Practical Machine Learning Project

Edric Kaw

11/25/2019

Project Introduction

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

Libraries

Loading the required libraries

```
library(caret)
```

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

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

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 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
```

Data Loading

```
training    <- read.csv('pml-training.csv', header=T)
validation  <- read.csv('pml-testing.csv', header=T)
dim(training)
```

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

```
dim(validation)
```

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

```
str(training)
```

```
## '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 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_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]
```

Data Cleansing

From the data structure shown above, noticed that the first seven columns have only little impact to the predictors "classe" and there are many columns with missing values.

Data cleansing process will remove the columns with more than 80% of missing values.

```
# Remove the first seven columns as they have little impact to the outcome classes
training_1    <- training[,-c(1:7)]
validation_1 <- validation[,-c(1:7)]


# Remove columns with more than 99% of missing values for training dataset
Train_Na_Col <- which(colSums(is.na(training_1) |training_1=="")>0.9*dim(training_1)[1])
training_1 <- training_1[,-Train_Na_Col]

# Remove columns with more than 80% of missing values for validation dataset
Valid_Na_Col <- which(colSums(is.na(validation_1) |validation_1=="")>0.9*dim(validation_1)[1])
validation_1 <- validation_1[,-Valid_Na_Col]
dim(training_1)
```

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

```
dim(validation_1)
```

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

From the data cleansing process, the dataset left only 53 variables to be used in the prediction.

Data Processing

Splitting dataset (training dataset) into 75% (training dataset) and 25% (testing dataset) for prediction purposes.

Validation data (originally named testing dataset) will be used later for validation purposes.

```r
set.seed(123)
inTrain <- createDataPartition(training_1$classe, p=0.75, list=FALSE)
trainData <- training_1[inTrain,]
testData  <- training_1[-inTrain,]
dim(trainData)
```

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

```r
dim(testData)
```

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

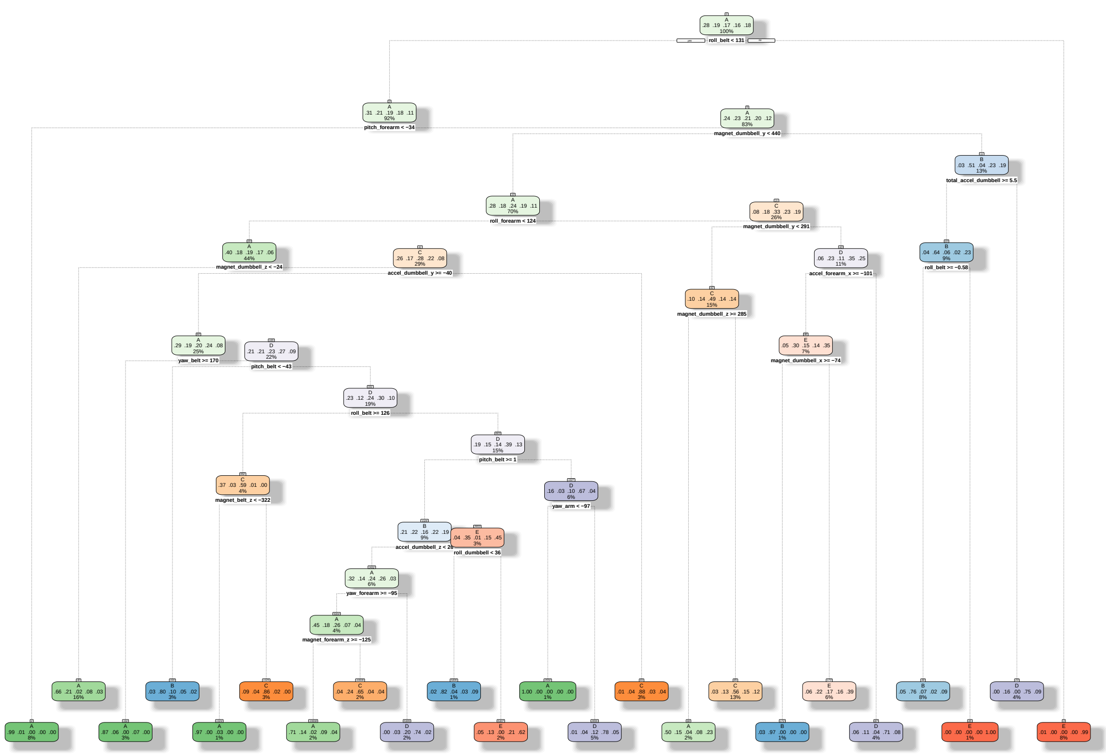
Model Building

In this section, we will using trainData to build three models:
1. Classification tree 2. Random Forest 3. Gradient Boosting Method

Prediction using Classification Tree Model:

```r
ClassTreeModel <- rpart(classe ~ ., data=trainData, method="class")
fancyRpartPlot(ClassTreeModel)
```

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



Rattle 2019−Dec−01 01:31:41 weiching

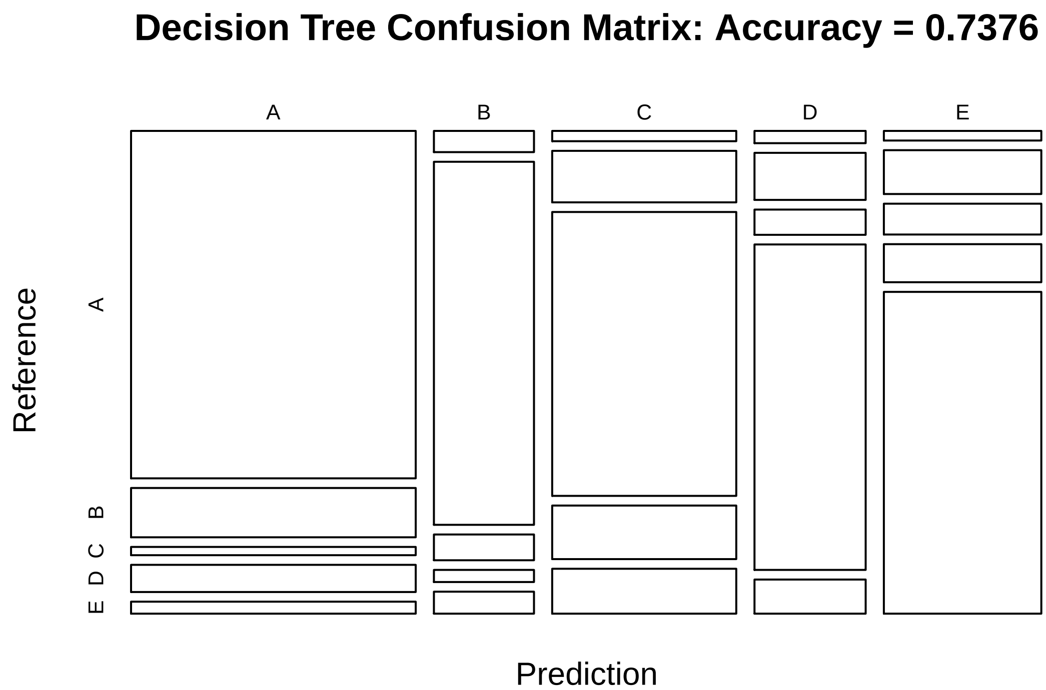
Testing the accuracy of classification tree model using Test Data

```r
PredictClassTreeModel <- predict(ClassTreeModel, testData, type = "class")
cmClassTree <- confusionMatrix(PredictClassTreeModel, testData$classe)
cmClassTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1304  185   31  102   45
##          B   28  479   34   16   29
##          C   25  125  689  130  109
##          D   18   69   37  477   50
##          E   20   91   64   79  668
##
## Overall Statistics
##
##                Accuracy : 0.7376
##                  95% CI : (0.725, 0.7498)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6659
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9348  0.50474   0.8058  0.59328   0.7414
## Specificity            0.8966  0.97295   0.9039  0.95756   0.9365
## Pos Pred Value         0.7822  0.81741   0.6391  0.73272   0.7245
## Neg Pred Value         0.9719  0.89115   0.9566  0.92311   0.9415
## Prevalence             0.2845  0.19352   0.1743  0.16395   0.1837
## Detection Rate         0.2659  0.09768   0.1405  0.09727   0.1362
## Detection Prevalence   0.3399  0.11949   0.2198  0.13275   0.1880
## Balanced Accuracy      0.9157  0.73884   0.8549  0.77542   0.8390
```

Plot matrix result for Classification Tree model:

```r
plot(cmClassTree$table, col = cmClassTree$byClass,
     main = paste("Decision Tree Confusion Matrix: Accuracy =", ... = round(cmClassTree$overall['Accura
```



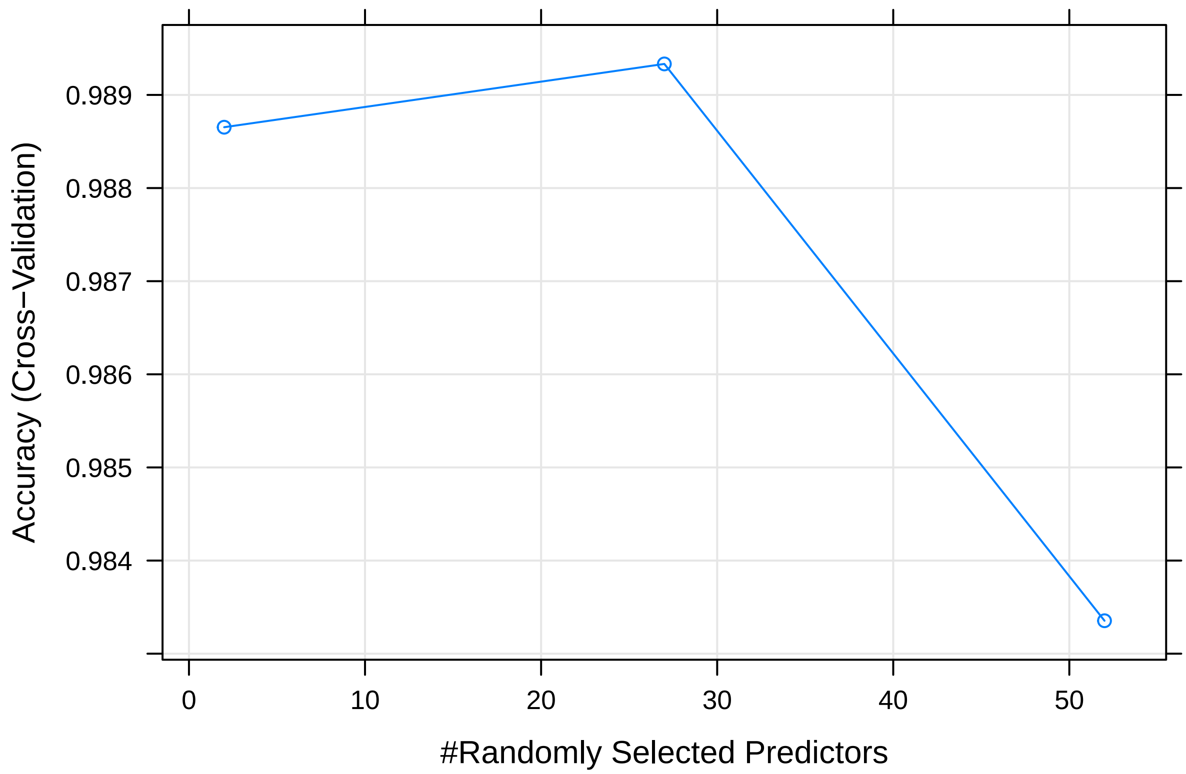
Prediction using Random Forest:

```
set.seed(123)
trControl <- trainControl(method="cv", number=3, verboseIter = FALSE)
randomForestModel <- train(classe ~ ., data=trainData, method="rf", trControl = trControl, verbose=FALS
randomForestModel
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9812, 9812, 9812
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9886533  0.9856460
##   27    0.9893328  0.9865060
##   52    0.9833537  0.9789421
##
## Accuracy was used to select the optimal model using the largest value.
```

```
## The final value used for the model was mtry = 27.
```

```r
plot(randomForestModel)
```



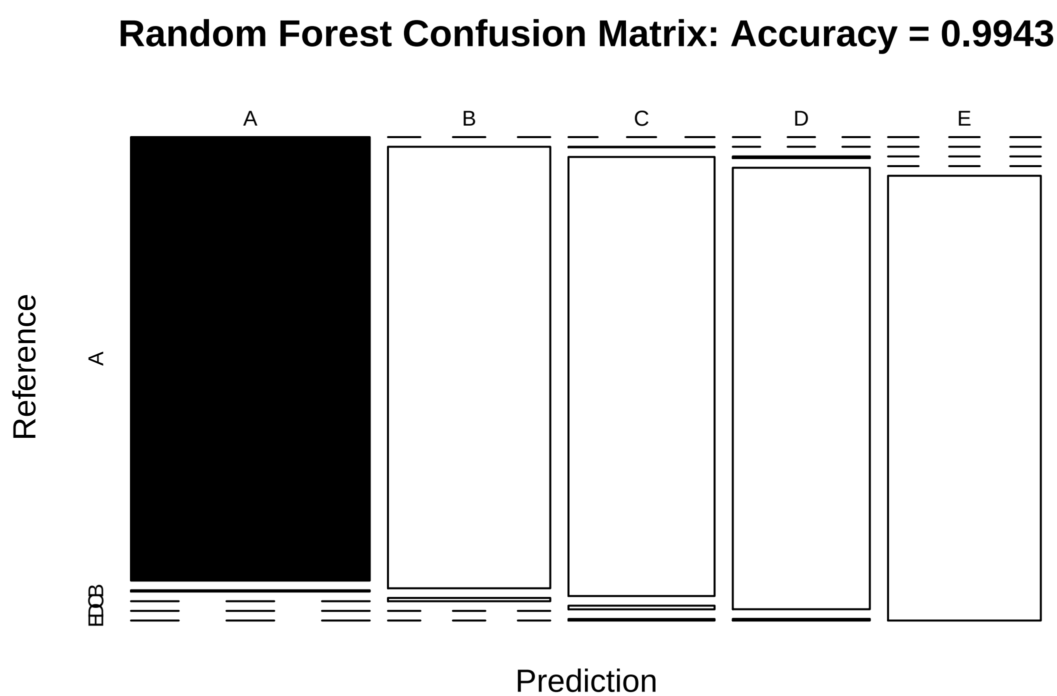

```r
PredictionRandomForestModel <- predict(randomForestModel, testData)
cmRandomForest <- confusionMatrix(PredictionRandomForestModel, testData$classe)
cmRandomForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    4    0    0    0
##          B    0  944    7    0    0
##          C    0    1  845    7    3
##          D    0    0    3  797    3
##          E    0    0    0    0  895
##
## Overall Statistics
##
##                Accuracy : 0.9943
##                  95% CI : (0.9918, 0.9962)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                    Kappa : 0.9928
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9883   0.9913   0.9933
## Specificity            0.9989   0.9982   0.9973   0.9985   1.0000
## Pos Pred Value         0.9971   0.9926   0.9871   0.9925   1.0000
## Neg Pred Value         1.0000   0.9987   0.9975   0.9983   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1723   0.1625   0.1825
## Detection Prevalence   0.2853   0.1939   0.1746   0.1637   0.1825
## Balanced Accuracy      0.9994   0.9965   0.9928   0.9949   0.9967
```

Plot matrix result for Random Forest Model:

```r
plot(cmRandomForest$table, col = cmRandomForest$byClass,
                main = paste("Random Forest Confusion Matrix: Accuracy =", round(cmRandomForest$overall
```

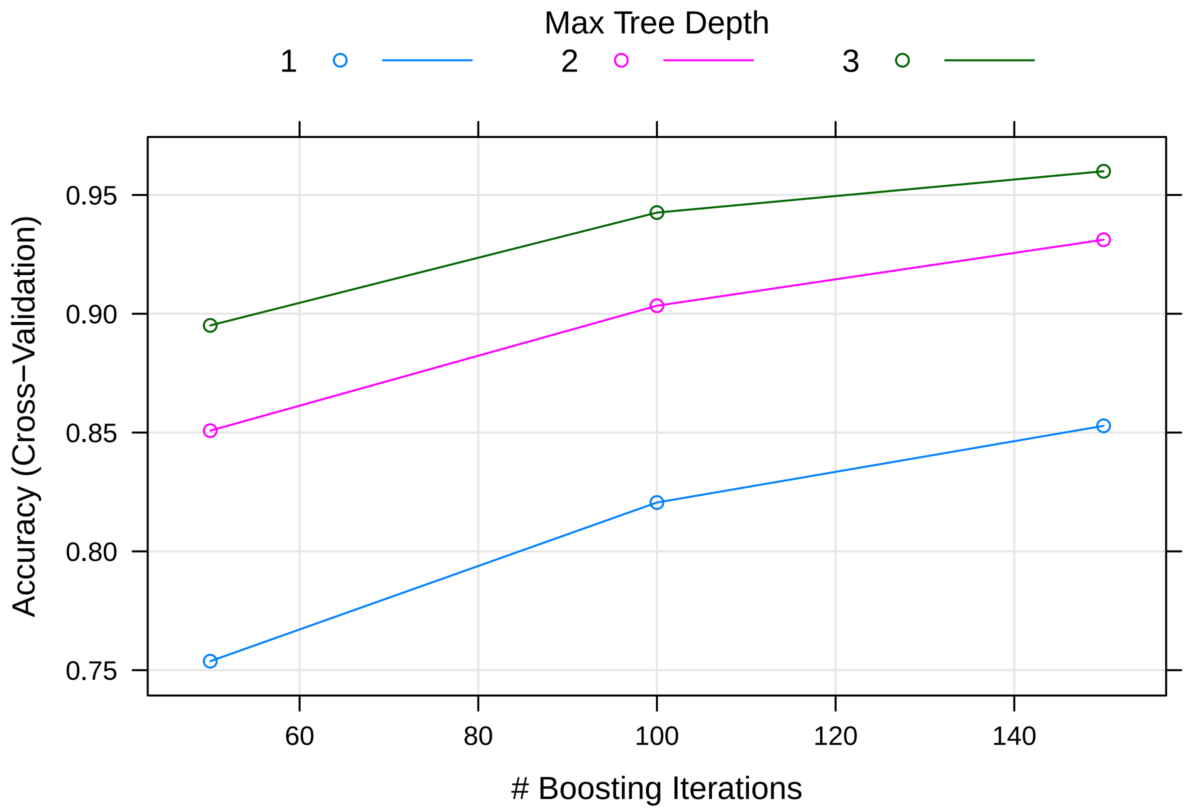


Prediction using Gradient Boosting Method:

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

```
## Stochastic Gradient Boosting
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9811, 9813, 9812
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7537707  0.6879173
##   1                  100      0.8205595  0.7728805
##   1                  150      0.8528328  0.8138268
##   2                   50      0.8507948  0.8109329
##   2                  100      0.9033832  0.8777094
##   2                  150      0.9311727  0.9129087
##   3                   50      0.8950941  0.8671898
##   3                  100      0.9425873  0.9273503
##   3                  150      0.9599811  0.9493725
##
## 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.
```

```r
plot(GBMModel)
```

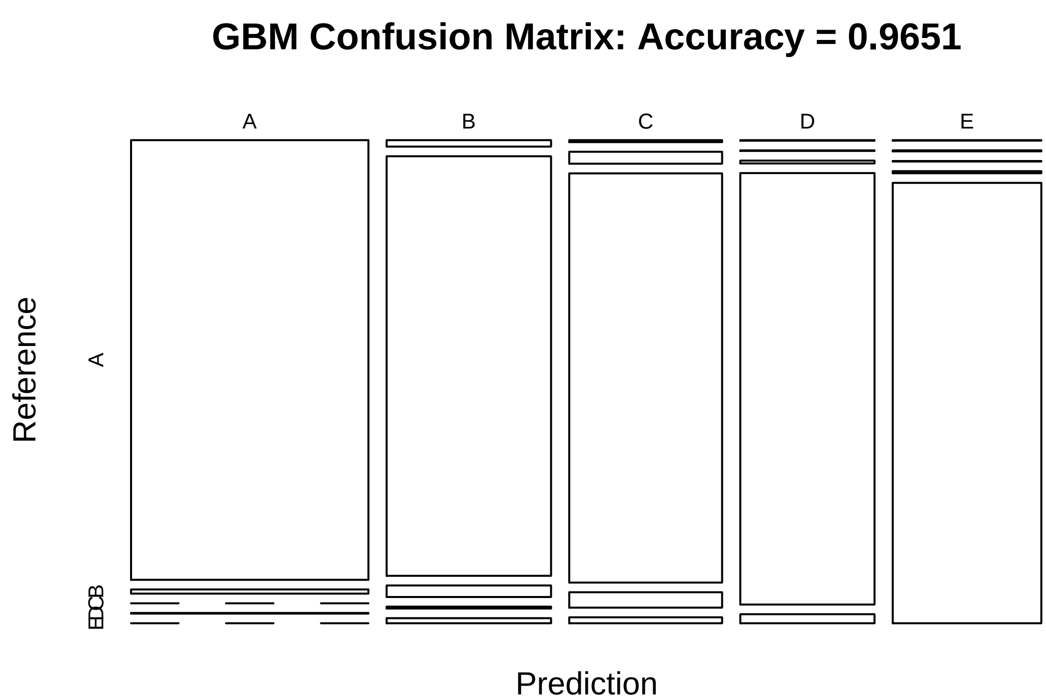



```r
PredictionGBMModel <- predict(GBMModel, testData)
cmGBM <- confusionMatrix(PredictionGBMModel, testData$classe)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1375   13    0    2    0
##          B   14  909   25    4   11
##          C    4   24  824   31   12
##          D    1    1    5  763   16
##          E    1    2    1    4  862
##
## Overall Statistics
##
##                Accuracy : 0.9651
##                  95% CI : (0.9596, 0.9701)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9559
##
##  Mcnemar's Test P-Value : 4.609e-07
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9857   0.9579   0.9637   0.9490   0.9567
## Specificity            0.9957   0.9863   0.9825   0.9944   0.9980
## Pos Pred Value         0.9892   0.9439   0.9207   0.9707   0.9908
## Neg Pred Value         0.9943   0.9899   0.9923   0.9900   0.9903
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2804   0.1854   0.1680   0.1556   0.1758
## Detection Prevalence   0.2834   0.1964   0.1825   0.1603   0.1774
## Balanced Accuracy      0.9907   0.9721   0.9731   0.9717   0.9774
```

```
plot(cmGBM$table, col = cmGBM$byClass, main = paste("GBM Confusion Matrix: Accuracy =", round(cmGBM$ove
```



Conclusion

Random Forest Model having the highest accuracy (*0.9943*) in prediction compared to other models. Out-of-sample-error is only 0.0557.

Data Validation

For the data validation, we will using *random forest model* as it has the highest accuracy.

```r
Prediction <- predict(randomForestModel,newdata=validation_1)
Prediction
```

```
##  [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
```