction

This report provides the data analysis for the Assignment Project in the *Course Practical Machine Learning*. With the help of R package knitr, this report will present the output of the analysis in the html format on both Github and RPubs.com.

The aim of this report si to build up an analytical approach for the data analysis process of the course project. It involves the prediction on the performance of 6 participants in some certain exercises. The following section will provide the algorithm of the machine learning process and submit the prediction for the Quiz in the Project Instruction.

Project Description

To collect a large amount of data about personal activity inexpensively is possible now with devices such as Jawbone Up, Nike FuelBand, and Fitbit. A group of enthusiasts use these devices to take measurements about themselves, regularly to improve their health, or to find patterns in their behavior.

One of the central issues that people regularly ask is about how much of a particular activity they do, but they rarely quantify how well they do it.

In the following sections, my 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. The collected data from this activity will be used for data analysis to answer the mentioned issues.

More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Also there are some more introductions in the following websites: http://groupware.les.inf.puc-rio.br/har

Project Analysis

Dataset Introduction

The data analyzed here is provided in the project instruction, from the following links: 1. training data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv 2. test data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://web.archive.org/web/20161224072740/http://groupware.les.inf.puc-rio.br/har.

The data from this site will be used in the following analysis to construct the model. The main collaborators of this website is

- Wallace Ugulino (wugulino at inf dot puc-rio dot br)
- Eduardo Velloso
- Hugo Fuks

And the introduction of the data source is from:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Exploratory Analysis

In this sub-section, I will start with some data pre-processing procedure and the exploratory analysis for the dataset.

Data Pre-processing

Starting with downloading the data, the solution procedure is as follows.

```
#Download the file from link in the instruction
data1 <- "training.csv"
data2 <- "testing.csv"

if(!file.exists(data1)){
        fileUrl<-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        download.file(fileUrl,data1,method = "curl")
}

if(!file.exists(data2)){
        fileUrl<-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
        download.file(fileUrl,data2,method = "curl")
}

# Load the data
train_data <- read.csv(data1, strip.white = TRUE, na.strings = c("NA",""))
test_data <- read.csv(data2, strip.white = TRUE, na.strings = c("NA",""))
```

Then, I check the data and do the primary data processing.

```
# Check the data
dim(train_data)
```

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

```
str(train_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 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  NA NA NA NA ...
##  $ kurtosis_picth_belt     : chr  NA NA NA NA ...
##  $ kurtosis_yaw_belt       : chr  NA NA NA NA ...
##  $ skewness_roll_belt      : chr  NA NA NA NA ...
##  $ skewness_roll_belt.1    : chr  NA NA NA NA ...
##  $ skewness_yaw_belt       : chr  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            : chr  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            : chr  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      : chr  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       : chr  NA NA NA NA ...
##  $ kurtosis_picth_arm      : chr  NA NA NA NA ...
##  $ kurtosis_yaw_arm        : chr  NA NA NA NA ...
##  $ skewness_roll_arm       : chr  NA NA NA NA ...
##  $ skewness_pitch_arm      : chr  NA NA NA NA ...
##  $ skewness_yaw_arm        : chr  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  : chr  NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : chr  NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : chr  NA NA NA NA ...
##  $ skewness_roll_dumbbell  : chr  NA NA NA NA ...
##  $ skewness_pitch_dumbbell : chr  NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : chr  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        : chr  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        : chr  NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##    [list output truncated]
```

```
dim(test_data)
```

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

```
str(test_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]
```

```
# Data pre-processing
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
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
```

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

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

```
library(rpart)

# Separate the dataset
training_partition <- createDataPartition(train_data$classe,p=0.75,list = FALSE)
train_subset <- train_data[training_partition,]
test_subset <- train_data[-training_partition,]

# pre-process the subset
low_var <- nearZeroVar(train_subset)
train_subset <- train_subset[,-low_var]
test_subset <- test_subset[,-low_var]

var_na <- sapply(train_subset,function(x){
                                mean(is.na(x))>0.95})
train_subset <- train_subset[,var_na==FALSE]
test_subset <- test_subset[,var_na==FALSE]
```

```
train_subset <- train_subset[ , -(1:5)]
test_subset  <- test_subset [ , -(1:5)]

dim(train_subset)
```

```
## [1] 14718    54
```

```
dim(test_subset)
```

```
## [1] 4904   54
```

The upper code creates two partitions from the training dataset. Then I start to cleanse these two subsets.

```
# Data pre-processing
library(rattle)
library(randomForest)
library(caret)
library(rpart)

# Separate the dataset
training_partition <- createDataPartition(train_data$classe,p=0.75,list = FALSE)
train_subset <- train_data[training_partition,]
test_subset <- train_data[-training_partition,]

# pre-process the subset
low_var <- nearZeroVar(train_subset)
train_subset <- train_subset[,-low_var]
test_subset <- test_subset[,-low_var]

var_na <- sapply(train_subset,function(x){
                                mean(is.na(x))>0.95})
train_subset <- train_subset[,var_na==FALSE]
test_subset <- test_subset[,var_na==FALSE]

train_subset <- train_subset[ , -(1:5)]
test_subset  <- test_subset [ , -(1:5)]

dim(train_subset)
```

```
## [1] 14718    54
```

```
dim(test_subset)
```

```
## [1] 4904   54
```

Exploring Correlation

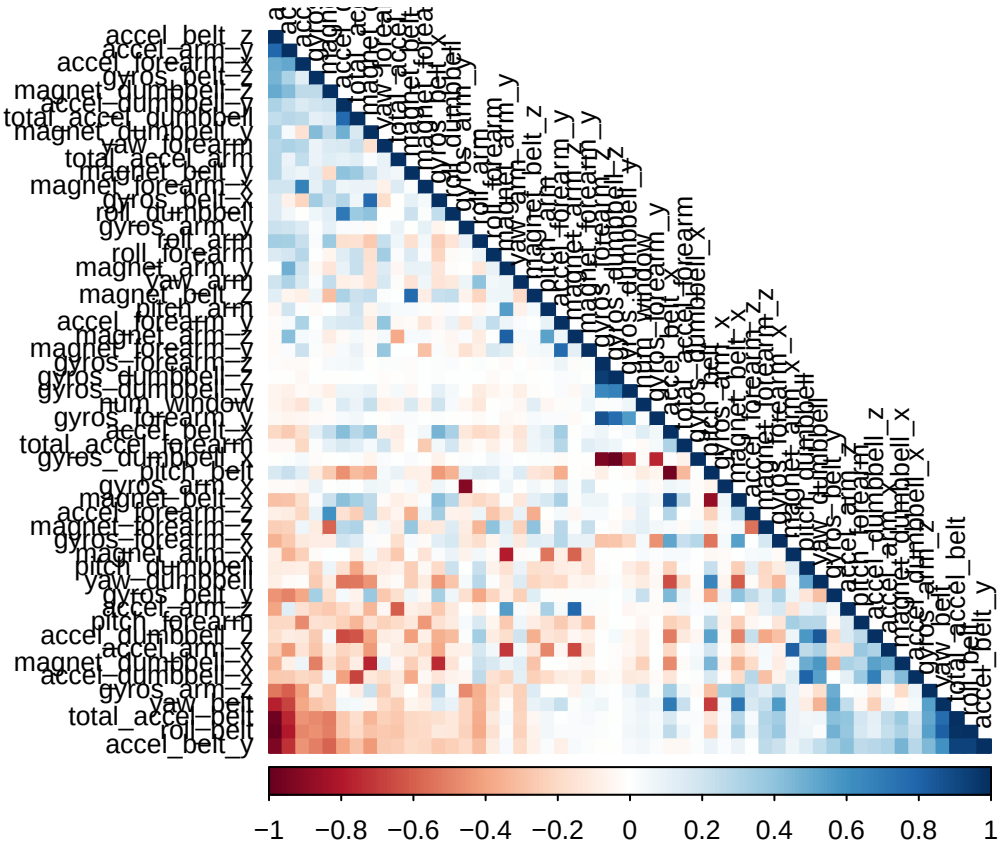
Here I go on with the correlation analysis between the variables before the model construction.

```
# Apply the correlation analysis for the dataset
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(lattice)
library(ggplot2)
library(rpart.plot)

correlation <- cor(train_subset[,-54])
corrplot(correlation, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



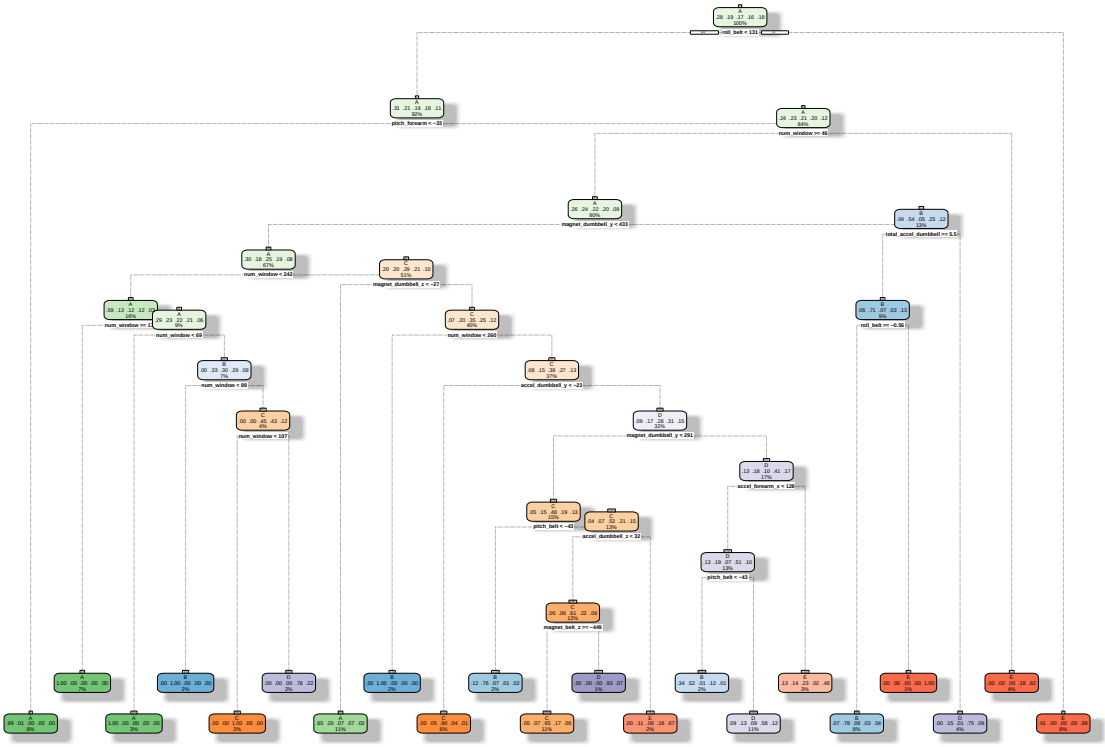
In the upper correlation analysis, all the involved variables are shown in the correlation plot. The blue colour represents the positive correlation efficients, while the red colour represents the negative efficients.

Prediction Model Construction

In this section, I apply three different approaches (Decision Tree Model, Generalized Boosted Model and Random Forest Model) for the modelling process on the training dataset. The one with higehr accuracy efficient will be used for the quiz section in the end of the project.

```
# Plot the decision tree
set.seed(1000)
decision_tree <- rpart(classe~.,data=train_subset,method="class")
fancyRpartPlot(decision_tree)
```

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



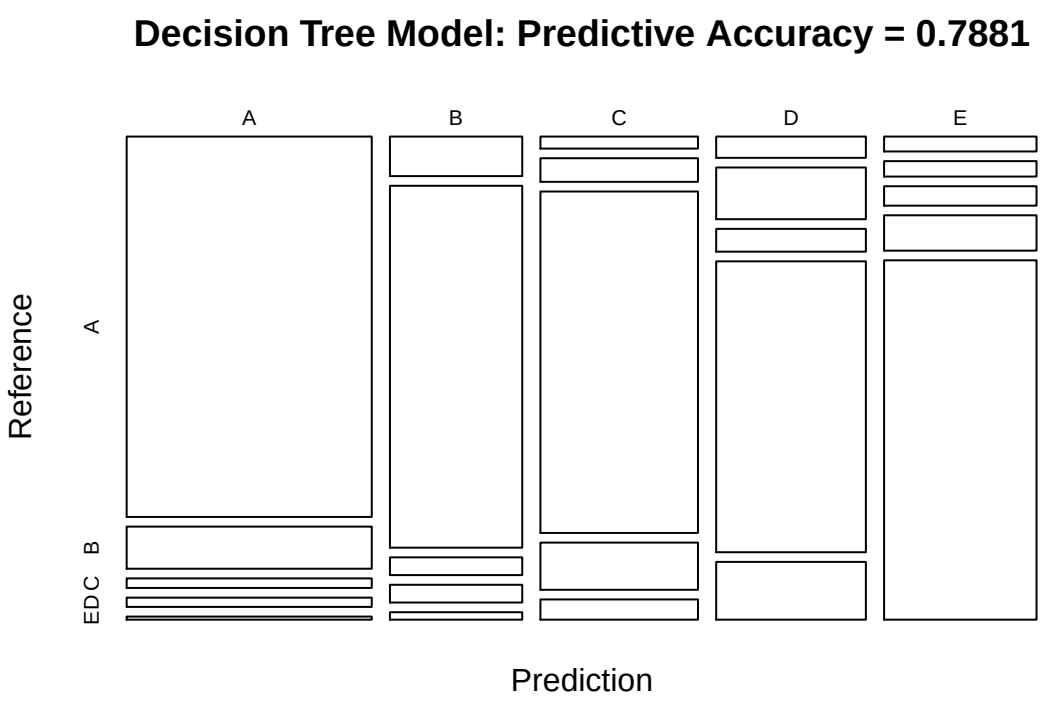
Rattle 2020−Jul−11 17:49:48 Lionpf

```
# Predictions from the decision tree model
prediction_decision_tree <- predict(decision_tree,newdata = test_subset,
                                    type = "class")
conf_decision_tree <- confusionMatrix(prediction_decision_tree,as.factor(test_subset$classe))
conf_decision_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1229  136   31   30   10
##          B   69  631   31   31   13
##          C   25   49  709   98   42
##          D   42  102   45  574  114
##          E   30   31   39   71  722
##
## Overall Statistics
##
##                Accuracy : 0.7881
##                  95% CI : (0.7764, 0.7995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.732
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8810   0.6649   0.8292   0.7139   0.8013
## Specificity            0.9410   0.9636   0.9471   0.9261   0.9573
## Pos Pred Value         0.8558   0.8142   0.7681   0.6545   0.8085
## Neg Pred Value         0.9521   0.9230   0.9633   0.9429   0.9554
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2506   0.1287   0.1446   0.1170   0.1472
## Detection Prevalence   0.2928   0.1580   0.1882   0.1788   0.1821
## Balanced Accuracy      0.9110   0.8143   0.8882   0.8200   0.8793
```

```
plot(conf_decision_tree$table, col = conf_decision_tree$byClass,
     main = paste("Decision Tree Model: Predictive Accuracy =",
                  round(conf_decision_tree$overall['Accuracy'], 4)))
```



Then the generalized boost model is as follows.

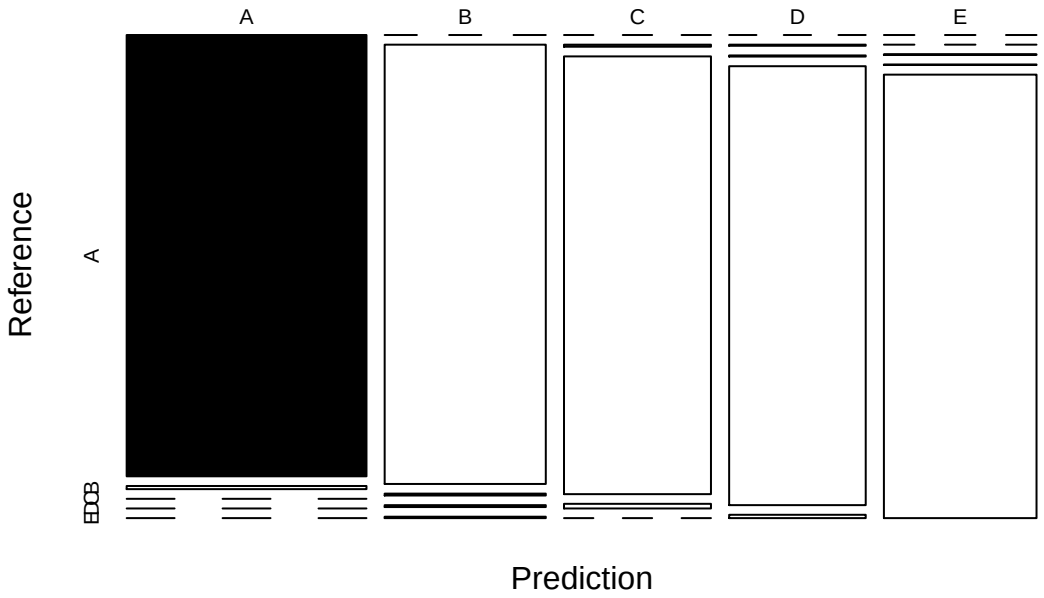
```
# Generalized Boosted Model
set.seed(1000)
control_GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
fit_GBM  <- train(classe ~ ., data = train_subset, method = "gbm",
                  trControl = control_GBM, verbose = FALSE)
fit_GBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```r
predict_GBM <- predict(fit_GBM, newdata=test_subset)
conf_GBM <- confusionMatrix(predict_GBM, as.factor(test_subset$classe))
conf_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395   10    0    0    0
##          B    0  933    4    4    3
##          C    0    4  848    9    0
##          D    0    2    2  790    6
##          E    0    0    1    1  892
##
## Overall Statistics
##
##                Accuracy : 0.9906
##                  95% CI : (0.9875, 0.9931)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9881
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9831   0.9918   0.9826   0.9900
## Specificity            0.9972   0.9972   0.9968   0.9976   0.9995
## Pos Pred Value         0.9929   0.9883   0.9849   0.9875   0.9978
## Neg Pred Value         1.0000   0.9960   0.9983   0.9966   0.9978
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1903   0.1729   0.1611   0.1819
## Detection Prevalence   0.2865   0.1925   0.1756   0.1631   0.1823
## Balanced Accuracy      0.9986   0.9902   0.9943   0.9901   0.9948
```

```r
plot(conf_GBM$table, col = conf_GBM$byClass,
     main = paste("GBM - Accuracy =", round(conf_GBM$overall['Accuracy'], 4)))
```



The random forest model comes in the following section.

```
# Random forest model
set.seed(1000)
control_RFM <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_RFM  <- train(classe ~ ., data = train_subset, method = "rf",
                  trControl = control_RFM, verbose = FALSE)
fit_RFM$finalModel
```

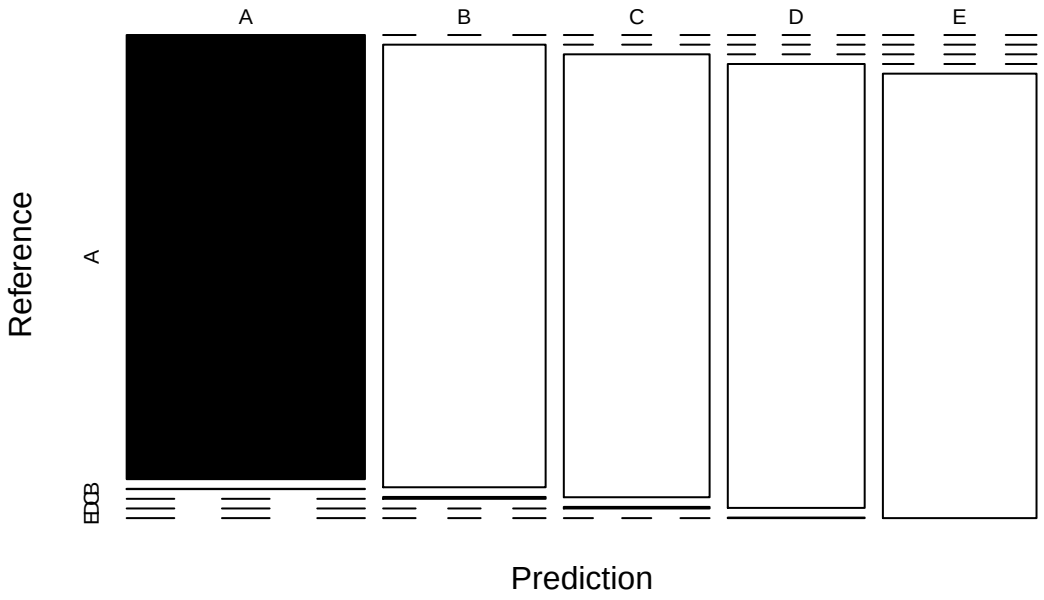
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4183    1    0    0    1 0.0004778973
## B    7 2836    4    1    0 0.0042134831
## C    0    3 2564    0    0 0.0011686794
## D    0    0    8 2403    1 0.0037313433
## E    0    1    0    2 2703 0.0011086475
```

```
predict_RFM <- predict(fit_RFM, newdata = test_subset)
conf_RFM <- confusionMatrix(predict_RFM, as.factor(test_subset$classe))
conf_RFM
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  948    4    0    0
##          C    0    0  851    3    0
##          D    0    0    0  801    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9982
##                  95% CI : (0.9965, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9977
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   0.9953   0.9963   0.9989
## Specificity            0.9997   0.9990   0.9993   0.9998   1.0000
## Pos Pred Value         0.9993   0.9958   0.9965   0.9988   1.0000
## Neg Pred Value         1.0000   0.9997   0.9990   0.9993   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1735   0.1633   0.1835
## Detection Prevalence   0.2847   0.1941   0.1741   0.1635   0.1835
## Balanced Accuracy      0.9999   0.9990   0.9973   0.9980   0.9994
```

```r
plot(conf_RFM$table, col = conf_RFM$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(conf_RFM$overall['Accuracy'], 4)))
```



Apply the selected data on the test.data

The upper analysis indicates the accuracy of three different model.

1. Decision Tree Model: 0.7471
2. Generalized Boosted Model: 0.9861

3. Random Forest Model: 0.9978

Here the random forest model is selected to be the model for test data.

```
predict_test <- predict(fit_RFM, newdata=test_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
```