Prediction Assignment Writeup

Shiqi Yang

9/11/2019

Executive Summary

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, our 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.

Data Loading and Exploring

```
# load data
training <- read.csv('./pml-training.csv', header=T)
testing <- read.csv('./pml-testing.csv', header=T)
dim(training)

## [1] 19622   160

dim(testing)

## [1]  20 160

# explore data
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 1323084232 1323084232 1323084232 1323084232
...
##  $ 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 -0.02 -
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]
```

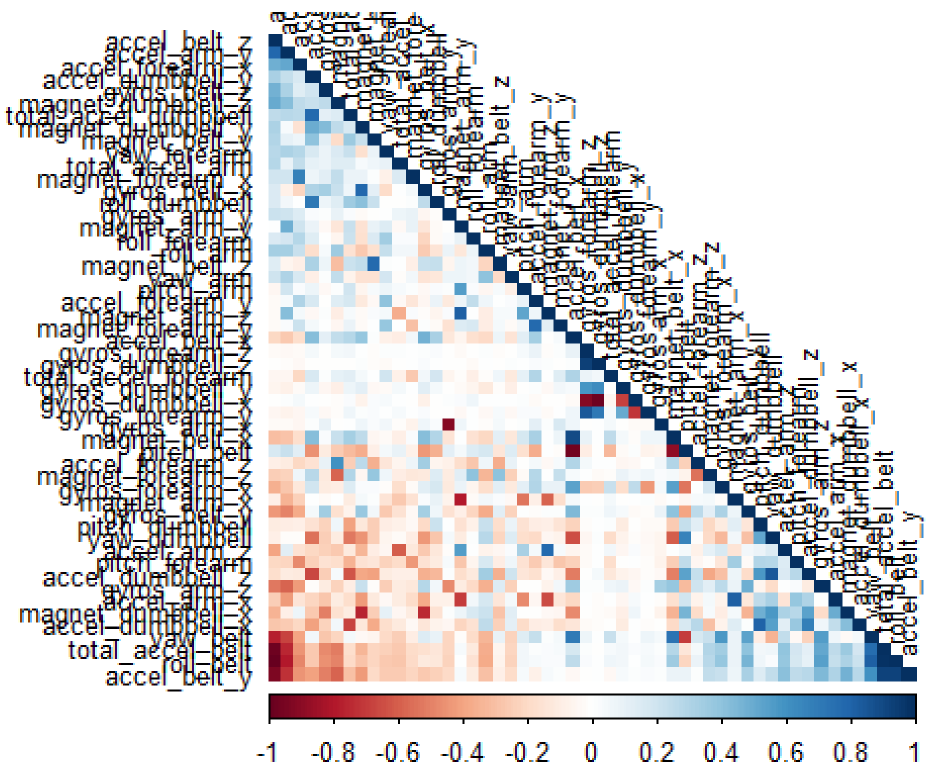
We can notice that many columns have NA values or blank values on almost every observation. So we will remove them, because they will not produce any information. The first seven columns give information about the people who did the test, and also timestamps. We will not take them in our model.

```r
# remove columns having at least 90% of NA or blank values on the training
dataset
indColToRemove <-
which(colSums(is.na(training)|training=="")>0.9*dim(training)[1])
TrainDataClean <- training[,-indColToRemove]
# remove first 7 columns
TrainDataClean <- TrainDataClean[,-c(1:7)]
dim(TrainDataClean)
```

```
## [1] 19622    53

# remove columns having at least 90% of NA or blank values on the testing
dataset
indColToRemove <- which(colSums(is.na(testing)
|testing=="")>0.9*dim(testing)[1])
TestDataClean <- testing[,-indColToRemove]
dim(TestDataClean)

## [1] 20 60
```

Correlation Analysis

```
library(corrplot)

## corrplot 0.84 loaded

corMatrix <- cor(TrainDataClean[, -53])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



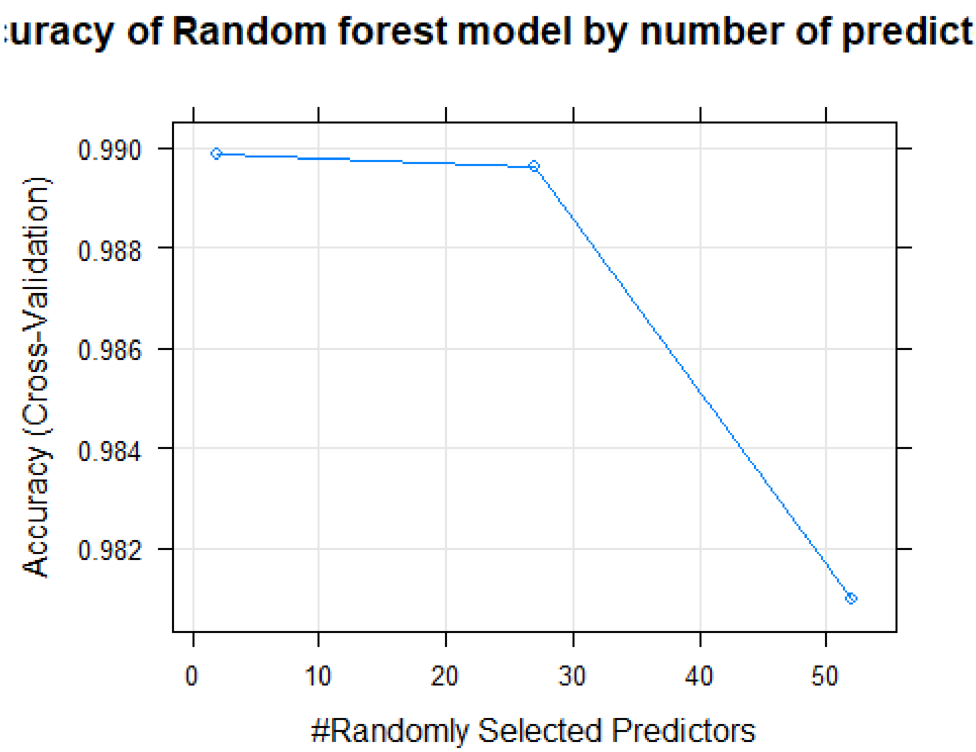
Model Building

Three methods will be applied - Random Forests, Classification Tree, and Gradient Boosted Model. In order to limit the effects of overfitting, and improve the efficicency of the models, we will use the cross-validation technique (3 folds).

Random Forests

```
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)

# Data partition
set.seed(12345)
inTrain1 <- createDataPartition(TrainDataClean$classe, p=0.75, list=FALSE)
Train1 <- TrainDataClean[inTrain1,]
Test1 <- TrainDataClean[-inTrain1,]
# Model fit
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFit_RandForest <- train(classe ~ ., data=Train1, method="rf",
trControl=controlRF)
modFit_RandForest

## 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: 9813, 9811, 9812
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9898762  0.9871920
##   27    0.9896044  0.9868486
##   52    0.9809759  0.9759325
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

# predict on test set
predict_RandForest <- predict(modFit_RandForest, newdata=Test1)
confMat_RandForest <- confusionMatrix(predict_RandForest, Test1$classe)

# display confusion matrix and model accuracy
confMat_RandForest$table; confMat_RandForest$overall[1]
```

```
##            Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  948    4    0    0
##          C    0    0  851   15    0
##          D    0    0    0  784    1
##          E    0    0    0    5  900

##  Accuracy
## 0.9946982

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



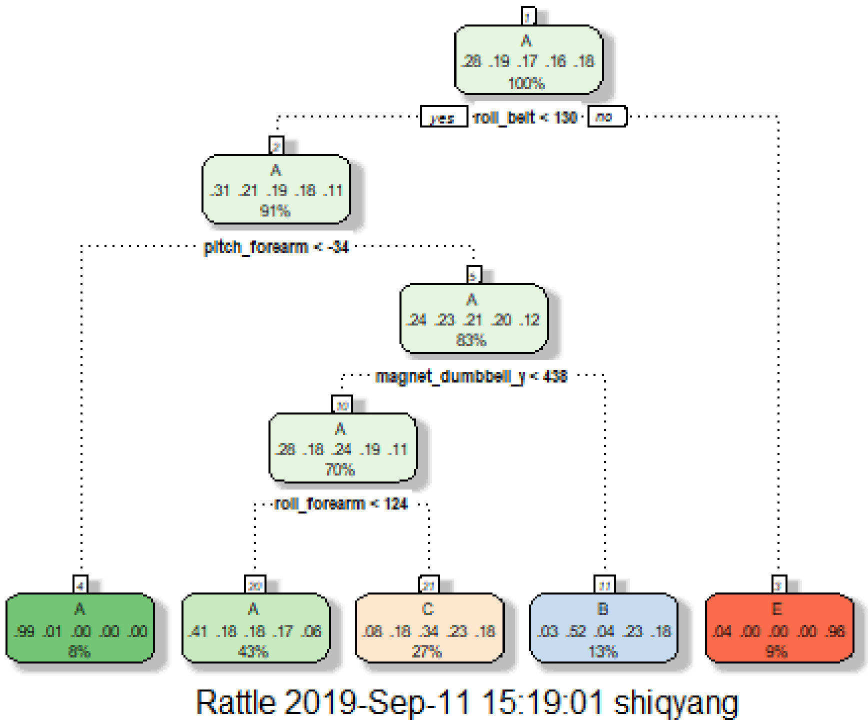

```
# Compute the variable importance
MostImpVars <- varImp(modFit_RandForest)
MostImpVars

## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                    Overall
## roll_belt           100.00
## yaw_belt             85.26
## magnet_dumbbell_z    71.54
## pitch_belt           64.88
```

```
## magnet_dumbbell_y   62.45
## pitch_forearm       59.98
## roll_forearm        52.94
## magnet_dumbbell_x   51.00
## accel_belt_z        47.69
## accel_dumbbell_y    45.83
## magnet_belt_z       43.87
## magnet_belt_y       43.22
## roll_dumbbell       42.18
## roll_arm            36.48
## accel_forearm_x     35.00
## accel_dumbbell_z    34.56
## accel_dumbbell_x    30.08
## yaw_dumbbell        28.82
## gyros_belt_z        28.61
## gyros_dumbbell_y    28.48
```

It appears that the optimal number of predictors is two, which provides 99% accuracy using cross-validation with 3 folds. The out of sample error is expected to be 1%. The accuracy drops as the number of predictors increases. The top two most important variables are roll_belt and yaw_belt.

Classification Trees

```r
# model fit
set.seed(12345)
ControlCT <- trainControl(method="cv", number=3)
modFit_Tree <- train(classe ~ ., data=Train1, method="rpart",
trControl=ControlCT)
fancyRpartPlot(modFit_Tree$finalModel)
```




```
# prediction on Test dataset
predict_Tree <- predict(modFit_Tree, newdata=Test1)
confMat_Tree <- confusionMatrix(predict_Tree, Test1$classe)

# display confusion matrix and model accuracy
confMat_Tree$table; confMat_Tree$overall[1]

##             Reference
## Prediction    A    B    C    D    E
##          A 1252  396  434  343  114
##          B   30  317   24  151  132
##          C   90  236  397  310  229
##          D    0    0    0    0    0
##          E   23    0    0    0  426

##  Accuracy
## 0.4877651
```

The classification tree provides only 49% accuracy which is not a good model for predicting “classe”. The out of sample error is expected to be 51%.

Gradient boosting machine

```
# model fit
set.seed(12345)
ControlCT <- trainControl(method="cv", number=3)
modFit_GBM <- train(classe~., data=Train1, method="gbm", trControl=ControlCT,
```

```
verbose=FALSE)
modFit_GBM

## 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: 9812, 9812, 9812
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7505096  0.6834086
##   1                  100      0.8192010  0.7710836
##   1                  150      0.8515423  0.8121142
##   2                   50      0.8536486  0.8145984
##   2                  100      0.9053540  0.8802344
##   2                  150      0.9304933  0.9120383
##   3                   50      0.8952303  0.8673463
##   3                  100      0.9394619  0.9233917
##   3                  150      0.9588259  0.9479042
##
## 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.

# prediction on Test dataset
predict_GBM <- predict(modFit_GBM,newdata=Test1)
confMat_GBM <- confusionMatrix(predict_GBM, Test1$classe)

# display confusion matrix and model accuracy
confMat_GBM$table; confMat_GBM$overall[1]

##           Reference
## Prediction    A    B    C    D    E
##          A 1374   20    0    2    0
##          B   17  899   21    3    6
##          C    1   30  817   30   10
##          D    2    0   17  762    8
##          E    1    0    0    7  877

##  Accuracy
## 0.9643148
```

The gradient boosting machine model provides 96% accuracy. The out of sample error is expected to be 4%.

Conclusion

The random forest model provides the highest accuracy, which will be used to predict classe for the test data set.

```
FinalPredict_RandForest <- predict(modFit_RandForest, newdata=TestDataClean)
FinalPredict_RandForest

##  [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
```