Final Project, Practical Machine Learning

Sayyed Mohsen Vazirizade

January 20, 2019

Download the Required files and libraries

```
#Adding the required libraries
rm(list = setdiff(ls(), lsf.str()))
wants <- c("caret", "ggplot2","corrplot","rpart","rpart.plot","RColorBrewer","rattle","randomForest")
has   <- wants %in% rownames(installed.packages())
if(any(!has)) install.packages(wants[!has])
for (pkg in wants) {library(pkg, character.only = TRUE)}
```

```
## Warning: package 'caret' was built under R version 3.5.1

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.1

## Warning: package 'corrplot' was built under R version 3.5.2

## corrplot 0.84 loaded

## Warning: package 'rpart.plot' was built under R version 3.5.2

## Warning: package 'rattle' was built under R version 3.5.2

## 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.

## Warning: package 'randomForest' was built under R version 3.5.2

## 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
```

Downloading the Data and import it

```
#Downloading the Data and import it
destfile1 <- "TrainingData.csv"
destfile2 <- "TestData.csv"
URLAddress1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
URLAddress2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if (!file.exists(destfile1)) {download.file(URLAddress1,destfile1, mode='wb')}
```

```r
if (!file.exists(destfile2)) {download.file(URLAddress2,destfile2, mode='wb')}
TrainData <- read.csv(destfile1, header = TRUE)
TestData <- read.csv(destfile2, header = TRUE)
dim(TrainData)
```

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

removing the columns they have missing data

```r
#removing the columns they have missing data
TrainDataC <- TrainData[,complete.cases(t(TrainData))]
dim(TrainDataC)
```

```
## [1] 19622    93
```

```r
TrainDataCC <-  TrainDataC[,-nearZeroVar(TrainDataC)]
dim(TrainDataCC)
```

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

```r
TrainDataCC <-  TrainDataCC[,-c(1,2,3,4,5)]
dim(TrainDataCC)
```

```
## [1] 19622    54
```

```r
TestDataC  <- TestData[,complete.cases(t(TrainData))]
TestDataCC <-  TestDataC[,-nearZeroVar(TrainDataC)]
TestDataCC <-  TestDataCC[,-c(1,2,3,4,5)]
dim(TestDataCC)
```

```
## [1] 20 54
```

By removing the columns having at least one missing data, the number of the variables are reduced from 160 to 93

Seperating data for Training and Testing

```r
#Seperating data for Training and Testing
set.seed(1)
inTrain <- createDataPartition(TrainDataCC$classe, p = 0.7, list = FALSE)
TrainDataCCTrain <- TrainDataCC[inTrain, ]
TrainDataCCTest <- TrainDataCC[-inTrain, ]
dim(TrainDataCCTrain)
```

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

```r
dim(TrainDataCCTest)
```

```
## [1] 5885   54
```

The number of data sets in Training and Testing are 13737 to 5885, respectively.

Correlation Matirx

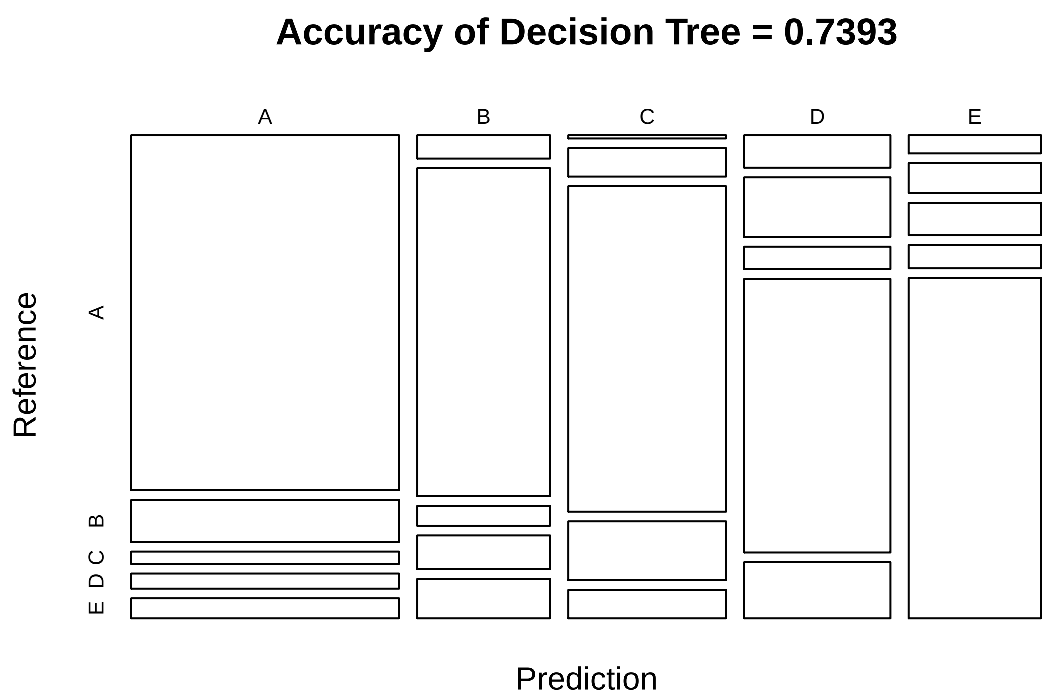
```r
#Correlation Matirx
#corMatrix <- cor(TrainDataCCTrain[,-c(54)])
#corrplot(corMatrix, order = "FPC", method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0, 0,
```

Method 1: Classifaction Tree

```
set.seed(1)
Method1 <- rpart(classe ~ ., data=TrainDataCCTrain, method="class")
#Method1 <- train(classe ~ ., method="rpart", data=TrainDataCCTrain)
#fancyRpartPlot(Method1)
#or rpart.plot(Method1)
predictMethod1 <- predict(Method1, TrainDataCCTest, type = "class")
cmTree <- confusionMatrix(predictMethod1, TrainDataCCTest$classe)
cmTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1504  178   52   64   85
##          B   49  689   42   71   83
##          C    8   71  812  147   71
##          D   75  138   52  633  130
##          E   38   63   68   49  713
##
## Overall Statistics
##
##                Accuracy : 0.7393
##                  95% CI : (0.7279, 0.7505)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6691
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8984   0.6049   0.7914   0.6566   0.6590
## Specificity            0.9100   0.9484   0.9389   0.9197   0.9546
## Pos Pred Value         0.7987   0.7377   0.7322   0.6158   0.7658
## Neg Pred Value         0.9575   0.9091   0.9552   0.9319   0.9255
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2556   0.1171   0.1380   0.1076   0.1212
## Detection Prevalence   0.3200   0.1587   0.1884   0.1747   0.1582
## Balanced Accuracy      0.9042   0.7766   0.8651   0.7882   0.8068
```

```
plot(cmTree$table, col = cmTree$byClass, main = paste("Accuracy of Decision Tree =", round(cmTree$overal
```



Method 2: Random Forest

```
#Method 2: Random Forest
set.seed(1)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
Method2 <- randomForest(classe ~ ., data=TrainDataCCTrain, proximity=TRUE)
#Method2 <- train(classe ~ ., data=TrainDataCCTrain, method="rf", trControl=controlRF)
Method2$finalModel
```

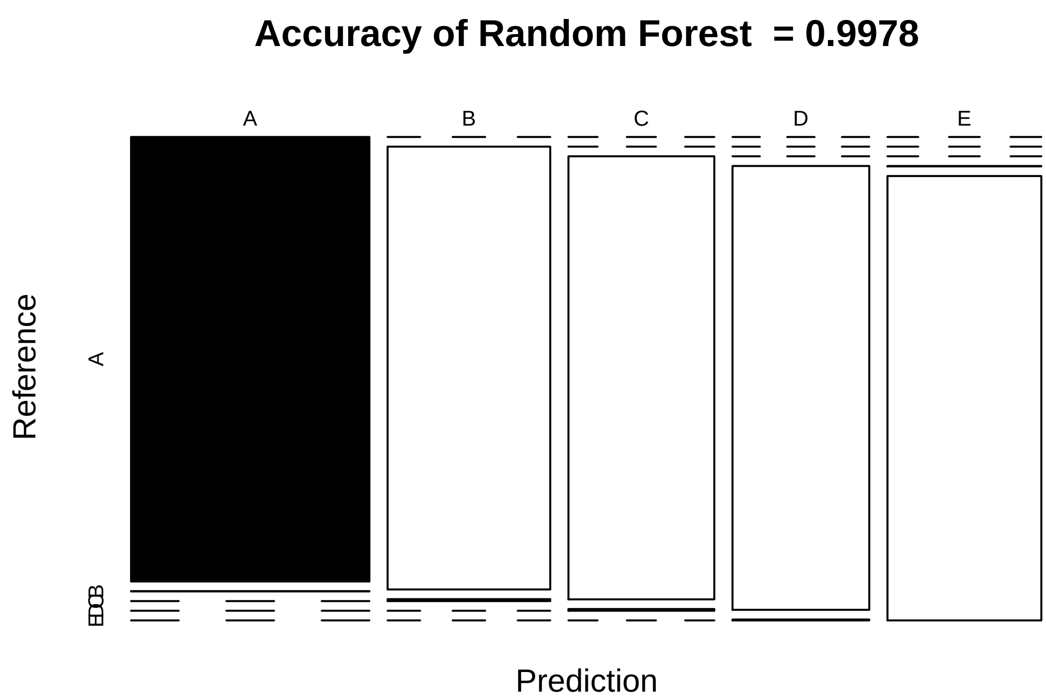
```
## NULL
```

```
predictMethod2 <- predict(Method2, TrainDataCCTest)
cmRF<- confusionMatrix(predictMethod2, TrainDataCCTest$classe)
cmRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    5    0    0
##          C    0    0 1021    4    0
##          D    0    0    0  959    2
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9978
```

```
##                    95% CI : (0.9962, 0.9988)
##       No Information Rate : 0.2845
##       P-Value [Acc > NIR] : < 2.2e-16
##
##                     Kappa : 0.9972
##    Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9951   0.9948   0.9982
## Specificity            0.9998   0.9989   0.9992   0.9996   0.9998
## Pos Pred Value         0.9994   0.9956   0.9961   0.9979   0.9991
## Neg Pred Value         1.0000   0.9998   0.9990   0.9990   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1735   0.1630   0.1835
## Detection Prevalence   0.2846   0.1942   0.1742   0.1633   0.1837
## Balanced Accuracy      0.9999   0.9990   0.9972   0.9972   0.9990
```

```
plot(cmRF$table, col = cmRF$byClass, main = paste("Accuracy of Random Forest  =", round(cmRF$overall['A
```




```
MostImpVars <- varImp(Method2)
MostImpVars
```

```
##                          Overall
## num_window             933.71962
## roll_belt              799.22199
```

```
## pitch_belt           457.39288
## yaw_belt             558.87646
## total_accel_belt     151.68997
## gyros_belt_x          63.52391
## gyros_belt_y          72.97806
## gyros_belt_z         194.11986
## accel_belt_x          86.74478
## accel_belt_y          81.96092
## accel_belt_z         263.95850
## magnet_belt_x        170.38336
## magnet_belt_y        264.05404
## magnet_belt_z        247.01068
## roll_arm             208.78982
## pitch_arm            108.17074
## yaw_arm              136.57721
## total_accel_arm       63.88403
## gyros_arm_x           78.39387
## gyros_arm_y           80.57290
## gyros_arm_z           33.93297
## accel_arm_x          155.47032
## accel_arm_y           96.24702
## accel_arm_z           80.18304
## magnet_arm_x         166.74962
## magnet_arm_y         137.72446
## magnet_arm_z         103.93766
## roll_dumbbell        280.87273
## pitch_dumbbell       124.13985
## yaw_dumbbell         168.64981
## total_accel_dumbbell 183.28356
## gyros_dumbbell_x      74.77004
## gyros_dumbbell_y     138.42671
## gyros_dumbbell_z      47.89293
## accel_dumbbell_x     176.44076
## accel_dumbbell_y     278.27883
## accel_dumbbell_z     218.33235
## magnet_dumbbell_x    320.23137
## magnet_dumbbell_y    439.23892
## magnet_dumbbell_z    490.03350
## roll_forearm         377.85168
## pitch_forearm        495.11688
## yaw_forearm          108.02224
## total_accel_forearm   61.95215
## gyros_forearm_x       45.82796
## gyros_forearm_y       73.82459
## gyros_forearm_z       50.58596
## accel_forearm_x      211.18437
## accel_forearm_y       86.57181
## accel_forearm_z      162.05369
## magnet_forearm_x     138.02601
## magnet_forearm_y     137.25045
## magnet_forearm_z     174.33619
```

Method 3: Generalized Boosted Regression Models

```
#Method 3: Generalized Boosted Regression Models
set.seed(1)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
Method3  <- train(classe ~ ., data=TrainDataCCTrain, method = "gbm", trControl = controlGBM, verbose = F
Method3
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10991, 10988, 10991, 10988, 10990
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7583909  0.6936536
##   1                  100      0.8299481  0.7846897
##   1                  150      0.8699855  0.8353990
##   2                   50      0.8838175  0.8528992
##   2                  100      0.9387780  0.9225333
##   2                  150      0.9605442  0.9500787
##   3                   50      0.9303346  0.9118176
##   3                  100      0.9673156  0.9586436
##   3                  150      0.9831124  0.9786375
##
## 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.
```

```
predictMethod3 <- predict(Method3, TrainDataCCTest)
cmGBM <- confusionMatrix(predictMethod3, TrainDataCCTest$classe)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1667   11    0    3    0
##          B    5 1118   12    7    4
##          C    0    9 1012   10    1
##          D    2    1    1  943    7
##          E    0    0    1    1 1070
##
## Overall Statistics
##
##                Accuracy : 0.9873
##                  95% CI : (0.9841, 0.99)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9839
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9816   0.9864   0.9782   0.9889
## Specificity            0.9967   0.9941   0.9959   0.9978   0.9996
## Pos Pred Value         0.9917   0.9756   0.9806   0.9885   0.9981
## Neg Pred Value         0.9983   0.9956   0.9971   0.9957   0.9975
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2833   0.1900   0.1720   0.1602   0.1818
## Detection Prevalence   0.2856   0.1947   0.1754   0.1621   0.1822
## Balanced Accuracy      0.9962   0.9878   0.9911   0.9880   0.9942
```

Final Answer, applying the best model

```r
#Final Answer, applying the best model
cmTree$overall[1]
```

```
##  Accuracy
## 0.7393373
```

```r
cmRF$overall[1]
```

```
## Accuracy
## 0.997791
```

```r
cmGBM$overall[1]
```

```
##  Accuracy
## 0.9872557
```

```r
Results <- predict(Method2,TestDataCC)
Results
```

```
##  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
```

The accuracy for decision tree classification is 0.7393373, for random forest is 0.997791, and for generlaized boosted regressio is 0.9872557 Since random forest provides the best accuracy, it is chosen for prediction.