Machine Learning Project

Richard Lai

March 3, 2017

Table of Contents:

A. 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, 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.

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

B. Dataset

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from http://groupware.les.inf.puc-rio.br/har.

Full source reference:

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.

Note: A special thank you to the above authors for such generous in allowing their data to be used for this kind of assignment.

The model layout

Our outcome for this data set is "classe" variable in the training set. A factor variable with 5 levels. Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

```
* Exactly according to the specification [Class A]
* Throwing the elbows to the front [Class B]
* Lifting the dumbbell only halfway [Class C]
* Lowering the dumbbell only halfway [Class D]
* Throwing the hips to the front [Class E]
```

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate.

The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg)."

C. Instruction of submission for this project

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. We use any of the other variables to predict with. A report output describing how to built the model, how we used cross validation, what we think the expected out of sample error is, and why we made the choices we did. We will also use this prediction model to predict 20 different test cases.

Peer Review

The submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).

Course Project Prediction Quiz Portion

Apply the machine learning algorithm to the 20 test cases available in the test data above and submit your predictions in appropriate format to the Course Project Prediction Quiz for automated grading.

D. Read the URL and Download the Datasets

```
UrlDwnload_Train <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

UrlDwnload_Test  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```
training <- read.csv(url(UrlDwnload_Train))
testing  <- read.csv(url(UrlDwnload_Test))
```

E. Exploratory Data Analysis

Reproducibility

Installing the different packages, loading libraries, and setting the seed for working environment:

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.3.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.2
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.3.2
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.3.2
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.2
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.3.2
```

```
# setting the overall seed for reproduceability
set.seed(12345)


# Cross Validation

# Create partition of Train (75%) and Test (25%) dataset
SetupTrain  <- createDataPartition(training$classe, p=0.75, list=FALSE)
TraingSet <- training[SetupTrain, ]
TestgSet  <- training[-SetupTrain, ]
```

Train Set

```
dim(TraingSet)
```

```
## [1] 14718   160
```

Test Set

```
dim(TestgSet)
```

```
## [1] 4904  160
```

Remove any variables with zero value for Train and Test Set

```
dim(TraingSet)
```

```
## [1] 14718   105
```

```
dim(TestgSet)
```

```
## [1] 4904  105
```

Remove any variables of NA for Train and Test Set

```
dim(TraingSet)
```

```
## [1] 14718    59
```

```
dim(TestgSet)
```

```
## [1] 4904   59
```

Remove identification variables for columns 1 to 5

Train Set

```
dim(TraingSet)
```

```
## [1] 14718    54
```

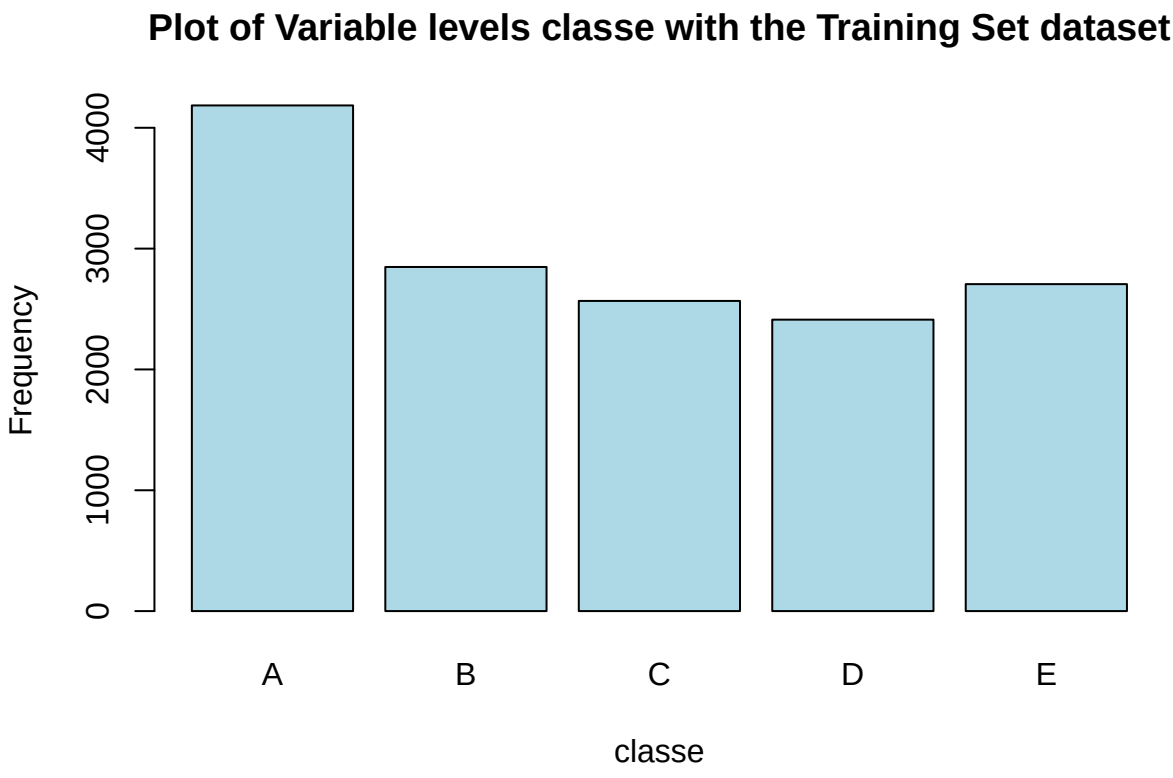
Test Set

```
dim(TestgSet)
```

```
## [1] 4904   54
```

F. Correlation Analysis

```
plot(TraingSet$classe, col="light blue", main="Plot of Variable levels classe with the Training Set data
```



G. Random Forest Model

Model Fit

```
set.seed(12345)
contrlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
ModelRF <- train(classe ~ ., data=TraingSet, method="rf", trControl=contrlRF)
ModelRF$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: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4184    0    0    0    1 0.0002389486
## B    5 2841    1    1    0 0.0024578652
## C    0    3 2564    0    0 0.0011686794
## D    0    0   10 2401    1 0.0045605307
## E    0    0    0    8 2698 0.0029563932
```

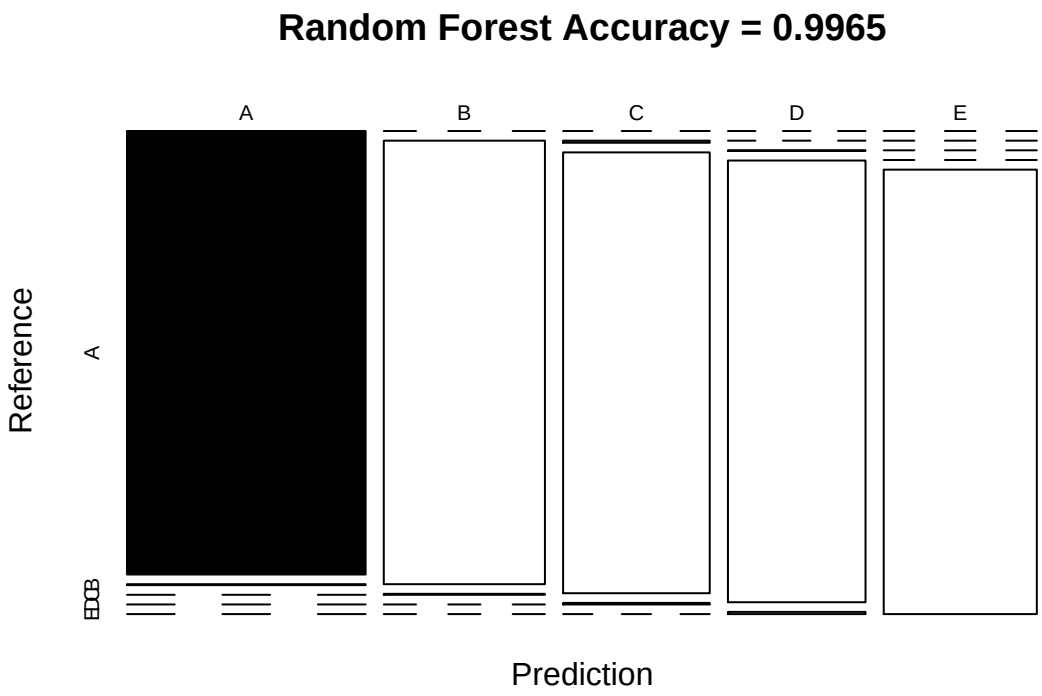
Prediction on Test Set

```
PredictRF <- predict(ModelRF, newdata=TestgSet)
confRF <- confusionMatrix(PredictRF, TestgSet$classe)
confRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  942    2    0    0
##          C    0    4  852    3    0
##          D    0    0    1  801    4
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9965
##                  95% CI : (0.9945, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9926   0.9965   0.9963   0.9956
## Specificity            0.9991   0.9995   0.9983   0.9988   1.0000
## Pos Pred Value         0.9979   0.9979   0.9919   0.9938   1.0000
## Neg Pred Value         1.0000   0.9982   0.9993   0.9993   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1921   0.1737   0.1633   0.1829
## Detection Prevalence   0.2851   0.1925   0.1752   0.1644   0.1829
## Balanced Accuracy      0.9996   0.9961   0.9974   0.9975   0.9978
```

Plot Matrix Results

```
plot(confRF$table, col = confRF$byClass,
     main = paste("Random Forest Accuracy =",
                  round(confRF$overall['Accuracy'], 4)))
```

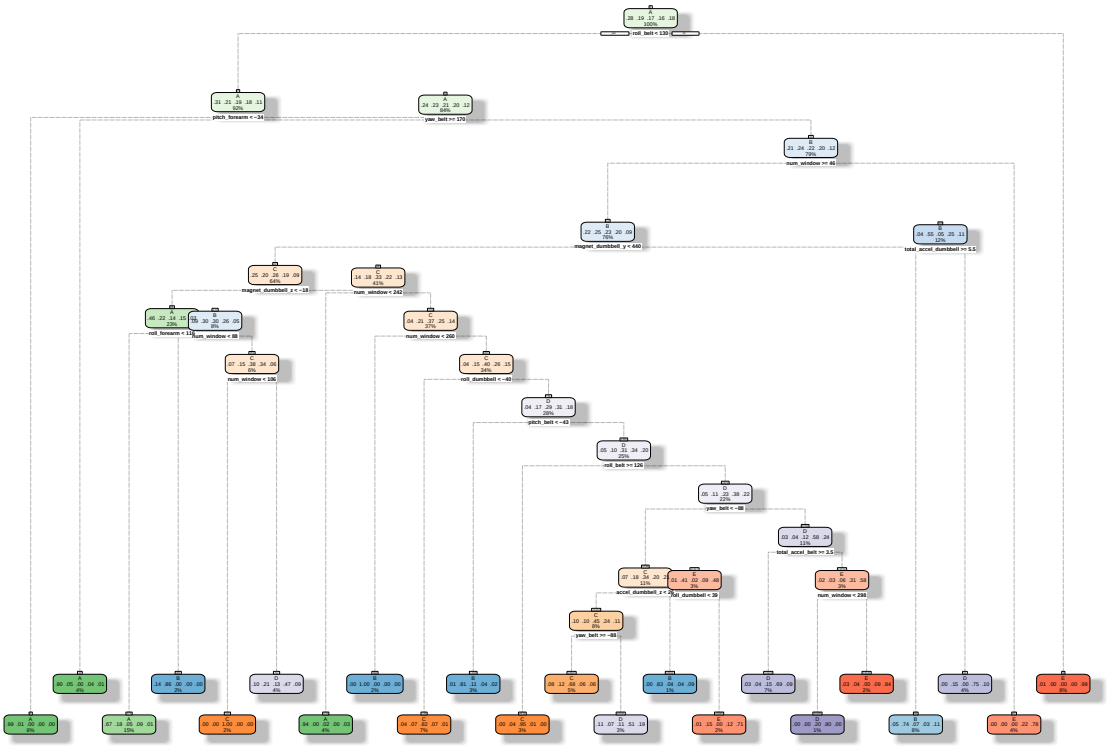


H. Decision Tree Model

Model Fit

```
set.seed(12345)
ModelDT <- rpart(classe ~ ., data=TraingSet, method="class")
fancyRpartPlot(ModelDT)
```

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



Prediction on Test Set

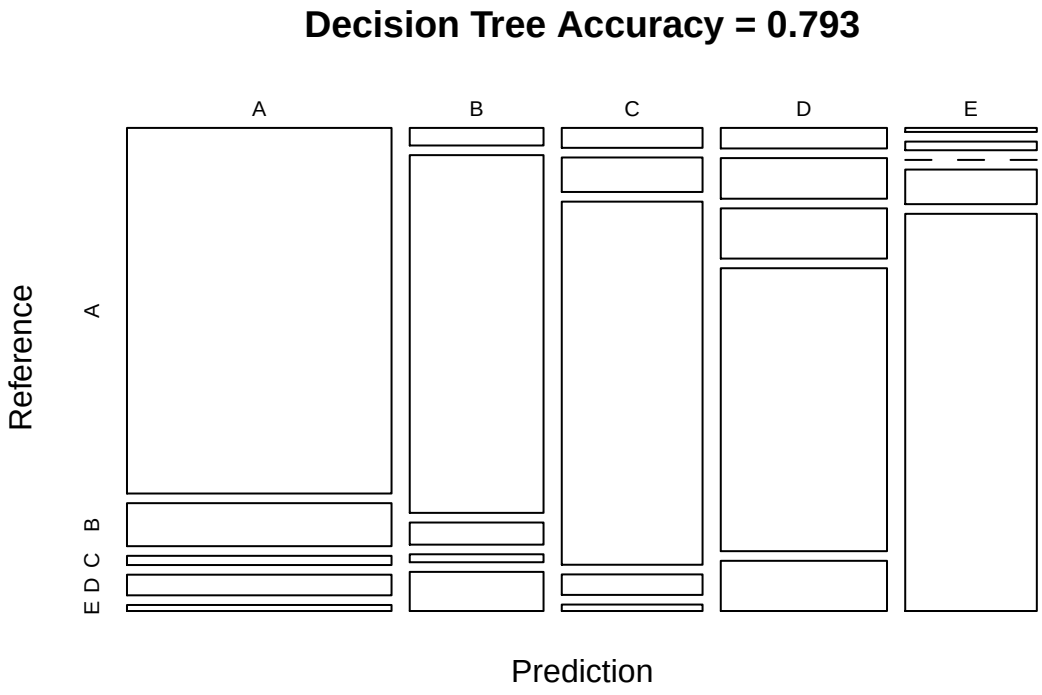
```
PredictDT <- predict(ModelDT, newdata=TestgSet, type="class")
confDT <- confusionMatrix(PredictDT, TestgSet$classe)
confDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1275  150   32   72   21
##          B   31  631   39   14   69
##          C   37   64  674   38   12
##          D   45   89  110  620  110
##          E    7   15    0   60  689
##
## Overall Statistics
##
##                Accuracy : 0.793
##                  95% CI : (0.7814, 0.8043)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7375
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140   0.6649   0.7883   0.7711   0.7647
## Specificity            0.9216   0.9613   0.9627   0.9137   0.9795
## Pos Pred Value         0.8226   0.8048   0.8170   0.6366   0.8936
## Neg Pred Value         0.9642   0.9228   0.9556   0.9532   0.9487
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2600   0.1287   0.1374   0.1264   0.1405
## Detection Prevalence   0.3161   0.1599   0.1682   0.1986   0.1572
## Balanced Accuracy      0.9178   0.8131   0.8755   0.8424   0.8721
```

Plot Matrix Results

```
plot(confDT$table, col = confDT$byClass,
     main = paste("Decision Tree Accuracy =",
                  round(confDT$overall['Accuracy'], 4)))
```



I. Generalized Boosted Model (GBM)

Model fit

```
set.seed(12345)
contrlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
ModelGBM  <- train(classe ~ ., data=TraingSet, method = "gbm",
                   trControl = contrlGBM, verbose = FALSE)
```

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

```
## Warning: package 'gbm' was built under R version 3.3.2
## Loading required package: survival
## Warning: package 'survival' was built under R version 3.3.2
##
## 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
## Warning: package 'plyr' was built under R version 3.3.2
```

```
ModelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 46 had non-zero influence.
```

Prediction on Test Set

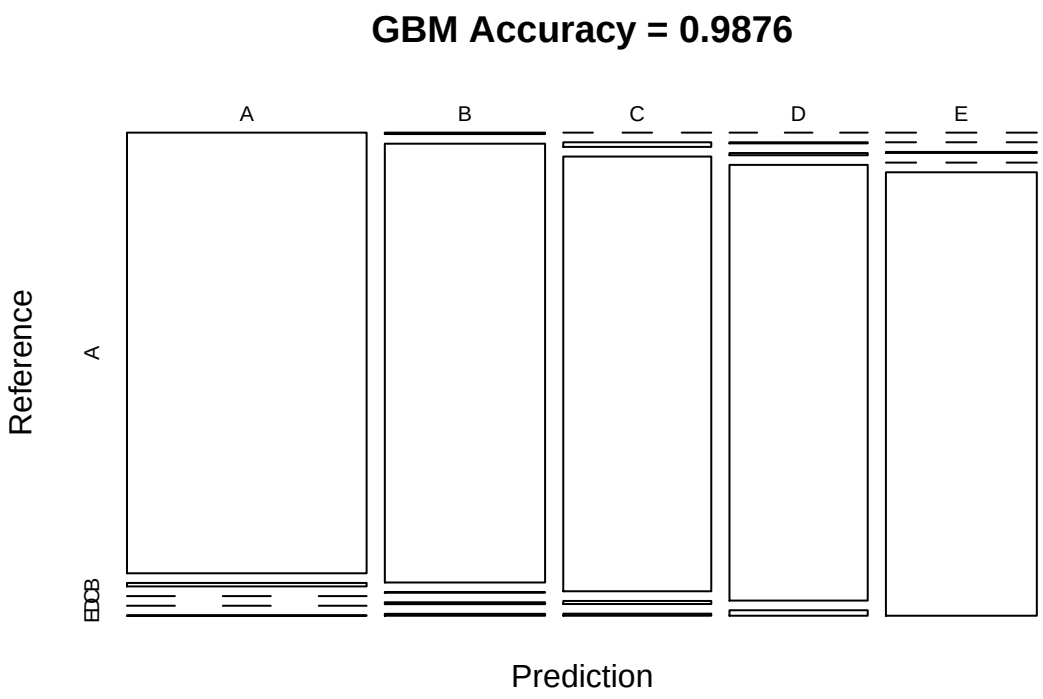
```
predictGBM <- predict(ModelGBM, newdata=TestgSet)
confGBM <- confusionMatrix(predictGBM, TestgSet$classe)
confGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392   11    0    0    1
##          B    3  927    1    4    4
##          C    0    9  848    6    4
##          D    0    2    4  794   10
##          E    0    0    2    0  882
##
## Overall Statistics
##
##                Accuracy : 0.9876
##                  95% CI : (0.9841, 0.9905)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9843
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9768   0.9918   0.9876   0.9789
## Specificity            0.9966   0.9970   0.9953   0.9961   0.9995
## Pos Pred Value         0.9915   0.9872   0.9781   0.9802   0.9977
```

```
## Neg Pred Value         0.9991   0.9945   0.9983   0.9976   0.9953
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1890   0.1729   0.1619   0.1799
## Detection Prevalence   0.2863   0.1915   0.1768   0.1652   0.1803
## Balanced Accuracy      0.9972   0.9869   0.9936   0.9918   0.9892
```

Plot Matrix Results

```
plot(confGBM$table, col = confGBM$byClass,
     main = paste("GBM Accuracy =", round(confGBM$overall['Accuracy'], 4)))
```



J. Summary

The accuracy of the 3 regression models above as following:

```
a.Random Forest: 0.9965
b.Decision Tree: 0.793
c.GBM: 0.9876
```

In summary, the accuracy for Random Forest model was 0.9965 (95% CI: (0.9945, 0.998)) compared to 0.793 (95% CI:(0.7814, 0.8043)) for Decision Tree model. Even though, Generalized Boosted Model (GBM) was 0.9876 (95% CI: (0.9841, 0.9905)) in which closly match to Random Forest accuracy, but the Random Forest algorithm performed better results than other 2 models. Therefore, we selected Random Forest Model with 99% validation for our test data set to comprises the 20 cases.

K. Using the Random Forest algorithm model to predict the 20 quiz results (testing dataset) as shown below:

```
PredictTestng <- predict(ModelRF, newdata=testing)
PredictTestng
```

```
##  [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
```