# Assignment: Prediction Assignment Writeup

*Josue Galarreta*

*May 12, 2020*

## 1. Overview

This document is the final report of the Peer Assessment project from Coursera's course Practical Machine Learning, as part of the Specialization in Data Science. It was built in RStudio, using its knitr functions, meant to be published in html format. This analysis meant to be the basis for the course quiz and a prediction assignment writeup. The main goal of the project is to predict the manner in which 6 participants performed some exercise as described below. This is the "classe" variable in the training set. The machine learning algorithm described here is applied to the 20 test cases available in the test data and the predictions are submitted in appropriate format to the Course Project Prediction Quiz for automated grading.

## 2. 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://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)]http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

## 3. Data Loading and Exploratory Analysis

### a. Data Source

The training data for this project are available here:

[Training Set]https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

[Test Set]https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

## b. Environment Setup

```
library(knitr)
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
library(corrplot)
set.seed(301)
```

## c. Data Loading and Cleaning

The next step is loading the dataset from the URL provided above. The training dataset is then partinioned in 2 to create a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations. The testing dataset is not changed and will only be used for the quiz results generation.

```
TrainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.c
sv"
TestUrl  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v"
TrainFile<-"pml-traininig.csv"
TestFile<-"pml-testing.csv"

# download the datasets
if(!file.exists(TrainFile))
{
    download.file(TrainUrl,destfile = TrainFile)
}
training <- read.csv(TrainFile)
if(!file.exists(TestFile))
{
    download.file(TestUrl,destfile = TestFile)
}
testing  <- read.csv(TestFile)

# create a partition using caret with the training dataset on 70,30 ratio
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)

TrainSet <- training[inTrain, ]

TestSet  <- training[-inTrain, ]
dim(TrainSet)
```

```
## [1] 13737   160
```

```
dim(TestSet)
```

```
## [1] 5885  160
```

Both created datasets have 160 variables. Let’s clean NA, The Near Zero variance (NZV) variables and the ID variables as well.

```
# remove variables with Nearly Zero Variance
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TestSet)
```

```
## [1] 5885  106
```

```
dim(TrainSet)
```

```
## [1] 13737   106
```

```
# remove variables that are mostly NA
AllNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet  <- TestSet[, AllNA==FALSE]
dim(TestSet)
```

```
## [1] 5885   59
```

```
dim(TrainSet)
```

```
## [1] 13737    59
```

```
# remove identification only variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

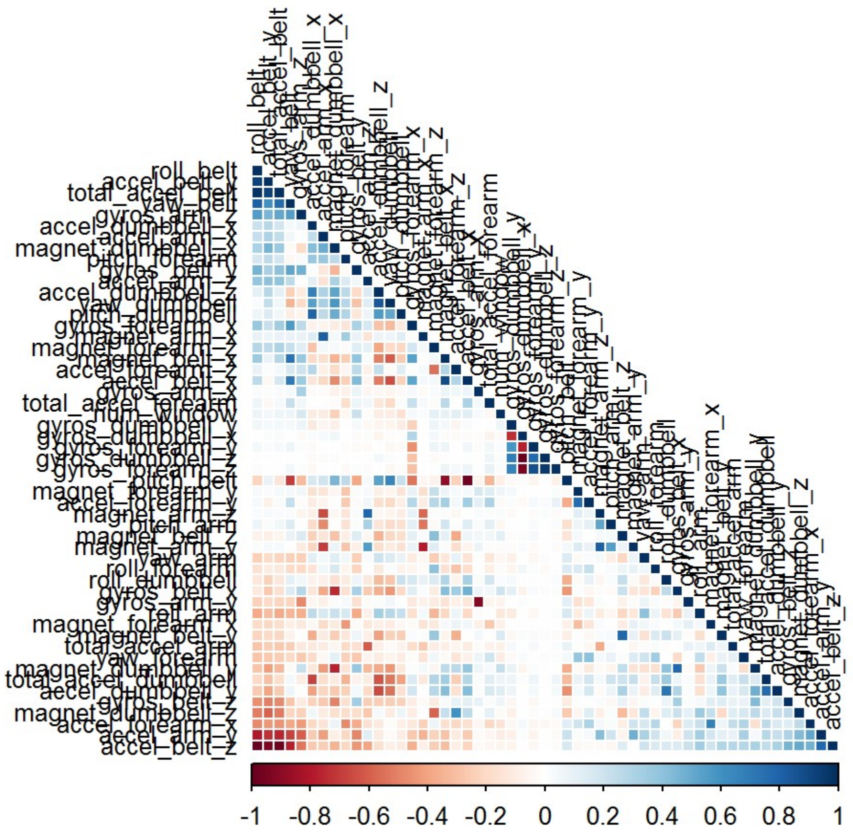
```
## [1] 13737    54
```

After cleaning, we can see that the number of vairables for the analysis are now only 53.

## d.Coorection Analysis

A correlation among variables is analysed before proceeding to the modeling procedures.

```
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



The highly correlated variables are shown in dark colors in the graph above. To make an even more compact analysis, a PCA (Principal Components Analysis) could be performed as pre-processing step to the datasets. Nevertheless, as the correlations are quite few, this step will not be applied for this assignment.

# 4. Prediction Model Building

Three popular methods will be applied to model the regressions (in the Train dataset) and the best one (with higher accuracy when applied to the Test dataset) will be used for the quiz predictions. The methods are: Random Forests, Decision Tree and Generalized Boosted Model, as described below. A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models.

## a. Random Forests

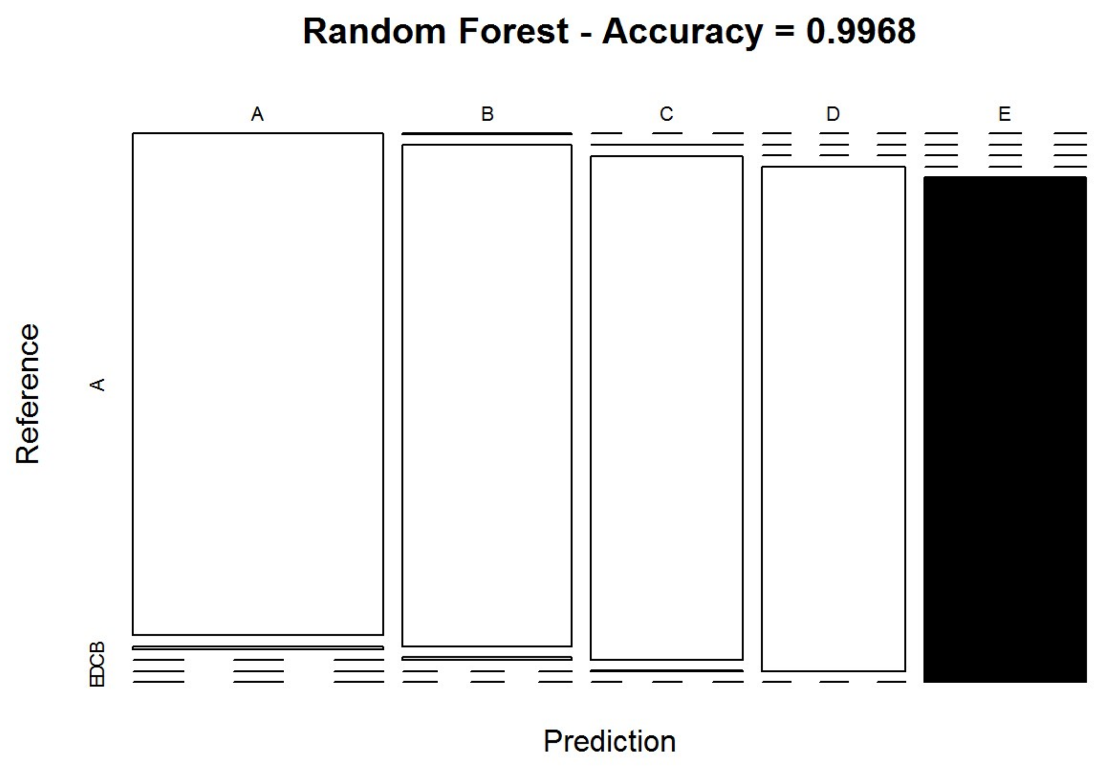
```
# model fit
set.seed(301)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                          trControl=controlRF)
modFitRandForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.28%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    6 2647    4    1    0 0.0041384500
## C    0    6 2389    1    0 0.0029215359
## D    0    0   11 2240    1 0.0053285968
## E    0    1    0    7 2517 0.0031683168
```

```
# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, TestSet$classe)
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673   10    0    0    0
##          B    1 1128    6    0    0
##          C    0    1 1020    1    0
##          D    0    0    0  963    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9903   0.9942   0.9990   1.0000
## Specificity            0.9976   0.9985   0.9996   1.0000   1.0000
## Pos Pred Value         0.9941   0.9938   0.9980   1.0000   1.0000
## Neg Pred Value         0.9998   0.9977   0.9988   0.9998   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1917   0.1733   0.1636   0.1839
## Detection Prevalence   0.2860   0.1929   0.1737   0.1636   0.1839
## Balanced Accuracy      0.9985   0.9944   0.9969   0.9995   1.0000
```

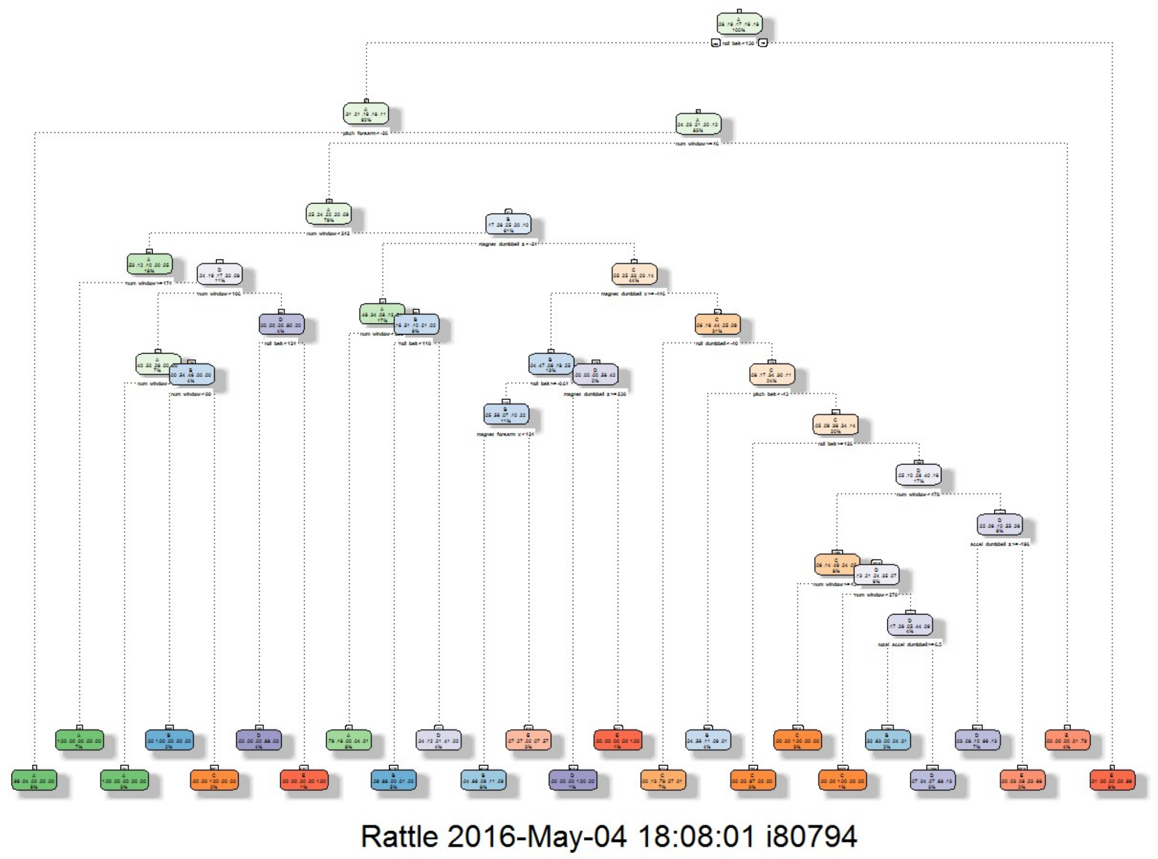
```
# plot matrix results
plot(confMatRandForest$table, col = confMatRandForest$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(confMatRandForest$overall['Accuracy'], 4)))
```



## b. Decision Tree

```
# model fit
set.seed(301)
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitDecTree)
```

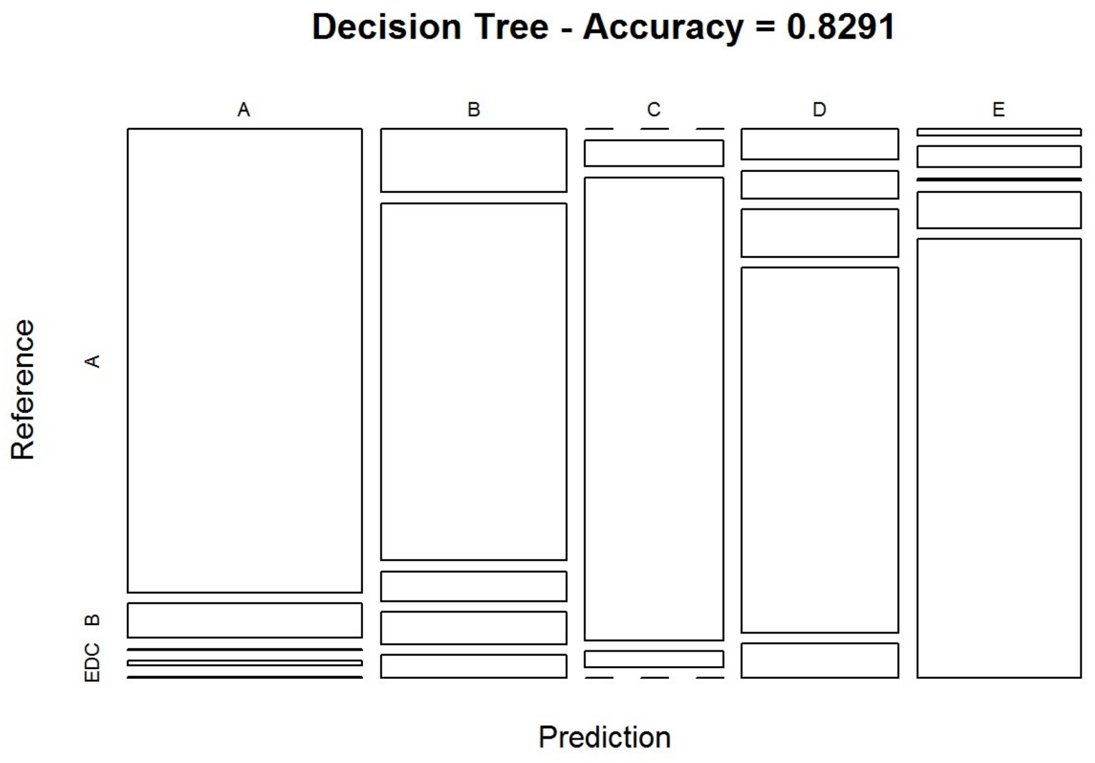
```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
# prediction on Test dataset
predictDecTree <- predict(modFitDecTree, newdata=TestSet, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, TestSet$classe)
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1441  107    2   15    5
##          B  156  880   73   80   56
##          C    0   48  848   29    0
##          D   64   58   98  761   72
##          E   13   46    5   79  949
##
## Overall Statistics
##
##                Accuracy : 0.8291
##                  95% CI : (0.8192, 0.8386)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7843
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8608   0.7726   0.8265   0.7894   0.8771
## Specificity            0.9694   0.9231   0.9842   0.9407   0.9702
## Pos Pred Value         0.9178   0.7068   0.9168   0.7227   0.8690
## Neg Pred Value         0.9460   0.9442   0.9641   0.9580   0.9723
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2449   0.1495   0.1441   0.1293   0.1613
## Detection Prevalence   0.2668   0.2116   0.1572   0.1789   0.1856
## Balanced Accuracy      0.9151   0.8479   0.9053   0.8650   0.9237
```

```r
# plot matrix results
plot(confMatDecTree$table, col = confMatDecTree$byClass,
     main = paste("Decision Tree - Accuracy =",
                  round(confMatDecTree$overall['Accuracy'], 4)))
```



## c. Generalized Boosted Model (GBM)

```
# model fit
set.seed(301)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=TrainSet, method = "gbm",
                    trControl = controlGBM, verbose = FALSE)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.2.4
```

```
## Loading required package: survival
```

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

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

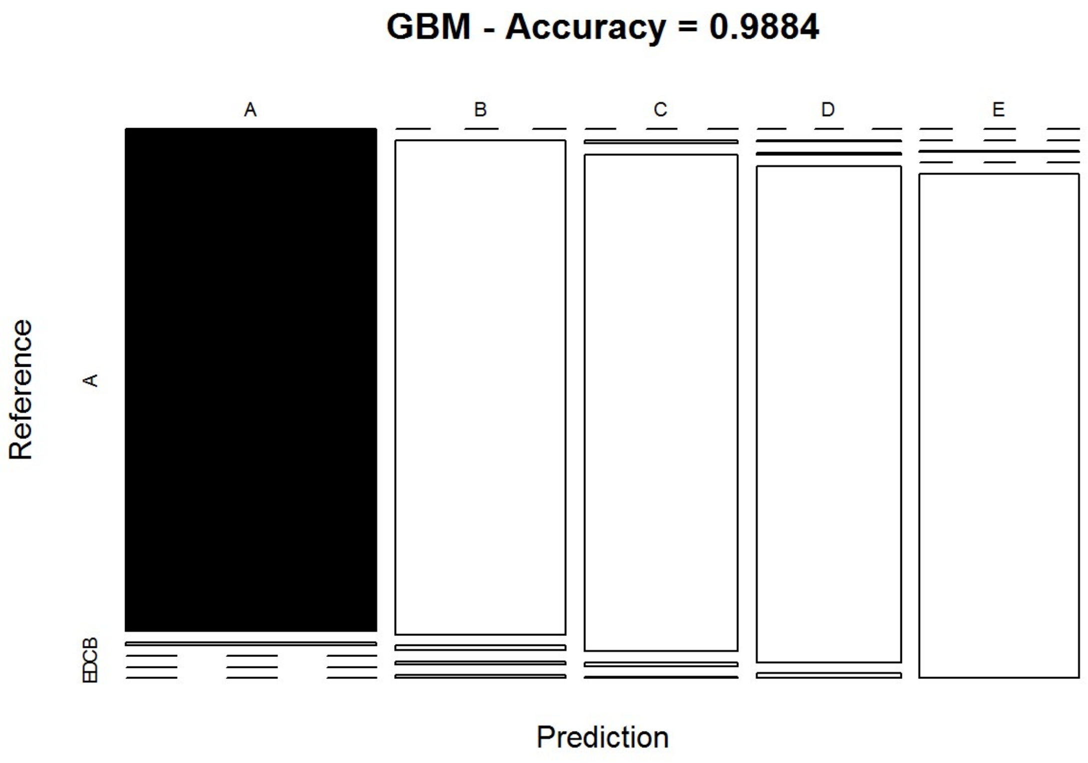
```
modFitGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 41 had non-zero influence.
```

```
# prediction on Test dataset
predictGBM <- predict(modFitGBM, newdata=TestSet)
confMatGBM <- confusionMatrix(predictGBM, TestSet$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   10    0    0    0
##          B    0 1119   11    7    6
##          C    0    7 1010    8    2
##          D    0    3    4  949    9
##          E    0    0    1    0 1065
##
## Overall Statistics
##
##                Accuracy : 0.9884
##                  95% CI : (0.9854, 0.991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9854
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9824   0.9844   0.9844   0.9843
## Specificity            0.9976   0.9949   0.9965   0.9967   0.9998
## Pos Pred Value         0.9941   0.9790   0.9834   0.9834   0.9991
## Neg Pred Value         1.0000   0.9958   0.9967   0.9970   0.9965
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1901   0.1716   0.1613   0.1810
## Detection Prevalence   0.2862   0.1942   0.1745   0.1640   0.1811
## Balanced Accuracy      0.9988   0.9887   0.9905   0.9906   0.9920
```

```
# plot matrix results
plot(confMatGBM$table, col = confMatGBM$byClass,
     main = paste("GBM - Accuracy =", round(confMatGBM$overall['Accuracy'],
4)))
```



## 5. Applying the selected Model to the Test Data

The accuracy of the 3 regression modeling methods above are:

Random Forest : 0.9968 Decision Tree : 0.8291 GBM : 0.9884 In that case, the Random Forest model will be applied to predict the 20 quiz results (testing dataset) as shown below.

```
predictTEST <- predict(modFitRandForest, newdata=testing)
predictTEST
```

```
##  [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
```