Practical Machine Learning Project

Mousa Al Qaisi

February 04, 2020

```
#Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now
possible to collect a large amount of data about personal activity rela
tively inexpensively. These type of devices are part of the quantified
self movement - a group of enthusiasts who take measurements about them
selves regularly to improve their health, to find patterns in their beh
avior, or because they are tech geeks. One thing that people regularly
do is quantify how much of a particular activity they do, but they rare
ly quantify how well they do it.

#In this project, your goal will be to use data from accelerometers on
the belt, forearm, arm, and dumbell of 6 participants. They were asked
to perform barbell lifts correctly and incorrectly in 5 different ways.
More information is available from the website here: http://groupware.l
es.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise D
ataset).

# Library Loading

library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.2

library(ggplot2)
library(rattle)

## Warning: package 'rattle' was built under R version 3.5.3

## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.3

## 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

# Data Loading

TrainData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predma
chlearn/pml-training.csv"),header=TRUE)
dim(TrainData)

## [1] 19622   160

TestData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmac
hlearn/pml-testing.csv"),header=TRUE)
dim(TestData)

## [1]  20 160

str(TrainData)

## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",
..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1
323084232 1323084232 1323084232 1323084232 1323084232 1323084232 132308
4232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328
304277 368296 440390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",
..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1
1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42
1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09
8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.
4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..:
1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1
1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1
1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..
: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..
: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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      : Factor w/ 4 levels "","#DIV/0!","0.00",
..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 -0.0
2 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -2
0 -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 60
2 609 ...
##  $ 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 2
1.8 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 0.0
2 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 10
9 110 ...
##  $ 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 34
1 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 51
8 516 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ 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  : Factor w/ 398 levels "","-0.0035","-0.0
073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0
233",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1
1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0
096",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0
084",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1
1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..
: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..
: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]

#The training data set is made of 19622 observations on 160 columns. We
can notice that many columns have NA values or blank values on almost e
very observation. So we will remove them, because they will not produce
any information. The first seven columns give information about the peo
ple who did the test, and also timestamps. We will not take them in our
model.

indColToRemove <- which(colSums(is.na(TrainData) |TrainData=="")>0.9*di
m(TrainData)[1])
TrainDataClean <- TrainData[,-indColToRemove]
TrainDataClean <- TrainDataClean[,-c(1:7)]
dim(TrainDataClean)

## [1] 19622    53

str(TrainDataClean)

## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.4
2 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.1
3 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -9
4.4 -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 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 -2
1 ...
##  $ 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 60
9 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -31
3 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -12
8 -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 -16
1 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ 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 -28
9 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 11
0 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -12
4 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -37
2 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 33
4 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 51
6 ...
##  $ 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 ...
##  $ 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 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -23
4 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
```

```
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -27
2 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -55
5 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 29
1 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -6
9 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8
27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -6
3.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -15
2 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.0
2 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ..
.
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0
.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 19
0 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 20
5 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -21
3 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22
...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 65
6 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 47
3 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1
1 1 1 1 1 1 1 1 ...
```

```
indColToRemove <- which(colSums(is.na(TestData) |TestData=="")>0.9*dim(
TestData)[1])
TestDataClean <- TestData[,-indColToRemove]
TestDataClean <- TestDataClean[,-1]
dim(TestDataClean)
```

```
## [1] 20 59
```

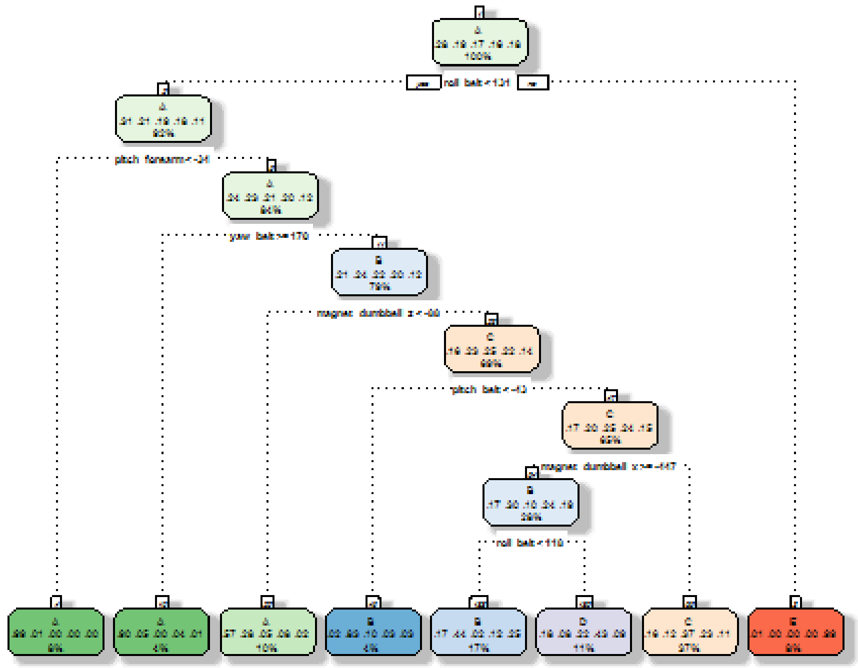
```
#After cleaning, the new training data set has only 53 columns.

str(TestDataClean)
```

```
## 'data.frame':    20 obs. of  59 variables:
##  $ user_name           : Factor w/ 6 levels "adelmo","carlitos",..:
6 5 5 1 4 5 5 5 2 3 ...
```

```
##  $ raw_timestamp_part_1: int  1323095002 1322673067 1322673075 13228
32789 1322489635 1322673149 1322673128 1322673076 1323084240 1322837822
...
##  $ raw_timestamp_part_2: int  868349 778725 342967 560311 814776 510
661 766645 54671 916313 384285 ...
##  $ cvtd_timestamp      : Factor w/ 11 levels "02/12/2011 13:33",..:
5 10 10 1 6 11 11 10 3 2 ...
##  $ new_window          : Factor w/ 1 level "no": 1 1 1 1 1 1 1 1 1 1
...
##  $ 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 ...
##  $ 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 -0.0
2 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 60
1 ...
##  $ magnet_belt_z       : int  -382 -309 -312 -304 -418 -291 -315 -30
5 -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 ...
##  $ 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 49
3 ...
##  $ 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 ...
##  $ total_accel_dumbbell: int  9 31 29 18 4 29 29 29 3 2 ...
##  $ gyros_dumbbell_x    : num  0.64 0.34 0.39 0.1 0.29 -0.59 0.34 0.3
7 0.03 0.42 ...
##  $ gyros_dumbbell_y    : num  0.06 0.05 0.14 -0.02 -0.47 0.8 0.16 0.
14 -0.21 0.51 ...
##  $ gyros_dumbbell_z    : num  -0.61 -0.71 -0.34 0.05 -0.46 1.1 -0.23
-0.39 -0.21 -0.03 ...
##  $ accel_dumbbell_x    : int  21 -153 -141 -51 -18 -138 -145 -140 0
-7 ...
##  $ accel_dumbbell_y    : int  -15 155 155 72 -30 166 150 159 25 -20
...
##  $ accel_dumbbell_z    : int  81 -205 -196 -148 -5 -186 -190 -191 9
7 ...
##  $ magnet_dumbbell_x   : int  523 -502 -506 -576 -424 -543 -484 -515
-519 -531 ...
##  $ magnet_dumbbell_y   : int  -528 388 349 238 252 262 354 350 348 3
21 ...
##  $ magnet_dumbbell_z   : int  -56 -36 41 53 312 96 97 53 -32 -164 ..
.
##  $ roll_forearm        : num  141 109 131 0 -176 150 155 -161 15.5 1
3.2 ...
##  $ pitch_forearm       : num  49.3 -17.6 -32.6 0 -2.16 1.46 34.5 43.
6 -63.5 19.4 ...
##  $ yaw_forearm         : num  156 106 93 0 -47.9 89.7 152 -89.5 -139
-105 ...
##  $ total_accel_forearm : int  33 39 34 43 24 43 32 47 36 24 ...
##  $ gyros_forearm_x     : num  0.74 1.12 0.18 1.38 -0.75 -0.88 -0.53
0.63 0.03 0.02 ...
##  $ gyros_forearm_y     : num  -3.34 -2.78 -0.79 0.69 3.1 4.26 1.8 -0
.74 0.02 0.13 ...
##  $ gyros_forearm_z     : num  -0.59 -0.18 0.28 1.8 0.8 1.35 0.75 0.4
9 -0.02 -0.07 ...
##  $ accel_forearm_x     : int  -110 212 154 -92 131 230 -192 -151 195
-212 ...
##  $ accel_forearm_y     : int  267 297 271 406 -93 322 170 -331 204 9
8 ...
##  $ accel_forearm_z     : int  -149 -118 -129 -39 172 -144 -175 -282
-217 -7 ...
##  $ magnet_forearm_x    : int  -714 -237 -51 -233 375 -300 -678 -109
0 -403 ...
```

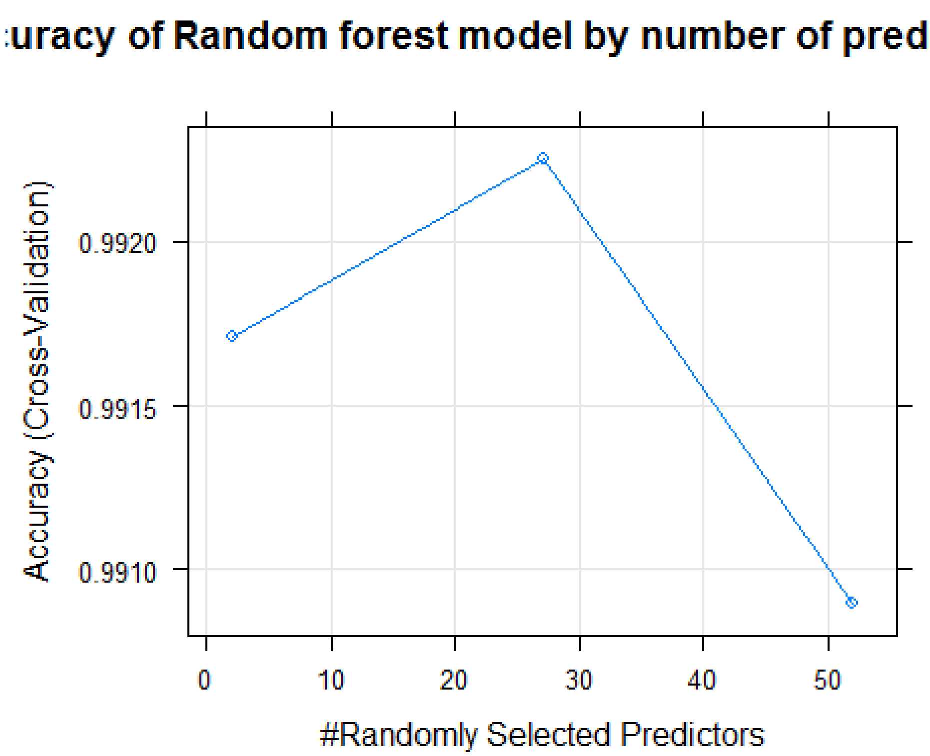
```
##  $ magnet_forearm_y    : int  419 791 698 783 -787 800 284 -619 652
723 ...
##  $ magnet_forearm_z    : int  617 873 783 521 91 884 585 -32 469 512
...
##  $ problem_id          : int  1 2 3 4 5 6 7 8 9 10 ...

set.seed(12345)
inTrain1 <- createDataPartition(TrainDataClean$classe, p=0.75, list=FAL
SE)
Train1 <- TrainDataClean[inTrain1,]
Test1 <- TrainDataClean[-inTrain1,]
dim(Train1)

## [1] 14718    53

dim(Test1)

## [1] 4904   53

#In the following sections, we will test 3 different models : * classif
ication tree * random forest * gradient boosting method
#In order to limit the effects of overfitting, and improve the efficice
ncy of the models, we will use the *cross-validation technique. We will
use 5 folds (usually, 5 or 10 can be used, but 10 folds gives higher ru
n times with no significant increase of the accuracy).

#Train with classification tree

trControl <- trainControl(method="cv", number=5)
model_CT <- train(classe~., data=Train1, method="rpart", trControl=trCo
ntrol)
fancyRpartPlot(model_CT$finalModel)
```



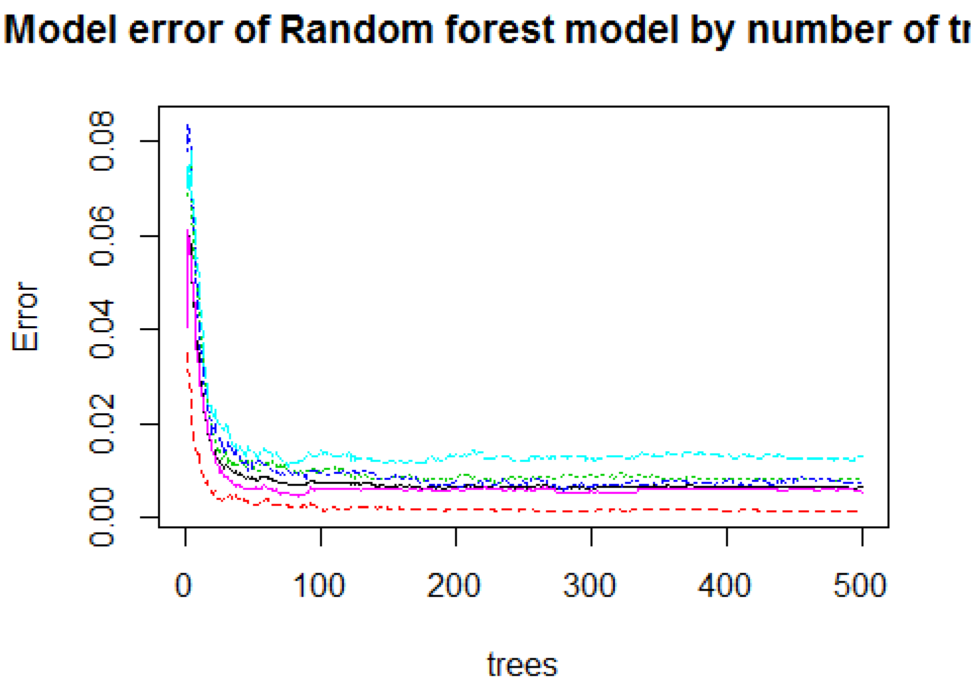
Rattle 2020-Feb-04 18:08:12 AL-Qaisi

```
trainpred <- predict(model_CT,newdata=Test1)
confMatCT <- confusionMatrix(Test1$classe,trainpred)
confMatCT$table

##             Reference
## Prediction   A   B   C   D   E
##          A 870 159 273  88   5
##          B 162 530 214  43   0
##          C  29  36 674 116   0
##          D  46 136 429 193   0
##          E  16 221 224  51 389

confMatCT$overall[1]

##  Accuracy
## 0.5415987

# Accuracy
# As we can notice that the accuracy of this first model is very low (a
bout 55%). This means that the outcome class will not be predicted very
well by the other predictors.
#Train with random forests
model_RF <- train(classe~., data=Train1, method="rf", trControl=trContr
ol, verbose=FALSE)
## Loading required package: randomForest

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

```
rfNews()
print(model_RF)

## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11776, 11775, 11773, 11774, 11774
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9917109  0.9895138
##   27    0.9922546  0.9902022
##   52    0.9908956  0.9884831
##
## Accuracy was used to select the optimal model using the largest valu
e.
## The final value used for the model was mtry = 27.

plot(model_RF,main="Accuracy of Random forest model by number of predic
tors")
```



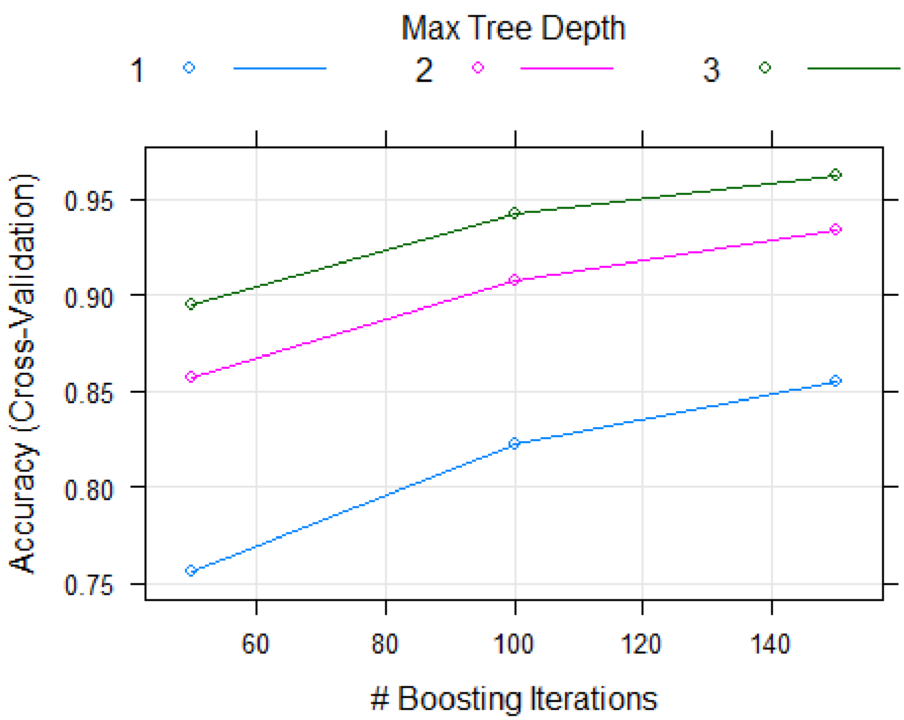
```r
trainpred <- predict(model_RF,newdata=Test1)

confMatRF <- confusionMatrix(Test1$classe,trainpred)

# display confusion matrix and model accuracy
confMatRF$table

##           Reference
## Prediction    A    B    C    D    E
##          A 1393    2    0    0    0
##          B    7  938    4    0    0
##          C    0    3  848    4    0
##          D    0    0   11  793    0
##          E    0    0    2    5  894

confMatRF$overall[1]

##  Accuracy 
## 0.9922512

names(model_RF$finalModel)

##  [1] "call"            "type"            "predicted"      
##  [4] "err.rate"        "confusion"       "votes"          
##  [7] "oob.times"       "classes"         "importance"     
## [10] "importanceSD"    "localImportance" "proximity"      
## [13] "ntree"           "mtry"            "forest"         
## [16] "y"               "test"            "inbag"          
## [19] "xNames"          "problemType"     "tuneValue"      
## [22] "obsLevels"       "param"

model_RF$finalModel$classes

## [1] "A" "B" "C" "D" "E"

plot(model_RF$finalModel,main="Model error of Random forest model by nu
mber of trees")
```




```
MostImpVars <- varImp(model_RF)
MostImpVars

## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                       Overall
## roll_belt             100.000
## pitch_forearm          58.550
## yaw_belt               56.063
## magnet_dumbbell_z      43.964
## pitch_belt             42.548
## magnet_dumbbell_y      39.531
## roll_forearm           37.888
## accel_dumbbell_y       21.385
## magnet_dumbbell_x      19.430
## roll_dumbbell          18.008
## accel_forearm_x        16.651
## magnet_belt_z          14.890
## accel_dumbbell_z       14.737
## accel_belt_z           12.774
## magnet_forearm_z       12.689
## total_accel_dumbbell   12.385
## magnet_belt_y          11.666
## yaw_arm                11.111
```

```
## gyros_belt_z          10.316
## magnet_belt_x          9.655

#With random forest, we reach an accuracy of 99.3% using cross-validati
on with 5 steps. This is very good
#Train with gradient boosting method
library(gbm)

## Warning: package 'gbm' was built under R version 3.5.3

## Loaded gbm 2.1.5

model_GBM <- train(classe~., data=Train1, method="gbm", trControl=trCon
trol, verbose=FALSE)
library(survival)

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

library(splines)
library(parallel)
print(model_GBM)

## Stochastic Gradient Boosting
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11773, 11774, 11775, 11776
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7561493  0.6909021
##   1                  100      0.8221896  0.7749601
##   1                  150      0.8550741  0.8166499
##   2                   50      0.8571806  0.8190870
##   2                  100      0.9080025  0.8835618
##   2                  150      0.9334817  0.9158273
##   3                   50      0.8954343  0.8676156
##   3                  100      0.9421795  0.9268312
##   3                  150      0.9625632  0.9526340
##
## 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 valu
e.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.

plot(model_GBM)
```




```
trainpred <- predict(model_GBM,newdata=Test1)

confMatGBM <- confusionMatrix(Test1$classe,trainpred)
confMatGBM$table

##             Reference
## Prediction    A    B    C    D    E
##          A 1377   16    2    0    0
##          B   36  877   33    3    0
##          C    0   21  820   13    1
##          D    0    1   30  768    5
##          E    2   10   16   13  860

confMatGBM$overall[1]

##  Accuracy
## 0.9588091

# The Conclusion
#This shows that the random forest model is the best one. We will then
use it to predict the values of classe for the test data set.
```

```r
FinalTestPred <- predict(model_RF,newdata=TestDataClean)
FinalTestPred

##  [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
```