Practical Machine Learning Project

Marijus B

Libraries

```
library(caret)
```

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

```
## Loading required package: 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(rpart)
library(rpart.plot)
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
```

```
library(data.table)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

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

```
library(ggplot2)
library(MASS)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.3
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x kernlab::alpha()       masks ggplot2::alpha()
## x dplyr::between()       masks data.table::between()
## x dplyr::combine()       masks randomForest::combine()
## x purrr::cross()         masks kernlab::cross()
## x dplyr::filter()        masks stats::filter()
## x dplyr::first()         masks data.table::first()
## x dplyr::lag()           masks stats::lag()
## x dplyr::last()          masks data.table::last()
## x purrr::lift()          masks caret::lift()
## x randomForest::margin() masks ggplot2::margin()
## x dplyr::select()        masks MASS::select()
## x purrr::transpose()     masks data.table::transpose()
```

```
library(data.table)
```

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, 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.

Project

The main goal of the project is predict the manner in which exercises were performed by six participants and it is labeled as "classe" in the training data set.

Data for the analysis consists of training and test data sets.

Data will be tested with 4 different models : *Classification Tree* , *Random Forest* , *Gradient Boosting Method* and *Linear Discriminant Analysis*.

WLE dataset for this project is a courtesy of: 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.

Getting Data

Data is obtained from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Training Data

```
trainingData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),hea
table(trainingData$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
#str(trainingData)
dim(trainingData)
```

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

```
#summary(trainingData)
```

Testing Data

```
testingData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),heade
table(testingData$classe)
```

```
## < table of extent 0 >
```

```
#str(testingData)
dim(testingData)
```

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

```
#summary(testingData)
```

Cleaning Data Sets

Removing variables from the data sets that are close to zero:

```
NSV <- nearZeroVar(trainingData,saveMetrics=TRUE)
NSV
```

```
##                          freqRatio percentUnique zeroVar   nzv
## X                         1.000000  100.00000000   FALSE FALSE
## user_name                 1.100679    0.03057792   FALSE FALSE
## raw_timestamp_part_1      1.000000    4.26562022   FALSE FALSE
## raw_timestamp_part_2      1.000000   85.53154622   FALSE FALSE
## cvtd_timestamp            1.000668    0.10192641   FALSE FALSE
## new_window               47.330049    0.01019264   FALSE  TRUE
## num_window                1.000000    4.37264295   FALSE FALSE
## roll_belt                 1.101904    6.77810621   FALSE FALSE
## pitch_belt                1.036082    9.37722964   FALSE FALSE
## yaw_belt                  1.058480    9.97349913   FALSE FALSE
## total_accel_belt          1.063160    0.14779329   FALSE FALSE
## kurtosis_roll_belt     1921.600000    2.02323922   FALSE  TRUE
## kurtosis_picth_belt     600.500000    1.61553358   FALSE  TRUE
## kurtosis_yaw_belt        47.330049    0.01019264   FALSE  TRUE
## skewness_roll_belt     2135.111111    2.01304658   FALSE  TRUE
```

```
## skewness_roll_belt.1      600.500000    1.72255631   FALSE  TRUE
## skewness_yaw_belt          47.330049    0.01019264   FALSE  TRUE
## max_roll_belt               1.000000    0.99378249   FALSE FALSE
## max_picth_belt              1.538462    0.11211905   FALSE FALSE
## max_yaw_belt              640.533333    0.34654979   FALSE  TRUE
## min_roll_belt               1.000000    0.93772296   FALSE FALSE
## min_pitch_belt              2.192308    0.08154113   FALSE FALSE
## min_yaw_belt              640.533333    0.34654979   FALSE  TRUE
## amplitude_roll_belt         1.290323    0.75425543   FALSE FALSE
## amplitude_pitch_belt        3.042254    0.06625217   FALSE FALSE
## amplitude_yaw_belt         50.041667    0.02038528   FALSE  TRUE
## var_total_accel_belt        1.426829    0.33126083   FALSE FALSE
## avg_roll_belt               1.066667    0.97339721   FALSE FALSE
## stddev_roll_belt            1.039216    0.35164611   FALSE FALSE
## var_roll_belt               1.615385    0.48924676   FALSE FALSE
## avg_pitch_belt              1.375000    1.09061258   FALSE FALSE
## stddev_pitch_belt           1.161290    0.21914178   FALSE FALSE
## var_pitch_belt              1.307692    0.32106819   FALSE FALSE
## avg_yaw_belt                1.200000    1.22311691   FALSE FALSE
## stddev_yaw_belt             1.693878    0.29558659   FALSE FALSE
## var_yaw_belt                1.500000    0.73896647   FALSE FALSE
## gyros_belt_x                1.058651    0.71348486   FALSE FALSE
## gyros_belt_y                1.144000    0.35164611   FALSE FALSE
## gyros_belt_z                1.066214    0.86127816   FALSE FALSE
## accel_belt_x                1.055412    0.83579655   FALSE FALSE
## accel_belt_y                1.113725    0.72877383   FALSE FALSE
## accel_belt_z                1.078767    1.52379982   FALSE FALSE
## magnet_belt_x               1.090141    1.66649679   FALSE FALSE
## magnet_belt_y               1.099688    1.51870350   FALSE FALSE
## magnet_belt_z               1.006369    2.32901845   FALSE FALSE
## roll_arm                   52.338462   13.52563449   FALSE FALSE
## pitch_arm                  87.256410   15.73234125   FALSE FALSE
## yaw_arm                    33.029126   14.65701763   FALSE FALSE
## total_accel_arm             1.024526    0.33635715   FALSE FALSE
## var_accel_arm               5.500000    2.01304658   FALSE FALSE
## avg_roll_arm               77.000000    1.68178575   FALSE  TRUE
## stddev_roll_arm            77.000000    1.68178575   FALSE  TRUE
## var_roll_arm               77.000000    1.68178575   FALSE  TRUE
## avg_pitch_arm              77.000000    1.68178575   FALSE  TRUE
## stddev_pitch_arm           77.000000    1.68178575   FALSE  TRUE
## var_pitch_arm              77.000000    1.68178575   FALSE  TRUE
## avg_yaw_arm                77.000000    1.68178575   FALSE  TRUE
## stddev_yaw_arm             80.000000    1.66649679   FALSE  TRUE
## var_yaw_arm                80.000000    1.66649679   FALSE  TRUE
## gyros_arm_x                 1.015504    3.27693405   FALSE FALSE
## gyros_arm_y                 1.454369    1.91621649   FALSE FALSE
## gyros_arm_z                 1.110687    1.26388747   FALSE FALSE
## accel_arm_x                 1.017341    3.95984099   FALSE FALSE
## accel_arm_y                 1.140187    2.73672409   FALSE FALSE
## accel_arm_z                 1.128000    4.03628580   FALSE FALSE
## magnet_arm_x                1.000000    6.82397309   FALSE FALSE
## magnet_arm_y                1.056818    4.44399144   FALSE FALSE
## magnet_arm_z                1.036364    6.44684538   FALSE FALSE
## kurtosis_roll_arm         246.358974    1.68178575   FALSE  TRUE
```

```
## kurtosis_picth_arm        240.200000    1.67159311   FALSE  TRUE
## kurtosis_yaw_arm         1746.909091    2.01304658   FALSE  TRUE
## skewness_roll_arm         249.558442    1.68688207   FALSE  TRUE
## skewness_pitch_arm        240.200000    1.67159311   FALSE  TRUE
## skewness_yaw_arm         1746.909091    2.01304658   FALSE  TRUE
## max_roll_arm               25.666667    1.47793293   FALSE  TRUE
## max_picth_arm              12.833333    1.34033228   FALSE FALSE
## max_yaw_arm                 1.227273    0.25991234   FALSE FALSE
## min_roll_arm               19.250000    1.41677709   FALSE  TRUE
## min_pitch_arm              19.250000    1.47793293   FALSE  TRUE
## min_yaw_arm                 1.000000    0.19366018   FALSE FALSE
## amplitude_roll_arm         25.666667    1.55947406   FALSE  TRUE
## amplitude_pitch_arm        20.000000    1.49831821   FALSE  TRUE
## amplitude_yaw_arm           1.037037    0.25991234   FALSE FALSE
## roll_dumbbell               1.022388   84.20650290   FALSE FALSE
## pitch_dumbbell              2.277372   81.74498012   FALSE FALSE
## yaw_dumbbell                1.132231   83.48282540   FALSE FALSE
## kurtosis_roll_dumbbell   3843.200000    2.02833554   FALSE  TRUE
## kurtosis_picth_dumbbell  9608.000000    2.04362450   FALSE  TRUE
## kurtosis_yaw_dumbbell      47.330049    0.01019264   FALSE  TRUE
## skewness_roll_dumbbell   4804.000000    2.04362450   FALSE  TRUE
## skewness_pitch_dumbbell  9608.000000    2.04872082   FALSE  TRUE
## skewness_yaw_dumbbell      47.330049    0.01019264   FALSE  TRUE
## max_roll_dumbbell           1.000000    1.72255631   FALSE FALSE
## max_picth_dumbbell          1.333333    1.72765263   FALSE FALSE
## max_yaw_dumbbell          960.800000    0.37203139   FALSE  TRUE
## min_roll_dumbbell           1.000000    1.69197839   FALSE FALSE
## min_pitch_dumbbell          1.666667    1.81429008   FALSE FALSE
## min_yaw_dumbbell          960.800000    0.37203139   FALSE  TRUE
## amplitude_roll_dumbbell     8.000000    1.97227602   FALSE FALSE
## amplitude_pitch_dumbbell    8.000000    1.95189073   FALSE FALSE
## amplitude_yaw_dumbbell     47.920200    0.01528896   FALSE  TRUE
## total_accel_dumbbell        1.072634    0.21914178   FALSE FALSE
## var_accel_dumbbell          6.000000    1.95698706   FALSE FALSE
## avg_roll_dumbbell           1.000000    2.02323922   FALSE FALSE
## stddev_roll_dumbbell       16.000000    1.99266130   FALSE FALSE
## var_roll_dumbbell          16.000000    1.99266130   FALSE FALSE
## avg_pitch_dumbbell          1.000000    2.02323922   FALSE FALSE
## stddev_pitch_dumbbell      16.000000    1.99266130   FALSE FALSE
## var_pitch_dumbbell         16.000000    1.99266130   FALSE FALSE
## avg_yaw_dumbbell            1.000000    2.02323922   FALSE FALSE
## stddev_yaw_dumbbell        16.000000    1.99266130   FALSE FALSE
## var_yaw_dumbbell           16.000000    1.99266130   FALSE FALSE
## gyros_dumbbell_x            1.003268    1.22821323   FALSE FALSE
## gyros_dumbbell_y            1.264957    1.41677709   FALSE FALSE
## gyros_dumbbell_z            1.060100    1.04984201   FALSE FALSE
## accel_dumbbell_x            1.018018    2.16593619   FALSE FALSE
## accel_dumbbell_y            1.053061    2.37488533   FALSE FALSE
## accel_dumbbell_z            1.133333    2.08949139   FALSE FALSE
## magnet_dumbbell_x           1.098266    5.74864948   FALSE FALSE
## magnet_dumbbell_y           1.197740    4.30129447   FALSE FALSE
## magnet_dumbbell_z           1.020833    3.44511263   FALSE FALSE
## roll_forearm               11.589286   11.08959331   FALSE FALSE
## pitch_forearm              65.983051   14.85577413   FALSE FALSE
```

```
## yaw_forearm               15.322835   10.14677403   FALSE FALSE
## kurtosis_roll_forearm    228.761905    1.64101519   FALSE  TRUE
## kurtosis_picth_forearm   226.070588    1.64611151   FALSE  TRUE
## kurtosis_yaw_forearm      47.330049    0.01019264   FALSE  TRUE
## skewness_roll_forearm    231.518072    1.64611151   FALSE  TRUE
## skewness_pitch_forearm   226.070588    1.62572623   FALSE  TRUE
## skewness_yaw_forearm      47.330049    0.01019264   FALSE  TRUE
## max_roll_forearm          27.666667    1.38110284   FALSE  TRUE
## max_picth_forearm          2.964286    0.78992967   FALSE FALSE
## max_yaw_forearm          228.761905    0.22933442   FALSE  TRUE
## min_roll_forearm          27.666667    1.37091020   FALSE  TRUE
## min_pitch_forearm          2.862069    0.87147080   FALSE FALSE
## min_yaw_forearm          228.761905    0.22933442   FALSE  TRUE
## amplitude_roll_forearm    20.750000    1.49322189   FALSE  TRUE
## amplitude_pitch_forearm    3.269231    0.93262664   FALSE FALSE
## amplitude_yaw_forearm     59.677019    0.01528896   FALSE  TRUE
## total_accel_forearm        1.128928    0.35674243   FALSE FALSE
## var_accel_forearm          3.500000    2.03343186   FALSE FALSE
## avg_roll_forearm          27.666667    1.64101519   FALSE  TRUE
## stddev_roll_forearm       87.000000    1.63082255   FALSE  TRUE
## var_roll_forearm          87.000000    1.63082255   FALSE  TRUE
## avg_pitch_forearm         83.000000    1.65120783   FALSE  TRUE
## stddev_pitch_forearm      41.500000    1.64611151   FALSE  TRUE
## var_pitch_forearm         83.000000    1.65120783   FALSE  TRUE
## avg_yaw_forearm           83.000000    1.65120783   FALSE  TRUE
## stddev_yaw_forearm        85.000000    1.64101519   FALSE  TRUE
## var_yaw_forearm           85.000000    1.64101519   FALSE  TRUE
## gyros_forearm_x            1.059273    1.51870350   FALSE FALSE
## gyros_forearm_y            1.036554    3.77637346   FALSE FALSE
## gyros_forearm_z            1.122917    1.56457038   FALSE FALSE
## accel_forearm_x            1.126437    4.04647844   FALSE FALSE
## accel_forearm_y            1.059406    5.11160942   FALSE FALSE
## accel_forearm_z            1.006250    2.95586586   FALSE FALSE
## magnet_forearm_x           1.012346    7.76679238   FALSE FALSE
## magnet_forearm_y           1.246914    9.54031189   FALSE FALSE
## magnet_forearm_z           1.000000    8.57710733   FALSE FALSE
## classe                     1.469581    0.02548160   FALSE FALSE
```

```
nsv <- nearZeroVar(trainingData)
nsv
```

```
##  [1]   6  12  13  14  15  16  17  20  23  26  51  52  53  54  55  56  57  58  59
## [20]  69  70  71  72  73  74  75  78  79  81  82  87  88  89  90  91  92  95  98
## [39] 101 125 126 127 128 129 130 131 133 134 136 137 139 142 143 144 145 146 147
## [58] 148 149 150
```

```
training <- trainingData[, -nsv]
testing  <- testingData[, -nsv]
dim(training)
```

```
## [1] 19622   100
```

```
dim(testing)
```

```
## [1]  20 100
```

Removing missing values, NAs from data sets:

```
indColToRemove <- which(colSums(is.na(trainingData) |trainingData=="")>0.8*dim(trainingData)[1])
training <- trainingData[,-indColToRemove]
str(training)
```

```
## 'data.frame':    19622 obs. of  60 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 1323
##  $ 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 ...
##  $ 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 ...
##  $ 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 -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 ...
##  $ 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 ...
```

```
dim(training)
```

```
## [1] 19622    60
```

```
indColToRemove <- which(colSums(is.na(testingData) |testingData=="")>0.8*dim(testingData)[1])
testing <- testingData[,-indColToRemove]
str(testing)
```

```
## '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 ...
```

```
dim(testing)
```

```
## [1] 20 60
```

Removing variables from data sets with no significance:

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

```
dim(training)
```

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

```
dim(testing)
```

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

After removing variables that has no significant role in modeling and NAs split training data set (partition) into 75% and 25% as Training and Testing data sets respectivelly:

```
set.seed(1234)
inTrain <- createDataPartition(y=training$classe,p=0.75, list=FALSE)
Training <- training[inTrain,]
Testing <- training[-inTrain,]
```

New dimensions of data sets:

```
dim(Training)
```

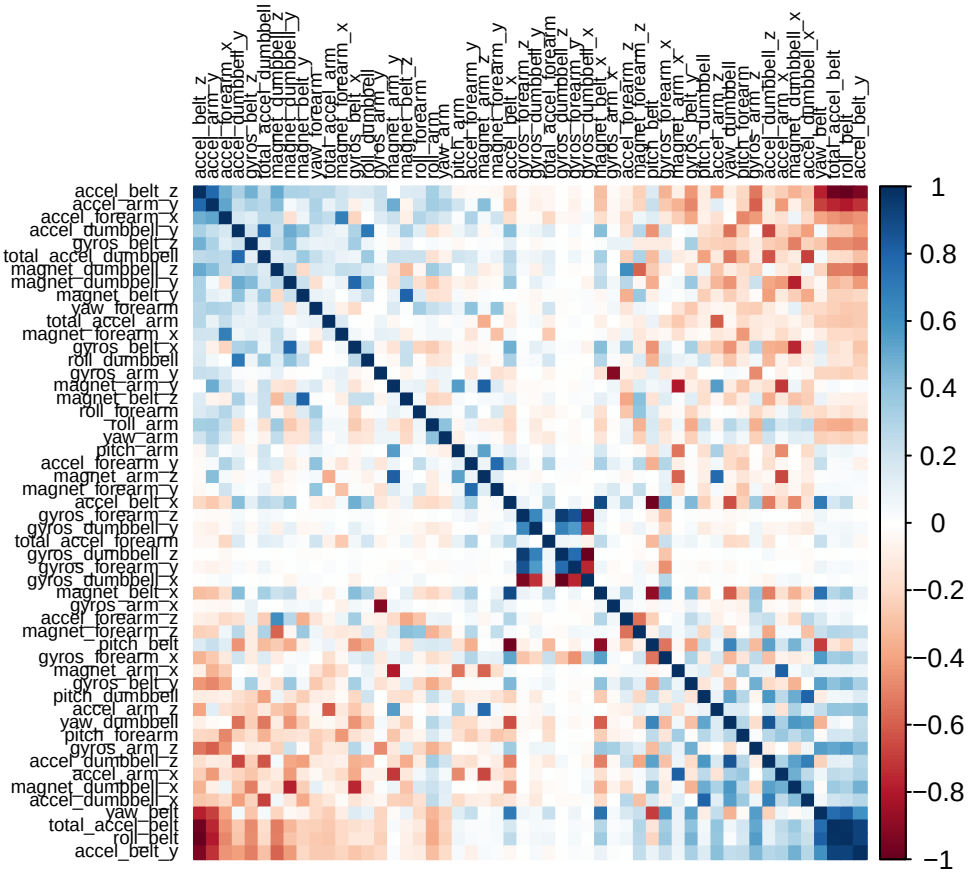
```
## [1] 14718    53
```

```
dim(Testing)
```

```
## [1] 4904   53
```

Correlation matrix of columns to map correleted predictors:

```
Corr_Matrix <- cor(Training[, -53])
corrplot(Corr_Matrix, order = "FPC", method = "color",
         tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```



Testing Models

This data analysis will test four models: *Classification Tree* , *Random Forest*, *Gradient Boosting Method*, *Linear Discriminant Analysis*.

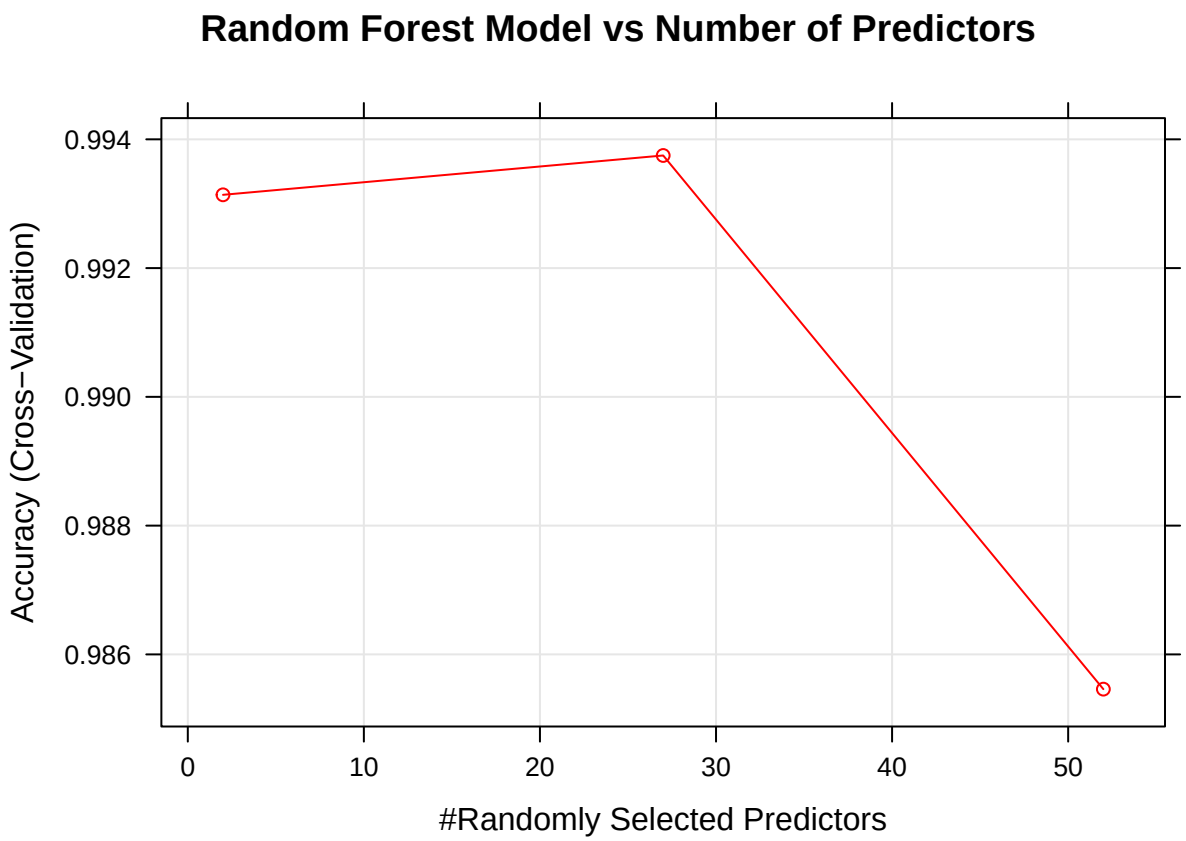
Random Forest model:

```
set.seed(4567)
```

```
trControl <- trainControl(method="cv", number=10)
mod_rf <- train(classe ~ ., data = Training, method = "rf", trControl=trControl, verbose=FALSE, ntree =
```

```
print(mod_rf)
```

```
plot(mod_rf,col="red", main="Random Forest Model vs Number of Predictors")
```



Validating Random Forest model:

```
pred_rf <- predict(mod_rf, Testing)
```

```
ConfMatRF <- confusionMatrix(pred_rf, Testing$classe)
ConfMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393   10    0    0    0
##          B    1  938    5    0    0
##          C    1    1  848    4    0
##          D    0    0    2  800    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9884   0.9918   0.9950   0.9989
## Specificity            0.9972   0.9985   0.9985   0.9993   1.0000
## Pos Pred Value         0.9929   0.9936   0.9930   0.9963   1.0000
## Neg Pred Value         0.9994   0.9972   0.9983   0.9990   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1913   0.1729   0.1631   0.1835
## Detection Prevalence   0.2861   0.1925   0.1741   0.1637   0.1835
## Balanced Accuracy      0.9979   0.9934   0.9952   0.9971   0.9994
```

Confusion matrix and model accuracy

```
ConfMatRF$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1393   10    0    0    0
##          B    1  938    5    0    0
##          C    1    1  848    4    0
##          D    0    0    2  800    1
##          E    0    0    0    0  900
```

Accuracy of the Random Forest model:

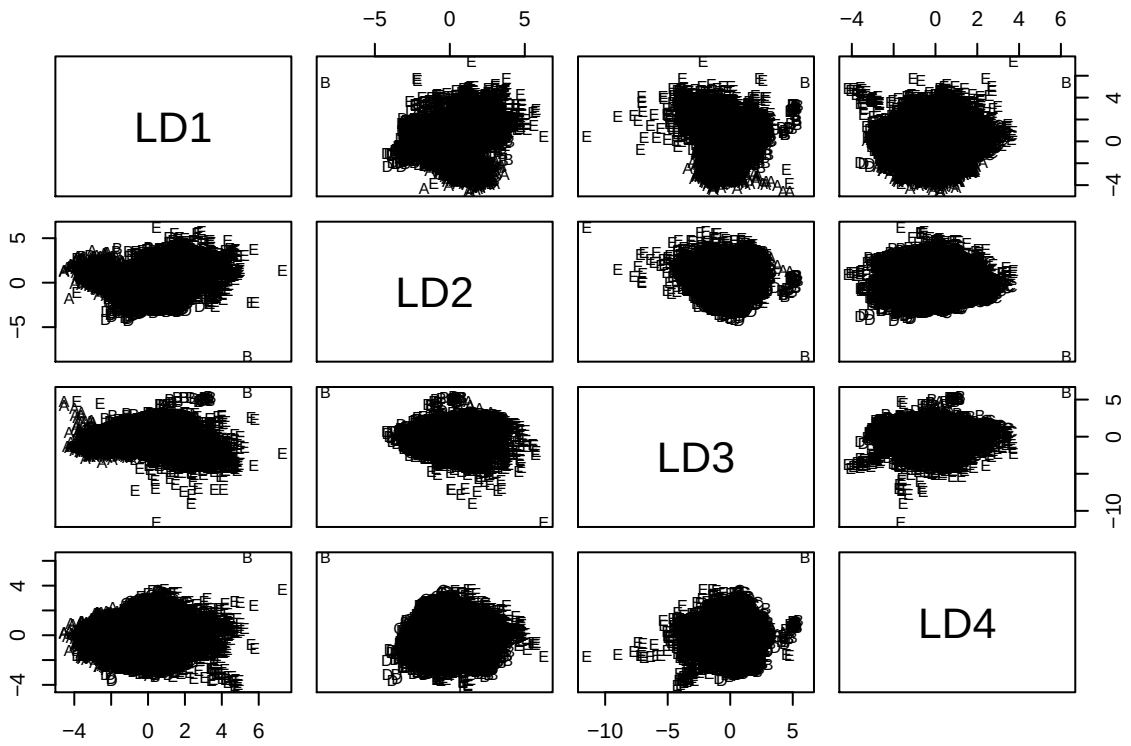
```
ConfMatRF$overall[1]
```

```
##  Accuracy
## 0.9949021
```

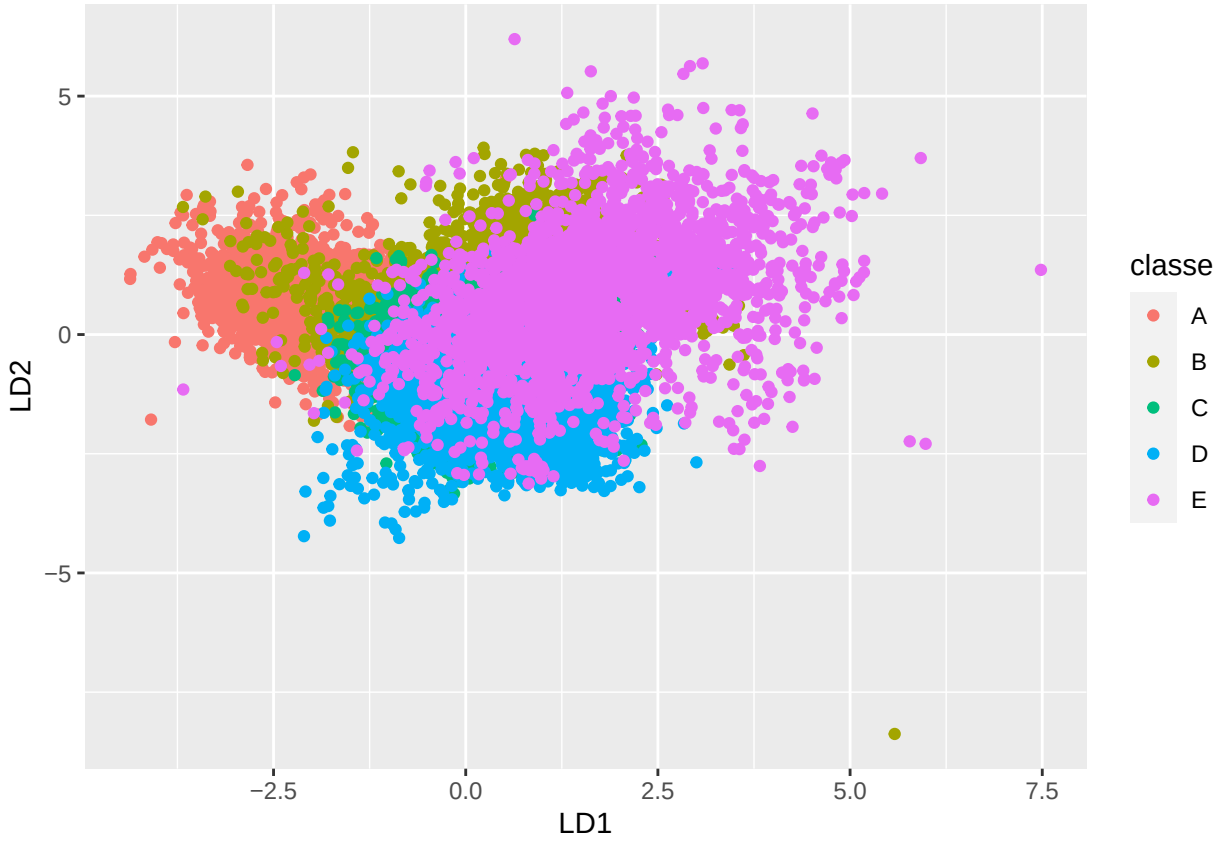
Accuracy of the Random Forest model is high, 99.4% which is most likely due to overfitting.

Linear Discriminant Analysis Model

```
mod_lda <- lda(classe~., data = Training)
plot(mod_lda)
```




```
lda.data <- cbind(Training, predict(mod_lda)$x)
ggplot(lda.data, aes(LD1, LD2)) +
  geom_point(aes(color = classe))
```



Validating model:

```
pred_lda <- predict(mod_lda, Testing)
```

Accuracy of Linear Discriminant Analysis model:

```
mean(pred_lda$class==Testing$classe)
```

```
## [1] 0.6969821
```

Accuracy of model is around 70%.

Misclasification rate:

```
lda.pred = (pred_lda$class)
lda.error = mean(Testing$classe != lda.pred)
lda.error
```

```
## [1] 0.3030179
```

Gradient Boosting Method

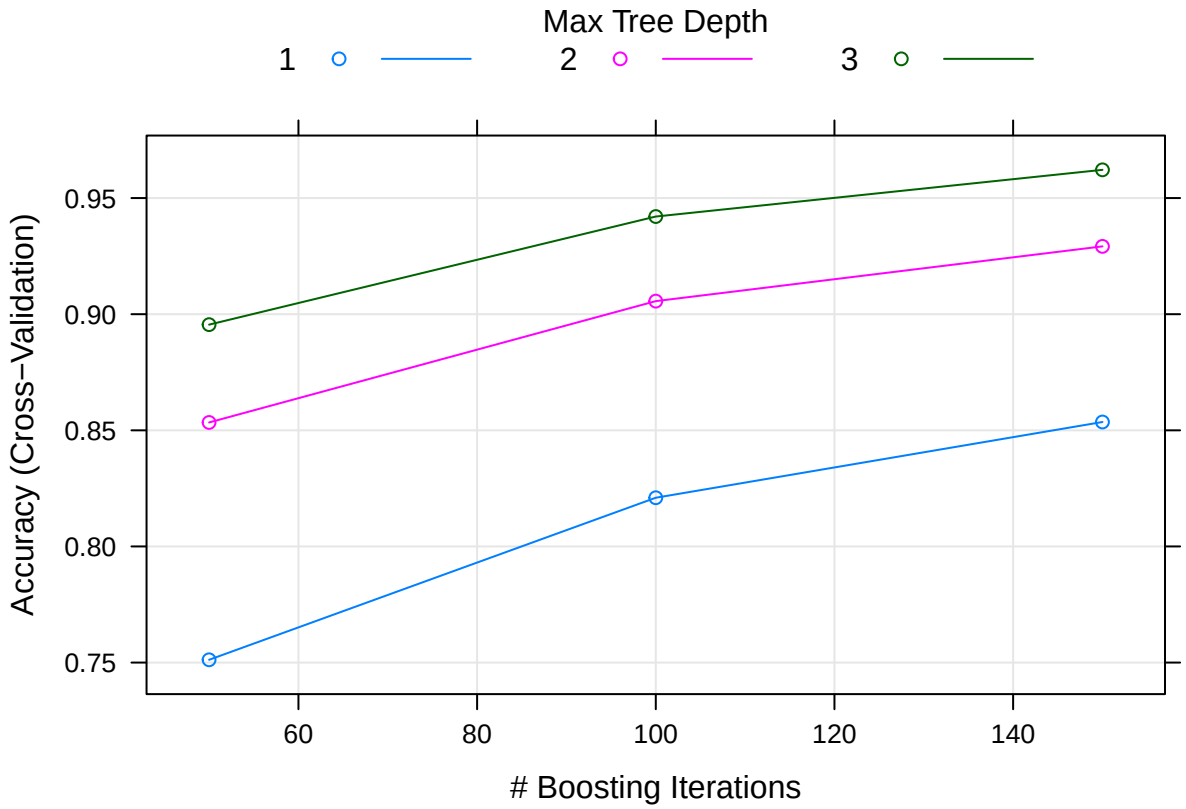
```
Model_GBM <- train(classe~., data=Training, method="gbm", trControl=trControl, verbose=FALSE)
```

```
print(Model_GBM)
```

```
## Stochastic Gradient Boosting
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13247, 13246, 13245, 13245, 13245, 13247, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7511889  0.6844740
##   1                  100      0.8209689  0.7733998
##   1                  150      0.8535819  0.8147349
##   2                   50      0.8533781  0.8142125
##   2                  100      0.9056248  0.8805588
##   2                  150      0.9292031  0.9103939
##   3                   50      0.8955038  0.8677166
##   3                  100      0.9420436  0.9266591
##   3                  150      0.9621551  0.9521181
##
## 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)
```



Validating the boosting method model:

```
Model_GBMpred <- predict(Model_GBM,newdata=Testing)

ConfMatGBM <- confusionMatrix(Model_GBMpred, Testing$classe)
ConfMatGBM$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1379   36    0    1    0
##          B   11  896   34    0   10
##          C    4   16  810   26    5
##          D    1    1   10  769   13
##          E    0    0    1    8  873
```

Accuracy of the model:

```
ConfMatGBM$overall[1]
```

```
## Accuracy
## 0.963907
```

Accuracy of the Gradient Boosting Method is high, around 96.4%.

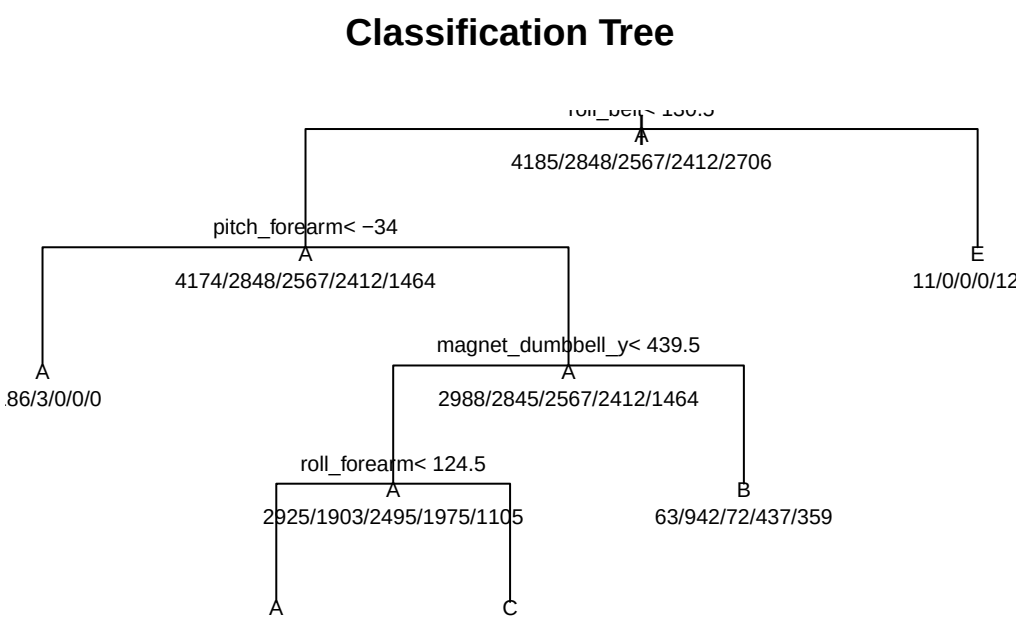
Classification Tree Model

```
trControl <- trainControl(method="cv", number=10)
Model_ClassTree <- train(classe~., data=Training, method="rpart", trControl=trControl)
```

```
Model_ClassTree$finalModel
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 13465  9291 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34 1189     3 A (1 0.0025 0 0 0) *
##      5) pitch_forearm>=-34 12276  9288 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 10403  7478 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 124.5 6467  3844 A (0.41 0.19 0.18 0.17 0.059) *
##         21) roll_forearm>=124.5 3936  2633 C (0.077 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=439.5 1873   931 B (0.034 0.5 0.038 0.23 0.19) *
##    3) roll_belt>=130.5 1253    11 E (0.0088 0 0 0 0.99) *
```

```
plot(Model_ClassTree$finalModel, uniform=TRUE,
     main="Classification Tree")
text(Model_ClassTree$finalModel, use.n=TRUE, all=TRUE, cex=.7)
```

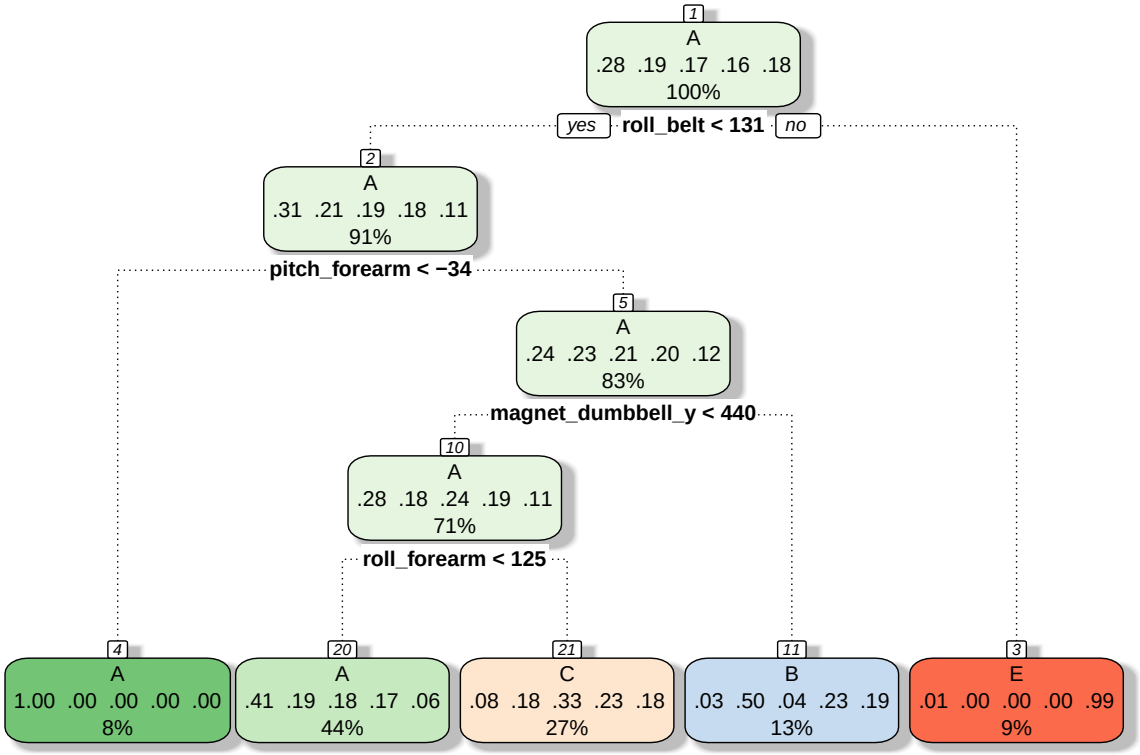



```
print(Model_ClassTree)
```

```
## CART
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13245, 13247, 13246, 13247, 13247, 13247, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03569733  0.5040773  0.35163107
##   0.05949555  0.4054068  0.19051168
##   0.11687079  0.3257347  0.06298508
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03569733.
```

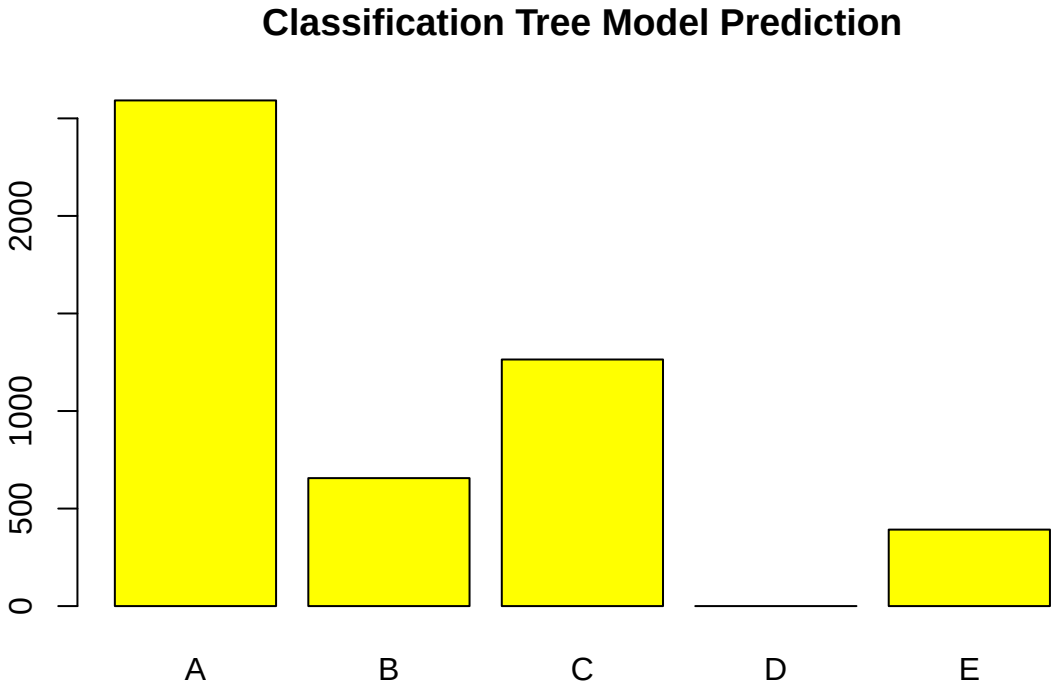
```
fancyRpartPlot(Model_ClassTree$finalModel)
```



Validating Classification Tree model using Testing data set:

```
Model_ClassTreePred <- predict(Model_ClassTree,newdata=Testing)
```

```
plot(Model_ClassTreePred, main="Classification Tree Model Prediction" , col="yellow")
```




```
confMatClassTree <- confusionMatrix(Model_ClassTreePred, Testing$classe)

confMatClassTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1274  390  401  386  141
##          B   18  344   36  131  127
##          C  100  215  418  287  244
##          D    0    0    0    0    0
##          E    3    0    0    0  389
##
## Overall Statistics
##
##                Accuracy : 0.4945
##                  95% CI : (0.4804, 0.5086)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3385
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9133  0.36249  0.48889   0.0000  0.43174
## Specificity            0.6244  0.92111  0.79106   1.0000  0.99925
## Pos Pred Value         0.4915  0.52439  0.33070      NaN  0.99235
## Neg Pred Value         0.9477  0.85758  0.87995   0.8361  0.88652
```

```
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2598  0.07015  0.08524   0.0000  0.07932
## Detection Prevalence   0.5285  0.13377  0.25775   0.0000  0.07993
## Balanced Accuracy      0.7688  0.64180  0.63997   0.5000  0.71550
```

```
confMatClassTree$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1274  390  401  386  141
##          B   18  344   36  131  127
##          C  100  215  418  287  244
##          D    0    0    0    0    0
##          E    3    0    0    0  389
```

Accuracy of the Classification Tree model:

```
confMatClassTree$overall[1]
```

```
##  Accuracy
## 0.4944943
```

Accuracy of Classification Tree model is around 50% which is very low so this model will not predict well

Conclusions

The most accurate results were using *Random Forests model*.

Validating Random Forests model with test dataset:

```
FinalPred <- predict(mod_rf,newdata=testing)
table(FinalPred)
```

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

```
print(FinalPred)
```

```
##  [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
```