# Course 8 Project

Kaushik Sivasankaran

6/25/2021

## Executive Summary

In this project, we aim to use the data from ccelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which did the exercise. We will use the "classe" variable in the training set as the main predictor to do to so. But during the exericse we will explore other viable predictors in the training dataset as well.

## Inital Preperation

```
# free up memory

rm(list = ls())

# set working directory

setwd("D:/Users/kaushik.sivasankaran/Desktop/R/Course 8/Course-8-Project")

# load libraries

library(knitr)
library(caret)
```

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

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

```
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 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
```

```
library(RColorBrewer)
library(corrplot)
```

```
## corrplot 0.88 loaded
```

Then let us set the seed to ensure reproducibility.

```
set.seed(12345)
```

# Loading and Cleansing the Data

## Loading the Data

```
# set the URL for the download
UrlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the datasets
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))

dim(training)
```

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

```
dim(testing)
```

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

## Cleaning the Data

Having a quick look at the data, there are a lot of columns with nearly zero variability. Alsom there are NA values in many of the variables, which can be removed with the following data cleansing process.

```
# removing valriables with Nearly Zero Variance

NZV <- nearZeroVar(training)
training <- training[, -NZV]
testing  <- testing[, -NZV]

dim(training)
```

```
## [1] 19622   100
```

```
dim(testing)
```

```
## [1]  20 100
```

This has now reduced the number of variables from 160 to 100.

```
# remove variables that are mostly NA

AllNA    <- sapply(training, function(x) mean(is.na(x))) > 0.95
training <- training[, AllNA==FALSE]
testing  <- testing[, AllNA==FALSE]

dim(training)
```

```
## [1] 19622    59
```

```
dim(testing)
```

```
## [1] 20 59
```

Further cleaning has now reduced the required variables to 59.

Next, it looks like the first 6 variables don't add much value towards our model. Hence, we will remove those variables.

```
training <- training[,7:59]

testing <- testing[,7:59]

dim(training)
```

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

```
dim(testing)
```

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

## Creating Data Partitions

Now that the data is fairly clean, we can go ahead and create the data partition

```
# create a partition with the training dataset

inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)

TrainSet <- training[inTrain,]

TestSet  <- training[-inTrain,]

dim(TrainSet)
```

```
## [1] 13737    53
```

```
dim(TestSet)
```

```
## [1] 5885   53
```

## Model Building For Prediction

We will be using 3 methods to model the predictions from the train dataset. The methods are: Random Forests, Decision Tree, and Generelized Boosted Method. The one with the highest accuracy when applied to the test dataset will be used as the final model fit.

### i. Random Forest

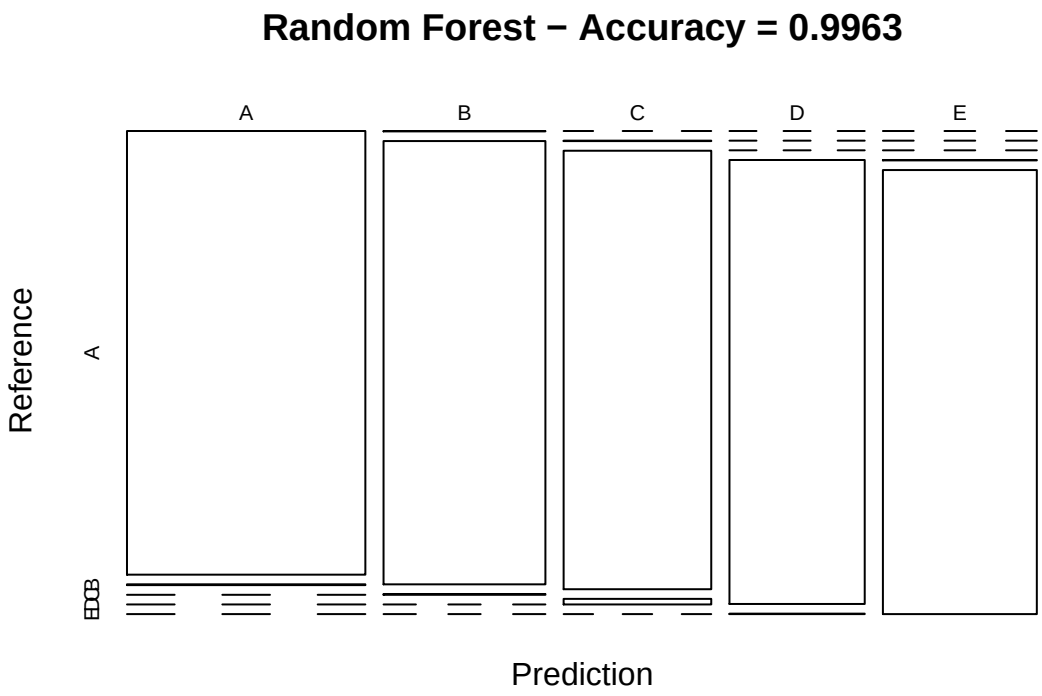
```
# model fit
set.seed(12345)

modFitRandForest <- randomForest(classe ~ ., data = TrainSet)


# 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    3    0    0    0
##          B    1 1135    2    0    0
##          C    0    1 1024   13    0
##          D    0    0    0  950    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9943, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9953
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   0.9981   0.9855   0.9991
## Specificity            0.9993   0.9994   0.9971   0.9998   0.9998
## Pos Pred Value         0.9982   0.9974   0.9865   0.9989   0.9991
## Neg Pred Value         0.9998   0.9992   0.9996   0.9972   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1740   0.1614   0.1837
## Detection Prevalence   0.2848   0.1934   0.1764   0.1616   0.1839
## Balanced Accuracy      0.9993   0.9979   0.9976   0.9926   0.9994
```

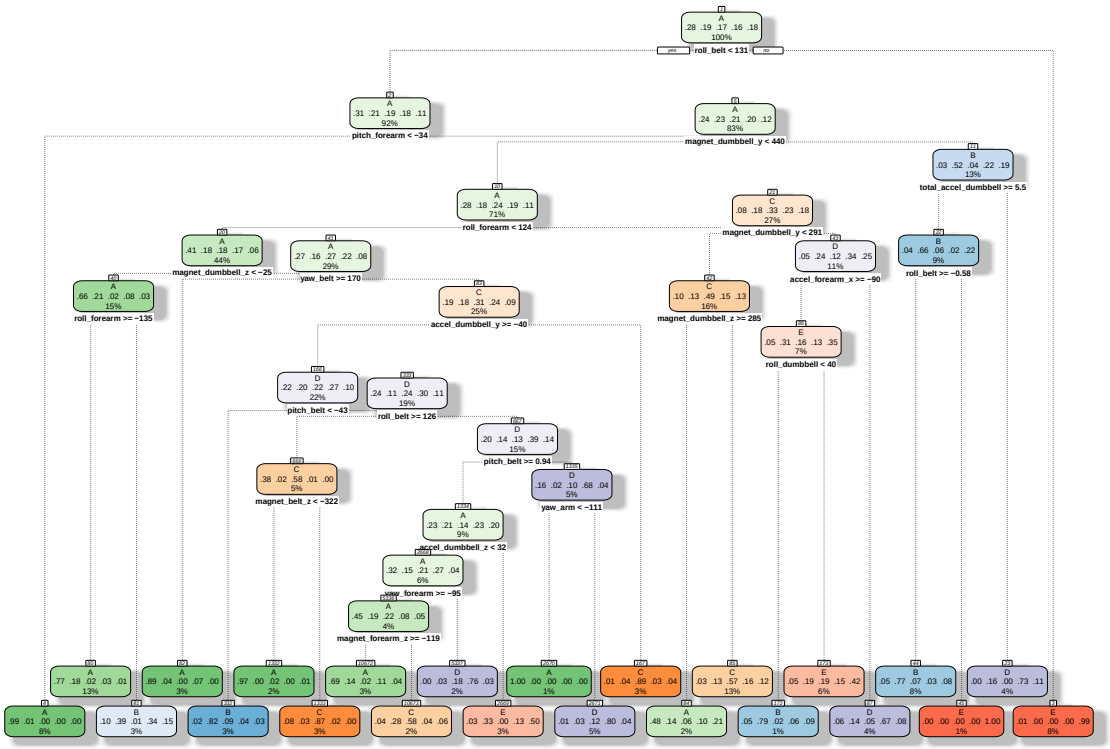
```
# plot matrix results
plot(confMatRandForest$table, col = confMatRandForest$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(confMatRandForest$overall['Accuracy'], 4)))
```




### ii. Decision Trees

```
# model fit
set.seed(12345)
modFitDecTree <- rpart(classe ~ ., data = TrainSet, method = "class")

fancyRpartPlot(modFitDecTree)
```



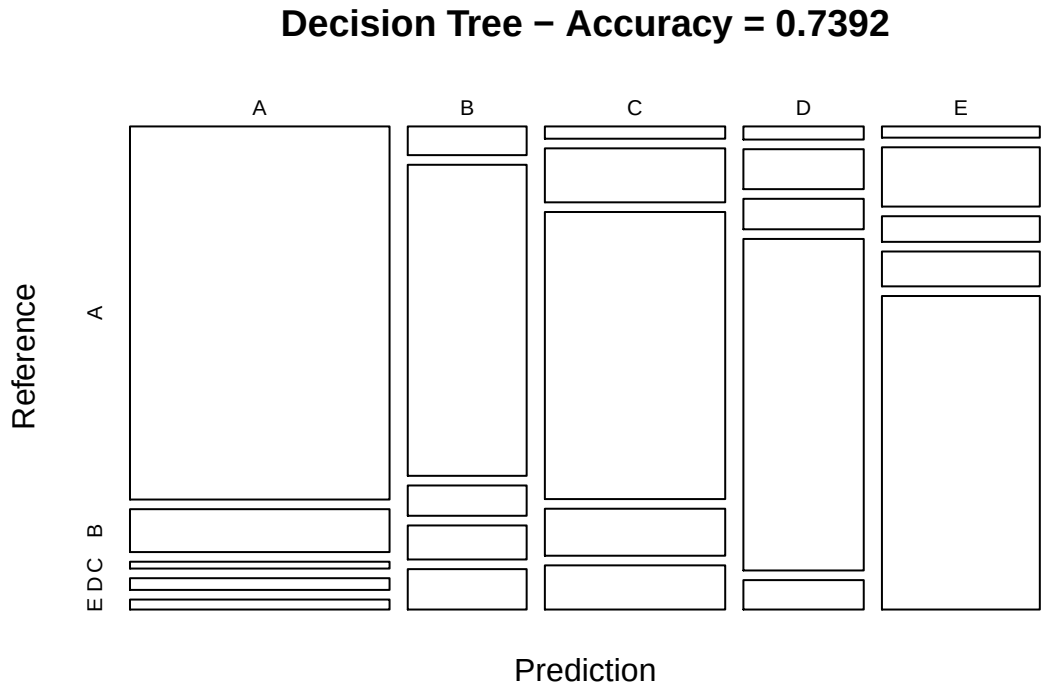

Rattle 2021−Jun−26 12:24:25 kaushik.sivasankaran

```
# 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 1532  176   28   48   41
##          B   54  585   57   64   76
##          C   35  154  819  134  126
##          D   25   76   58  631   56
##          E   28  148   64   87  783
##
## Overall Statistics
##
##                Accuracy : 0.7392
##                  95% CI : (0.7277, 0.7503)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6692
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9152  0.51361   0.7982   0.6546   0.7237
## Specificity            0.9304  0.94711   0.9076   0.9563   0.9319
## Pos Pred Value         0.8395  0.69976   0.6459   0.7459   0.7054
## Neg Pred Value         0.9650  0.89028   0.9552   0.9339   0.9374
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2603  0.09941   0.1392   0.1072   0.1331
## Detection Prevalence   0.3101  0.14206   0.2155   0.1438   0.1886
## Balanced Accuracy      0.9228  0.73036   0.8529   0.8054   0.8278
```

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




### iii. Generalized Boosted Model

```r
# model fit
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=TrainSet, method = "gbm",
                    trControl = controlGBM, verbose = FALSE)

modFitGBM$finalModel
```

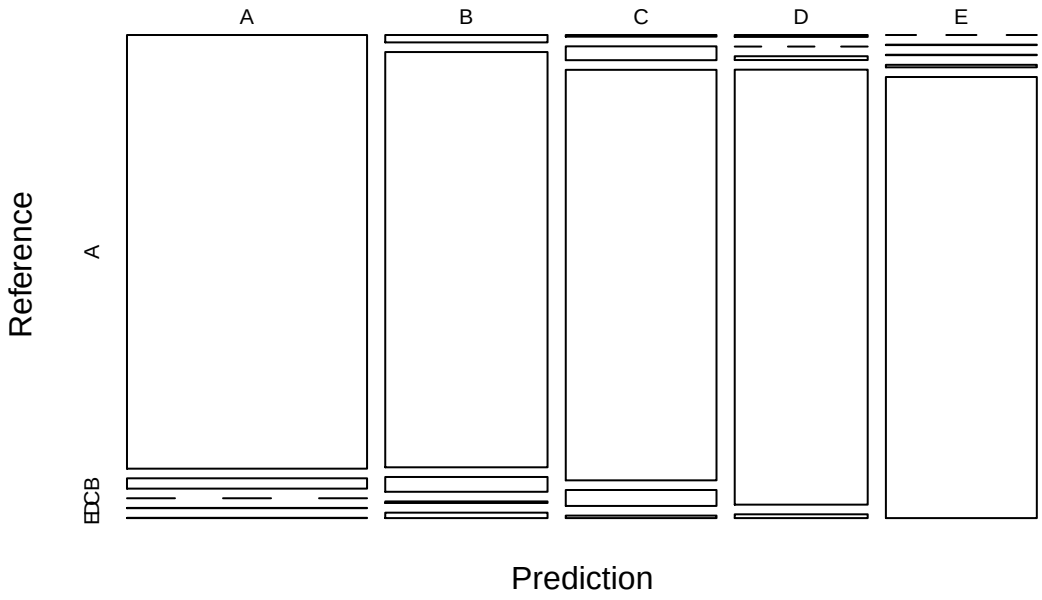
```
## A gradient boosted model with multinomial loss function.
```

```
## 150 iterations were performed.
## There were 52 predictors of which 51 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 1647   39    0    1    1
##          B   19 1066   38    4   14
##          C    4   33  979   38    6
##          D    4    0    8  915    8
##          E    0    1    1    6 1053
##
## Overall Statistics
##
##                Accuracy : 0.9618
##                  95% CI : (0.9565, 0.9665)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9516
##
##  Mcnemar's Test P-Value : 8.329e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9359   0.9542   0.9492   0.9732
## Specificity            0.9903   0.9842   0.9833   0.9959   0.9983
## Pos Pred Value         0.9757   0.9343   0.9236   0.9786   0.9925
## Neg Pred Value         0.9936   0.9846   0.9903   0.9901   0.9940
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1811   0.1664   0.1555   0.1789
## Detection Prevalence   0.2868   0.1939   0.1801   0.1589   0.1803
## Balanced Accuracy      0.9871   0.9601   0.9688   0.9726   0.9858
```

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




#Applying the Selected Model to the Test Data

The accuracy of the 3 regression modeling methods above are: a. Random Forest : 0.9963 b. Decision Tree : 0.7392 c. GBM : 0.9618

The Random Forest model has the best model fit since its accuracy is better than the other two models. Hence, the Random Forest model will be applied to predict the testing dataset.

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

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```