Practical Machine Learning Course Project

Aayush Shah

Introduction/Overview

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

The data consists of training dataset and testing dataset (to be used to validate the selected model).

The goal of this project is to predict the manner in which they did the exercise. This is the *classe* variable in the training set.

But first, we load all the libraries.

```
library(caret)
```

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

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

```
library(ggplot2)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 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
```

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

Exploratory Data Analysis

We get the data.

```
filename1<-"pml-testing.csv"
filename2<-"pml-training.csv"
if (!file.exists(filename1)){
  url1<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file(url1, filename1, method="curl")
}
if (!file.exists(filename2)){
  url2<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(url2, filename2, method="curl")
}
```

Then we load and read the data.

```
trainInput<-read.csv("pml-training.csv")
validationInput<-read.csv("pml-testing.csv")
```

Now we explore our data.

```
dim(trainInput)
```

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

```
dim(validationInput)
```

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

```
str(trainInput)
```

```
## '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 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ 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 -0.02 -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       : 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","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 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","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 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]
```

So there 19622 observations of 160 variables in the *trainInput* dataset.

But on observation we find that there are lots of variables with missing values. So we remove them before proceeding.

```
training<- trainInput[,colSums(is.na(trainInput)) == 0]
validation <- validationInput[,colSums(is.na(validationInput)) == 0]
str(training)
```

```
## 'data.frame':    19622 obs. of  93 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 1323084232 1323084232 1323084232 13
##  $ raw_timestamp_part_2 : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4844
```

```
##  $ 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_yaw_belt           : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_yaw_belt           : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_yaw_belt     : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 ...
##  $ 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      : 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 ...
##  $ 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","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell: Factor w/ 401 levels "","-0.0163","-0.0233",..: 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","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell: Factor w/ 402 levels "","-0.0053","-0.0084",..: 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_yaw_dumbbell       : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ min_yaw_dumbbell       : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_yaw_dumbbell : Factor w/ 3 levels "","#DIV/0!","0.00": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 -234 -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 -272 -269 -270 ...
##  $ magnet_dumbbell_x      : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y      : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z      : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ 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 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm            : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ kurtosis_roll_forearm  : Factor w/ 322 levels "","-0.0227","-0.0359",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_forearm : Factor w/ 323 levels "","-0.0073","-0.0442",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_forearm   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_forearm  : Factor w/ 323 levels "","-0.0004","-0.0013",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_forearm : Factor w/ 319 levels "","-0.0113","-0.0131",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_forearm   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_yaw_forearm        : Factor w/ 45 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_yaw_forearm        : Factor w/ 45 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_yaw_forearm  : Factor w/ 3 levels "","#DIV/0!","0.00": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 0.02 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  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y        : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z        : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x       : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y       : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z       : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe                 : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
str(validation)
```

```
## 'data.frame':    20 obs. of  60 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ 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 1322832789 1322489635 1322673149 1322
##  $ raw_timestamp_part_2: int  868349 778725 342967 560311 814776 510661 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 -0.02 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 ...
##  $ 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 ...
##  $ 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 0.37 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  -528 388 349 238 252 262 354 350 348 321 ...
##  $ 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 13.2 ...
##  $ 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 0.49 -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 98 ...
##  $ 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 ...
```

So now the the *trainInput* dataset is reduced to *training* dataset with 19622 observations of 93 variables.

The first seven variables have little impact on *classe* variable. So we remove them too.

```
training<-training[,-c(1:7)]
validation<-validation[,-c(1:7)]
```

Predictive Analysis

Lets split the training dataset into 70% *trainData* and 30% *testData* datasets.(The *validation* dataset will be used later to test the prodiction algorithm on the 20 cases.)

```
set.seed(1234)
inTrain<-createDataPartition(y=training$classe,p=0.7,list=FALSE)
trainData<-training[inTrain,]
testData<-training[-inTrain,]
```

On Observation, we see that lot of variable have near zero variance. So we remove them.

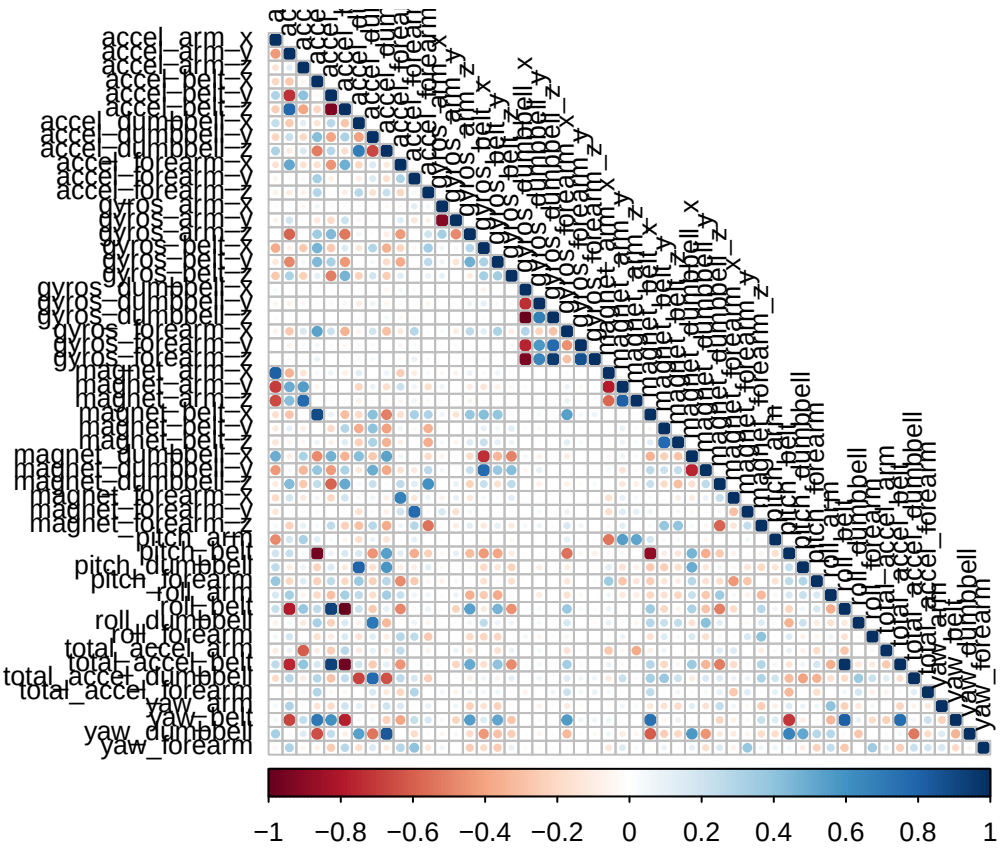
```
NZV <- nearZeroVar(trainData)
trainData <- trainData[, -NZV]
testData  <- testData[, -NZV]
dim(trainData)
```

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

Lets keep the *testData* aside for some time.

Now the question is to find the correlation in the remaining variables except the *classe* variable. Lets do that visually.

```
cor_matrix <- cor(trainData[, -53])
corrplot(cor_matrix, order = "alphabet", method = "circle", type = "lower",tl.cex = 0.8, tl.col = rgb(0
```



As we see, highly correlated variables have a darker colour intersection and bigger circles.

To get the names of these variables having correlation higher than 0.75 we do this;

```
high_corr = findCorrelation(cor_matrix, cutoff=0.75)
names(trainData)[high_corr]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "total_accel_belt"  "accel_dumbbell_z"  "accel_belt_x"
##  [7] "pitch_belt"        "magnet_dumbbell_x" "accel_dumbbell_y"
## [10] "magnet_dumbbell_y" "accel_arm_x"       "accel_dumbbell_x"
## [13] "accel_arm_z"       "magnet_arm_y"      "magnet_belt_z"
## [16] "accel_forearm_y"   "gyros_forearm_y"   "gyros_dumbbell_x"
## [19] "gyros_dumbbell_z"  "gyros_arm_x"
```

Now lets build our models.

We will use the following methods to predict the ***classe*** variable;

1] Classification Trees

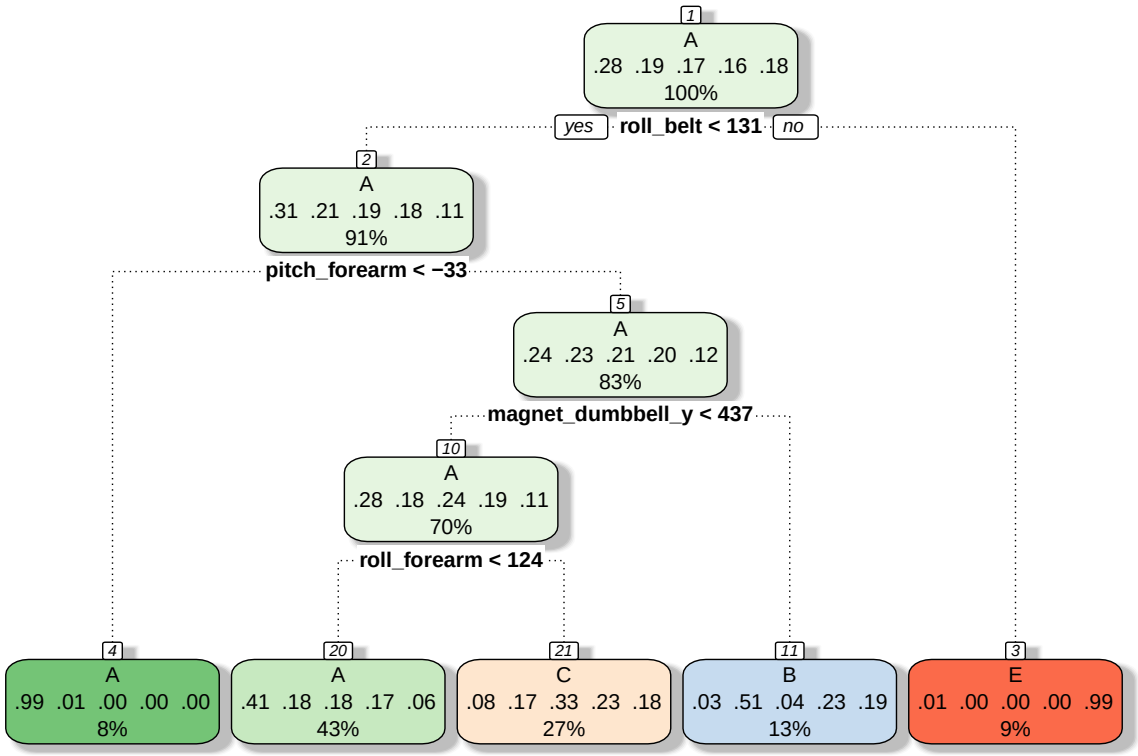
2] Random Forests

3] Generalized Boosted Models

In order to limit the effects of overfitting, and improve the efficicency of the models, we will use the **cross validation** technique. We will use 5 folds.

Classification Trees

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



Now we predict using the testData set.

```
pred<-predict(model_CT,newdata=testData)
conf_mat<-confusionMatrix(testData$classe,pred)
conf_mat$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1530   35  105    0    4
##          B  486  379  274    0    0
##          C  493   31  502    0    0
##          D  452  164  348    0    0
##          E  168  145  302    0  467
```

```
conf_mat$overall[1]
```

```
##  Accuracy
## 0.4890399
```

We see that the accuracy is 0.4890399, which is very low. So *classe* is not well predicted by other variables in this model.

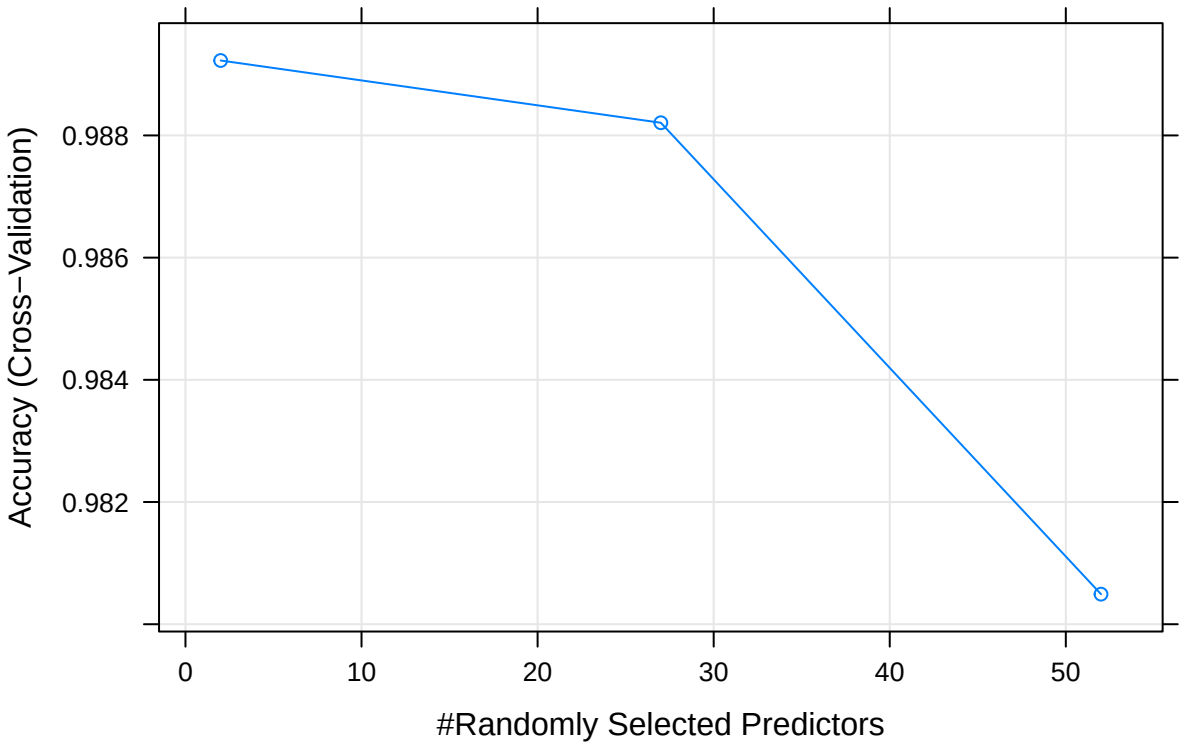
Random Forests

```r
set.seed(1234)
trControl <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRF1 <- train(classe ~ ., data=trainData, method="rf", trControl=trControl)
```

```r
print(modRF1)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9160, 9157, 9157
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9892261  0.9863694
##   27    0.9882075  0.9850808
##   52    0.9804919  0.9753144
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
plot(modRF1,main="Accuracy of Random forest model by number of predictors")
```



Now we predict using the testData set.

```
pred2<-predict(modRF1,newdata=testData)
conf_mat2<-confusionMatrix(testData$classe,pred2)
conf_mat2$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    8 1130    1    0    0
##          C    0    7 1018    1    0
##          D    0    0   11  952    1
##          E    0    0    1    2 1079
```

```
conf_mat2$overall[1]
```

```
##  Accuracy
## 0.9945624
```

We see that the accuracy is 0.9945624, which is very high. So *classe* is very well predicted by other variables in this model.

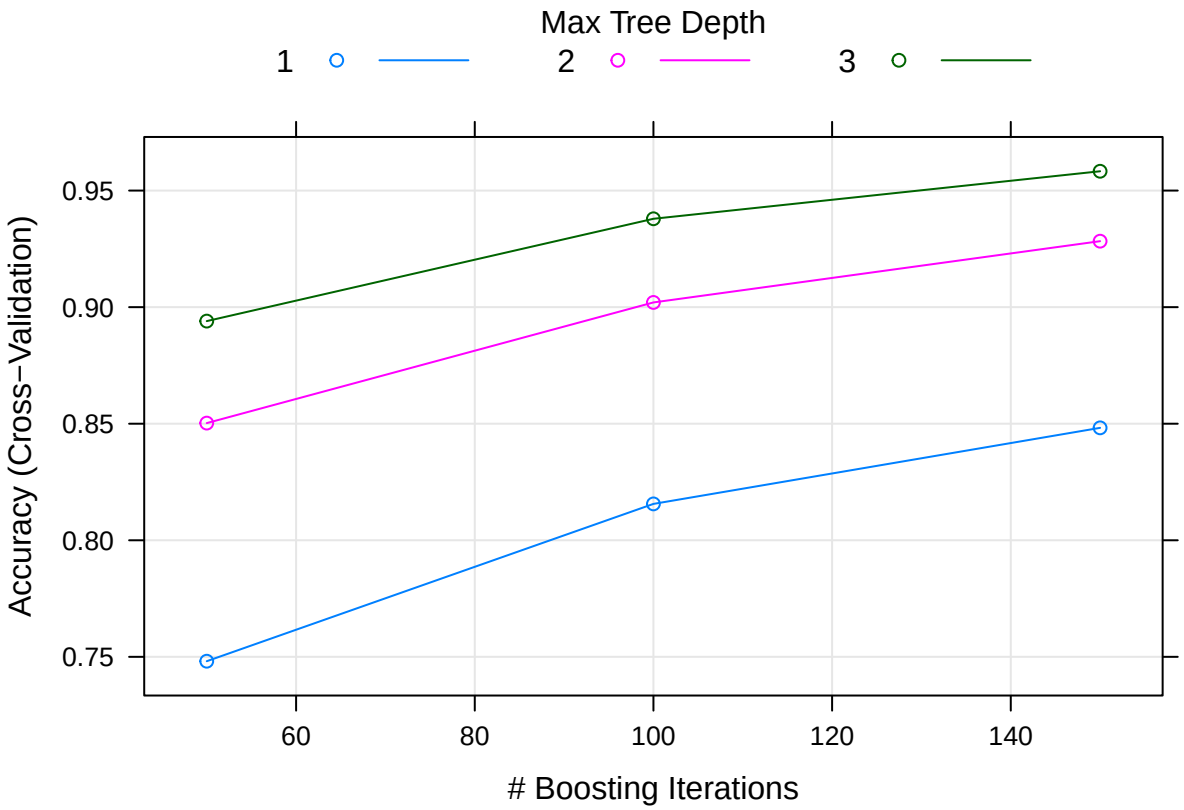
This is very good. But let's see what we can expect with Gradient boosting.

Gradient Boosting Method

```
model_GBM <- train(classe~., data=trainData, method="gbm", trControl=trControl, verbose=FALSE)
print(model_GBM)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9160, 9157, 9157
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7481246  0.6806916
##   1                  100      0.8156080  0.7667170
##   1                  150      0.8482213  0.8079820
##   2                   50      0.8502586  0.8102841
##   2                  100      0.9020166  0.8759677
##   2                  150      0.9282955  0.9092562
##   3                   50      0.8940083  0.8658143
##   3                  100      0.9379048  0.9214218
##   3                  150      0.9582879  0.9472293
##
## 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.
```

```
plot(model_GBM)
```



Now we predict using testData

```
pred3<-predict(model_GBM,newdata=testData)
conf_mat3<-confusionMatrix(testData$classe,pred3)
conf_mat3$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1651   14    1    6    2
##          B   47 1067   23    2    0
##          C    0   29  983   11    3
##          D    0    6   18  931    9
##          E    1    4   15   16 1046
```

```
conf_mat3$overall[1]
```

```
##  Accuracy
## 0.9648258
```

So the accuracy is 0.9648258

The accuracy rate using the random forest is very high *out-of-sample-error is equal to 0.0264*.

Predicting the validation data with the best model

On comparison, the accuracy of the Random Forest model is the highest. So will use it on the *validation* dataset.

```
result<-predict(modRF1,newdata=validation)
result
```

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

This *result* will be used to answer the quiz for the course.