Machine Learning Final Project

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. The goal of this project is 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).

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Data

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

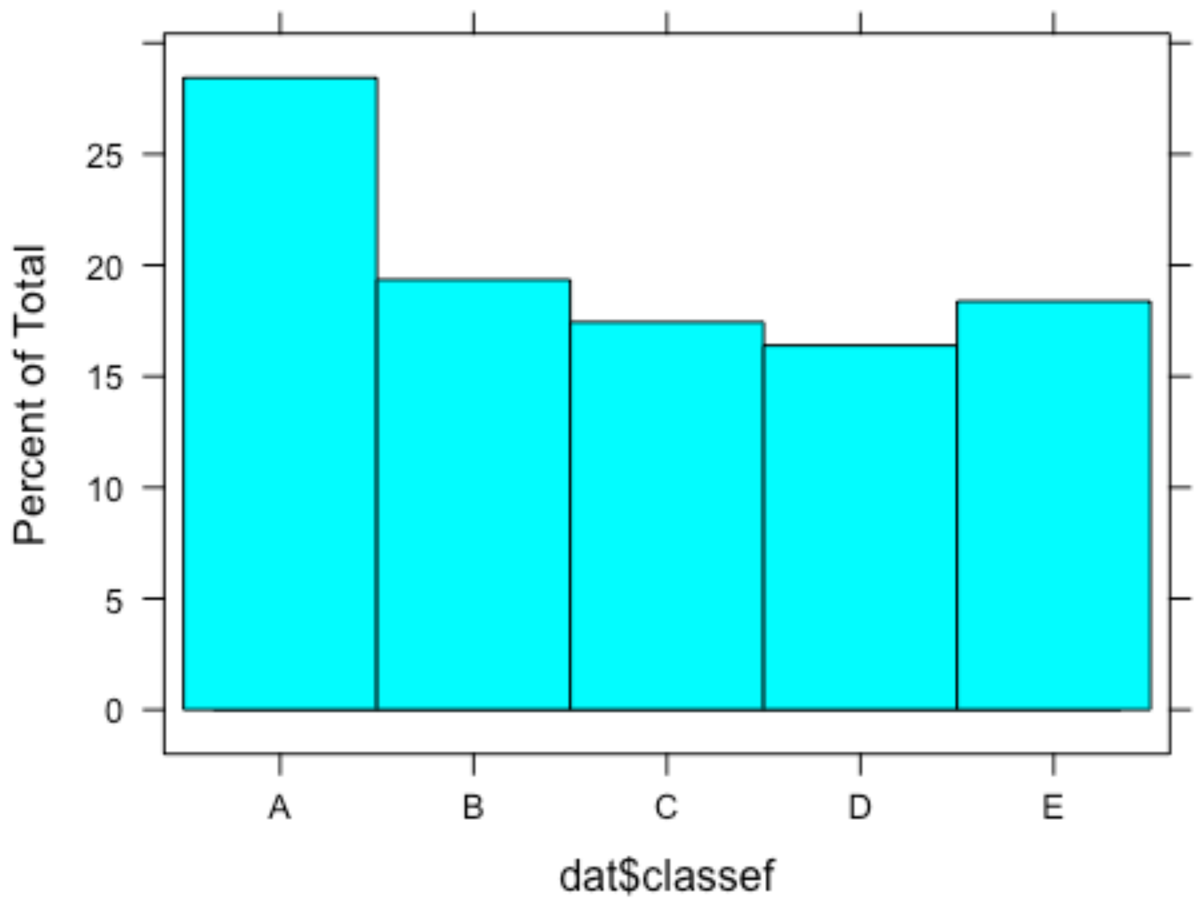
The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. ##The R Code and Results

Libraries and Data Import:

```
library(randomForest)

## randomForest 4.6-14

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

library(gbm)

## Loaded gbm 2.1.5

library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(kernlab)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha

## The following object is masked from 'package:randomForest':
##
##     margin

library(corrplot)

## corrplot 0.84 loaded

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.

##
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':
##
##     importance
```

```
library(rpart)
library(rmarkdown)
library(knitr)

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"

dat <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
datTest <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

#Descriptive Statistics of the classe variable (and factorization)
dat$classef <- factor(dat$classe)

## Warning in histogram.factor(dat$classef, dat): explicit 'data'
## specification ignored
```



Data Preparation Steps

Here I clean and preprocess the data. I remove the the varables that have no logical relationship to classe and also those with low correlation or non-zero variance.

```
#Data Cleaning and Preprocess

#Remove Date and Name Vars as these should not impact Classe
trainData <- dat[, -c(1:7)]

#Partition training dataset into initial training and testing

set.seed(1234)
inTrain <- createDataPartition(trainData$classef, p = 0.7, list =
FALSE)
trainData <- trainData[inTrain, ]
testData <- trainData[-inTrain, ]
dim(trainData)

## [1] 13737   154

#Removing the variables with low variance to help improve prediction

nsv<-nearZeroVar(trainData)

trainData <- trainData[, -nsv]
testData  <- testData[, -nsv]
dim(trainData)

## [1] 13737   122

#Clean out all variables without data
trainData<- trainData[, colSums(is.na(trainData)) == 0]
testData<- testData[, colSums(is.na(testData)) == 0]
dim(trainData)

## [1] 13737    54
```

Moving on to Model Building

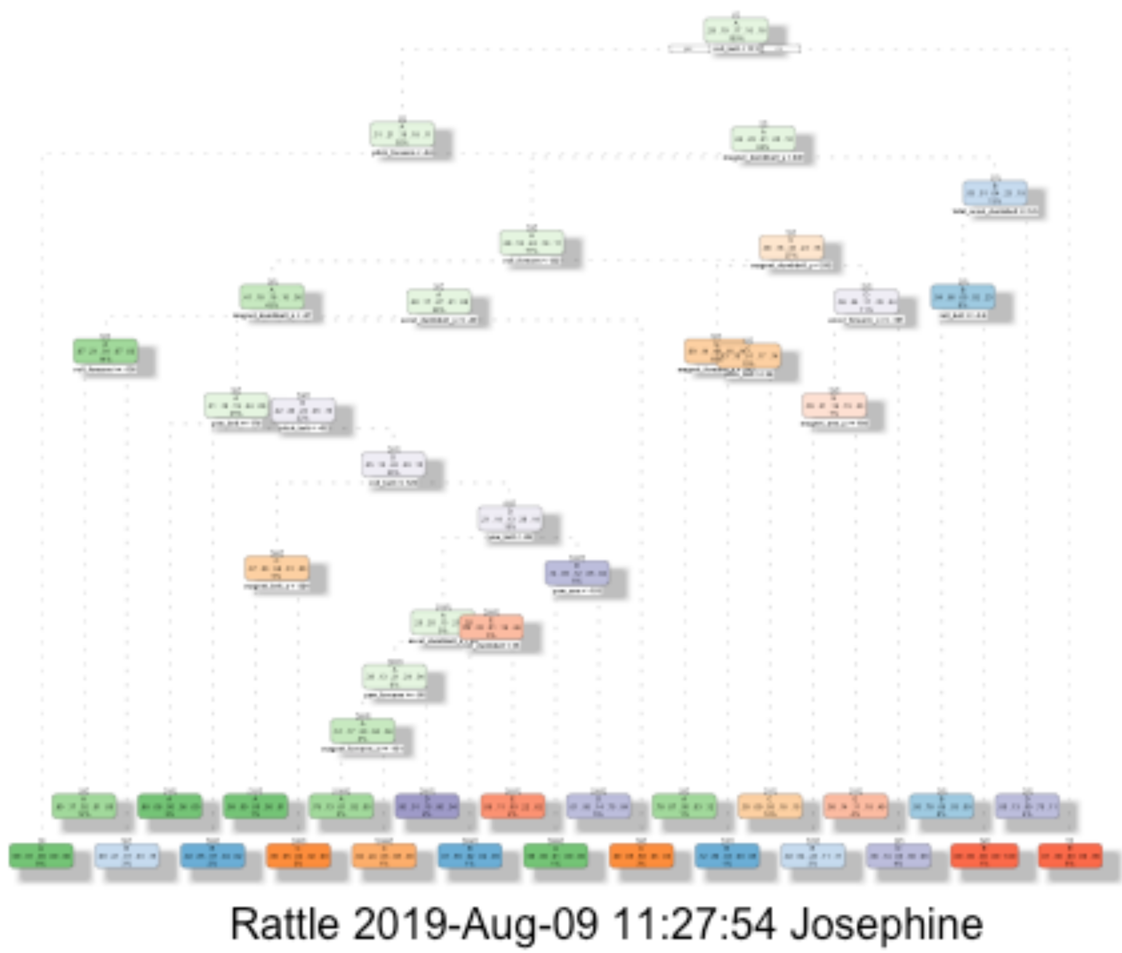
Here I try three types of models to see which will have the best predictive value. The first is Decision Tree, the second is GBM and the third is Random Forest.

Decision Tree

```
#Trying Classification Trees Model

set.seed(12345)
decisionTreeMod <- rpart(classef ~ ., data=trainData[,-53],
```

```
method="class")
fancyRpartPlot(decisionTreeMod)
```

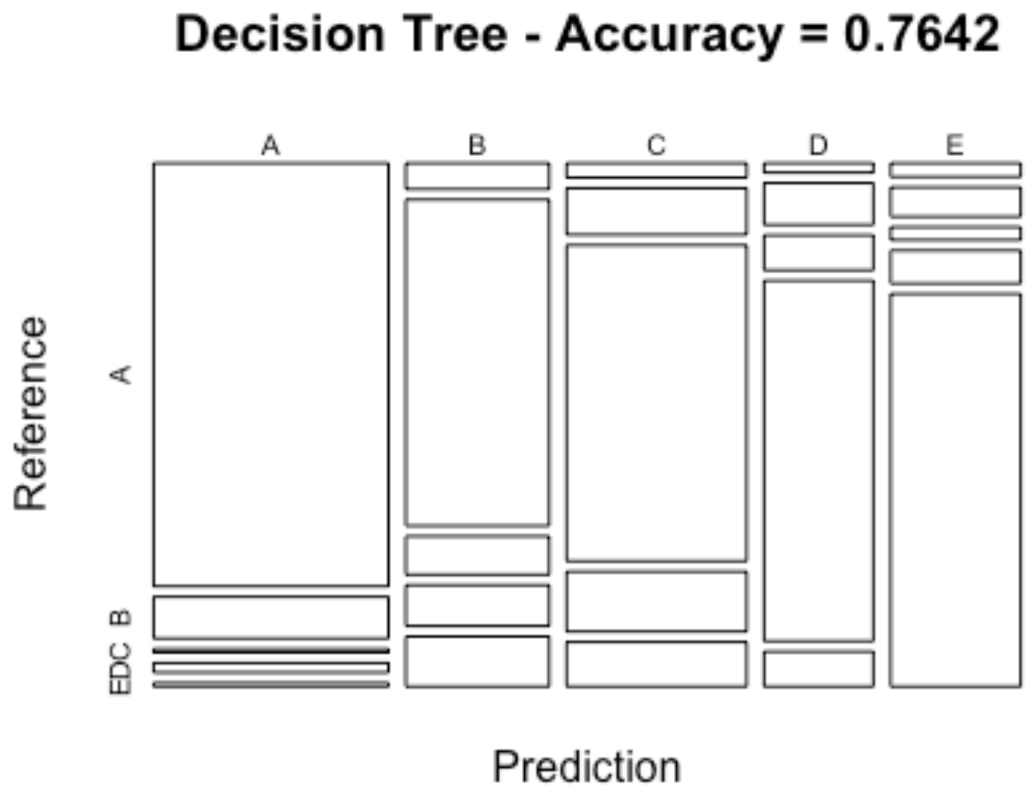
```
## Warning: labs do not fit even at cex 0.15, there may be some
overplotting
```




```
#Look at most influential variables
#Top 4:   roll_belt        accel_belt_z          pitch_belt
pitch_forearm


#Confusion Matrix - Looks at the predictive value of the model.
Calculates Sens, Spec Etc..
# Here we use the model built on the training set to predict the class
for test set data
#Sensitivity was highest for Class A and C

predictTreeMod <- predict(decisionTreeMod, testData, type = "class")
cmtree <- confusionMatrix(predictTreeMod, testData$classe)
cmtree
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1067  105    9   24    9
##          B   40  502   59   63   77
##          C   28   90  611  116   86
##          D   11   49   41  423   41
##          E   19   41   18   46  548
## 
## Overall Statistics
##                                           
##                Accuracy : 0.7642          
##                  95% CI : (0.751, 0.7771)
##     No Information Rate : 0.2826          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.7015          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9159   0.6379   0.8279   0.6295   0.7201
## Specificity            0.9503   0.9284   0.9055   0.9589   0.9631
## Pos Pred Value         0.8789   0.6775   0.6563   0.7487   0.8155
## Neg Pred Value         0.9663   0.9157   0.9602   0.9300   0.9383
## Prevalence             0.2826   0.1909   0.1790   0.1630   0.1846
## Detection Rate         0.2588   0.1218   0.1482   0.1026   0.1329
## Detection Prevalence   0.2944   0.1797   0.2258   0.1370   0.1630
## Balanced Accuracy      0.9331   0.7831   0.8667   0.7942   0.8416
```

```r
#Plot of the Accuracy of the Model to Predict the Classe Variable in
the Test Set
#Overall Accuracy is 76%, which is not bad IMO.
#Out of Sample Error is 24%

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

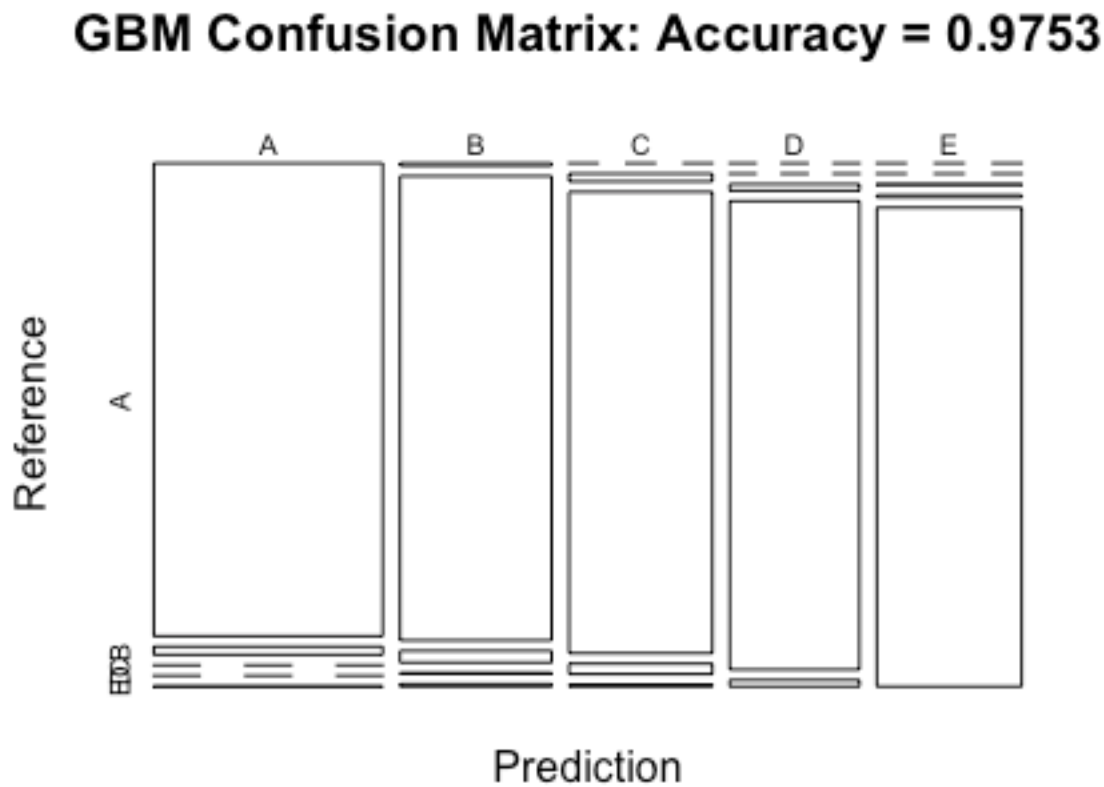


For Decision Tree, the overall accurary was 76% and the out of sample error was 24%(.24). Let's compare to other types of models:

GBM

```
#Try GBM
#
d<-trainData [,-53]
set.seed(1234)
cGM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modelGM  <- train(classef ~ ., data=d, method = "gbm", trControl = cGM,
verbose = FALSE)
modelGM$finalModel

## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.

#A gradient boosted model with multinomial loss function.
#150 iterations were performed.
```

```r
#There were 52 predictors of which 52 had non-zero influence.

predictGM <- predict(modelGM, newdata=testData[,-53])
cmGM <- confusionMatrix(predictGM, testData$classef)
cmGM

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1161   20    0    0    1
##          B    4  755   19    1    4
##          C    0   12  709   16    3
##          D    0    0    9  652    9
##          E    0    0    1    3  744
## 
## Overall Statistics
## 
##                Accuracy : 0.9753
##                  95% CI : (0.97, 0.9798)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9687
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9966   0.9593   0.9607   0.9702   0.9777
## Specificity            0.9929   0.9916   0.9908   0.9948   0.9988
## Pos Pred Value         0.9822   0.9642   0.9581   0.9731   0.9947
## Neg Pred Value         0.9986   0.9904   0.9914   0.9942   0.9950
## Prevalence             0.2826   0.1909   0.1790   0.1630   0.1846
## Detection Rate         0.2816   0.1831   0.1720   0.1581   0.1805
## Detection Prevalence   0.2867   0.1899   0.1795   0.1625   0.1814
## Balanced Accuracy      0.9947   0.9755   0.9758   0.9825   0.9882

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




```
#Accuracy of 98%, slightly lower than RF Model, but still very good
result
```

Accuracy of 98% is much better than Decision Tree Model.

Let's try one more for kicks:

Random Forest Model

```
RF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modelRF <- train(classef ~ ., data=d, method="rf", trControl=RF)
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: 0.75%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    3    1    0    1 0.001280082
## B   18 2632    7    1    0 0.009781791
## C    0   19 2368    9    0 0.011686144
## D    0    1   26 2224    1 0.012433393
## E    0    2    6    8 2509 0.006336634
```

```
summary (modelRF)
```

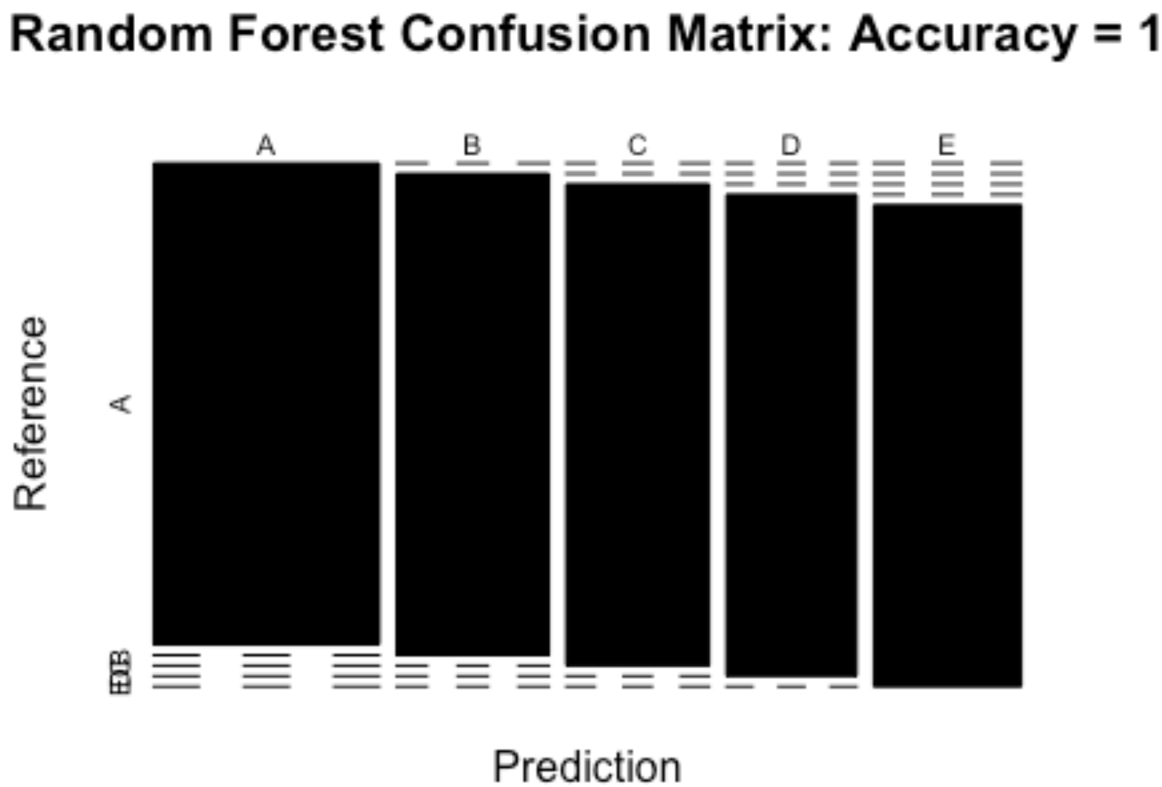
```
##                 Length Class      Mode
## call                4  -none-     call
## type                1  -none-     character
## predicted       13737  factor     numeric
## err.rate         3000  -none-     numeric
## confusion          30  -none-     numeric
## votes           68685  matrix     numeric
## oob.times       13737  -none-     numeric
## classes             5  -none-     character
## importance         52  -none-     numeric
## importanceSD        0  -none-     NULL
## localImportance     0  -none-     NULL
## proximity           0  -none-     NULL
## ntree               1  -none-     numeric
## mtry                1  -none-     numeric
## forest             14  -none-     list
## y               13737  factor     numeric
## test                0  -none-     NULL
## inbag               0  -none-     NULL
## xNames             52  -none-     character
## problemType         1  -none-     character
## tuneValue           1  data.frame list
## obsLevels           5  -none-     character
## param               0  -none-     list
```

```
#Use the RF model to predict the test data
#Perfect Sens/Spec/Accuracy- This might be overfitting?? Obviously a
better model then decision tree.


predictRF <- predict(modelRF, newdata=testData[,-53])
cmrf <- confusionMatrix(predictRF, testData$classef)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1165    0    0    0    0
##          B    0  787    0    0    0
```

```
##          C    0    0  738    0    0
##          D    0    0    0  672    0
##          E    0    0    0    0  761
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000    1.000    1.000   1.0000
## Specificity            1.0000   1.0000    1.000    1.000   1.0000
## Pos Pred Value         1.0000   1.0000    1.000    1.000   1.0000
## Neg Pred Value         1.0000   1.0000    1.000    1.000   1.0000
## Prevalence             0.2826   0.1909    0.179    0.163   0.1846
## Detection Rate         0.2826   0.1909    0.179    0.163   0.1846
## Detection Prevalence   0.2826   0.1909    0.179    0.163   0.1846
## Balanced Accuracy      1.0000   1.0000    1.000    1.000   1.0000
```

```
#Really nice plot looking at accuracy in test set. Very high predictive
value
plot(cmrf$table, col = cmrf$byClass, main = paste("Random Forest
Confusion Matrix: Accuracy =", round(cmrf$overall['Accuracy'], 4)))
```



The RF model has perfect Sens/Spec/Accuracy- This might be overfitting, but it is an improvement over decision tree and GBM.

I will use the RF model for the quiz. But, I will note that both the GBM and the RF had the same predictions for the final test dataset:

```
ResultsRF <- predict(modelRF, newdata=datTest)
ResultsRF

##  [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
```