Week 4 Assignment-Practical Machine Learning

Ambuj Tripathi

14 July 2021

** 1. Loading the Dataset**

The Dataset has been downloaded from the internet and has been loaded into two seperate dataframes, __“training__ and **“testing”**. The __“training__ data set has 19622 number of records and the __“testing__ data set has 20 records. The number of variables is 160.

```r
library(caret)
## Warning: package 'caret' was built under R version 3.4.4
## Loading required package: lattice
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 3.4.4
library(rpart)
library(rpart.plot)
## Warning: package 'rpart.plot' was built under R version 3.4.4
library(RColorBrewer)
library(RGtk2)
## Warning: package 'RGtk2' was built under R version 3.4.4
library(rattle)
## Warning: package 'rattle' was built under R version 3.4.4
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
library(randomForest)
## Warning: package 'randomForest' was built under R version 3.4.4
## 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
```

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.4.4
## 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 2.1.3
```

Here are the datasets, loaded directly from web and then downloaded.

```r
train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"



init_org_training_data <- read.csv(url(train_url))
init_org_testing_data <- read.csv(url(test_url))


dim(init_org_training_data)
```

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

```r
dim(init_org_testing_data)
```

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

** 2. Data Cleansing **

There are 3 parts in Data Cleansing.

A. **Removing Variables which are having nearly zero variance.**

```
non_zero_var <- nearZeroVar(init_org_training_data)



org_training_data <- init_org_training_data[,-non_zero_var]
org_testing_data <- init_org_testing_data[,-non_zero_var]

dim(org_training_data)
## [1] 19622   100
dim(org_testing_data)
## [1]  20 100
```

B. **Removing Variables which are having NA values. Our threshhold is 95%.**

```
na_val_col <- sapply(org_training_data, function(x) mean(is.na(x))) > 0.95

org_training_data <- org_training_data[,na_val_col == FALSE]
org_testing_data <- org_testing_data[,na_val_col == FALSE]

dim(org_training_data)
## [1] 19622    59
dim(org_testing_data)
## [1] 20 59
```

C. **Removing variables which are non-numeric and hence will not contribute into our model. The very first 7 variables are of that kind only. Hence those needs to be removed from the datasets.**

```
org_training_data <- org_training_data[,8:59]
org_testing_data <- org_testing_data[,8:59]

dim(org_training_data)
```

```
## [1] 19622    52
```

```
dim(org_testing_data)
```

```
## [1] 20 52
```

We can also check if the training data has the **“classe”** variable in it and the testing data has **“problem_id”** variable in it.

```
##colnames_train <- names(org_training_data)


##org_training_data <- init_org_training_data[, c(colnames_train, "problem_id")]


colnames(org_training_data)
```

```
##  [1] "pitch_belt"           "yaw_belt"             "total_accel_belt"
##  [4] "gyros_belt_x"         "gyros_belt_y"         "gyros_belt_z"
##  [7] "accel_belt_x"         "accel_belt_y"         "accel_belt_z"
## [10] "magnet_belt_x"        "magnet_belt_y"        "magnet_belt_z"
## [13] "roll_arm"             "pitch_arm"            "yaw_arm"
## [16] "total_accel_arm"      "gyros_arm_x"          "gyros_arm_y"
## [19] "gyros_arm_z"          "accel_arm_x"          "accel_arm_y"
## [22] "accel_arm_z"          "magnet_arm_x"         "magnet_arm_y"
## [25] "magnet_arm_z"         "roll_dumbbell"        "pitch_dumbbell"
## [28] "yaw_dumbbell"         "total_accel_dumbbell" "gyros_dumbbell_x"
## [31] "gyros_dumbbell_y"     "gyros_dumbbell_z"     "accel_dumbbell_x"
## [34] "accel_dumbbell_y"     "accel_dumbbell_z"     "magnet_dumbbell_x"
## [37] "magnet_dumbbell_y"    "magnet_dumbbell_z"    "roll_forearm"
## [40] "pitch_forearm"        "yaw_forearm"          "total_accel_forearm"
## [43] "gyros_forearm_x"      "gyros_forearm_y"      "gyros_forearm_z"
## [46] "accel_forearm_x"      "accel_forearm_y"      "accel_forearm_z"
## [49] "magnet_forearm_x"     "magnet_forearm_y"     "magnet_forearm_z"
## [52] "classe"
```

```
colnames(org_testing_data)
```

```
##  [1] "pitch_belt"           "yaw_belt"             "total_accel_belt"
##  [4] "gyros_belt_x"         "gyros_belt_y"         "gyros_belt_z"
##  [7] "accel_belt_x"         "accel_belt_y"         "accel_belt_z"
## [10] "magnet_belt_x"        "magnet_belt_y"        "magnet_belt_z"
## [13] "roll_arm"             "pitch_arm"            "yaw_arm"
## [16] "total_accel_arm"      "gyros_arm_x"          "gyros_arm_y"
## [19] "gyros_arm_z"          "accel_arm_x"          "accel_arm_y"
```

```
## [22] "accel_arm_z"          "magnet_arm_x"         "magnet_arm_y"
## [25] "magnet_arm_z"         "roll_dumbbell"        "pitch_dumbbell"
## [28] "yaw_dumbbell"         "total_accel_dumbbell" "gyros_dumbbell_x"
## [31] "gyros_dumbbell_y"     "gyros_dumbbell_z"     "accel_dumbbell_x"
## [34] "accel_dumbbell_y"     "accel_dumbbell_z"     "magnet_dumbbell_x"
## [37] "magnet_dumbbell_y"    "magnet_dumbbell_z"    "roll_forearm"
## [40] "pitch_forearm"        "yaw_forearm"          "total_accel_forearm"
## [43] "gyros_forearm_x"      "gyros_forearm_y"      "gyros_forearm_z"
## [46] "accel_forearm_x"      "accel_forearm_y"      "accel_forearm_z"
## [49] "magnet_forearm_x"     "magnet_forearm_y"     "magnet_forearm_z"
## [52] "problem_id"
```

** 3. Data Partitioning **

As per recommendation of the course __ *Practical Machine Learning*__ , we will be seggregating our **org_training_data** into 2 different parts, one is the training set (consisiting 60% of the total data) and test set (consisting 40% of the total data)

```
inTrain <- createDataPartition(org_training_data$classe, p=0.6, list=FALSE)
training <- org_training_data[inTrain,]
testing <- org_training_data[-inTrain,]


dim(training)
## [1] 11776    52
dim(testing)
## [1] 7846   52
```

** 4. Decision Tree Model **

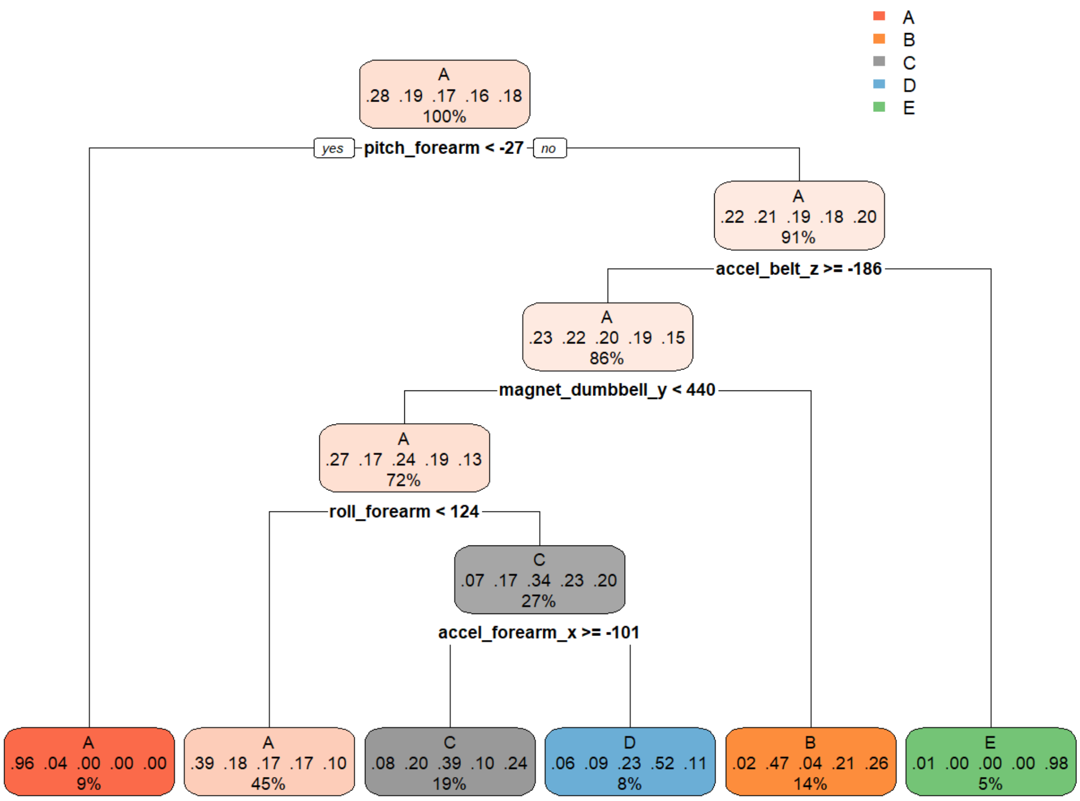
```
DT_modfit <- train(classe ~ ., data = training, method="rpart")
```

Prediction in terms of Decision Tree Model

```
DT_prediction <- predict(DT_modfit, testing)
confusionMatrix(DT_prediction, testing$classe)
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 2023  662  678  579  345
```

```
##          B   41  504   44  231  291
##          C  128  284  498  154  308
##          D   33   68  148  322   67
##          E    7    0    0    0  431
##
## Overall Statistics
##
##                Accuracy : 0.4815
##                  95% CI : (0.4704, 0.4926)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3206
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9064  0.33202  0.36404  0.25039  0.29889
## Specificity            0.5967  0.90408  0.86508  0.95183  0.99891
## Pos Pred Value         0.4719  0.45365  0.36297  0.50470  0.98402
## Neg Pred Value         0.9413  0.84944  0.86562  0.86626  0.86353
## Prevalence             0.2845  0.19347  0.17436  0.16391  0.18379
## Detection Rate         0.2578  0.06424  0.06347  0.04104  0.05493
## Detection Prevalence   0.5464  0.14160  0.17487  0.08132  0.05582
## Balanced Accuracy      0.7515  0.61805  0.61456  0.60111  0.64890
```

```
rpart.plot(DT_modfit$finalModel, roundint=FALSE)
```



We can see that the prediction accuracy is 50% which is not upto the desired level.

** 5. Random Forest Model **

```
RF_modfit <- train(classe ~ ., data = training, method = "rf", ntree = 100)
```

Prediction in terms of Random Forest Model

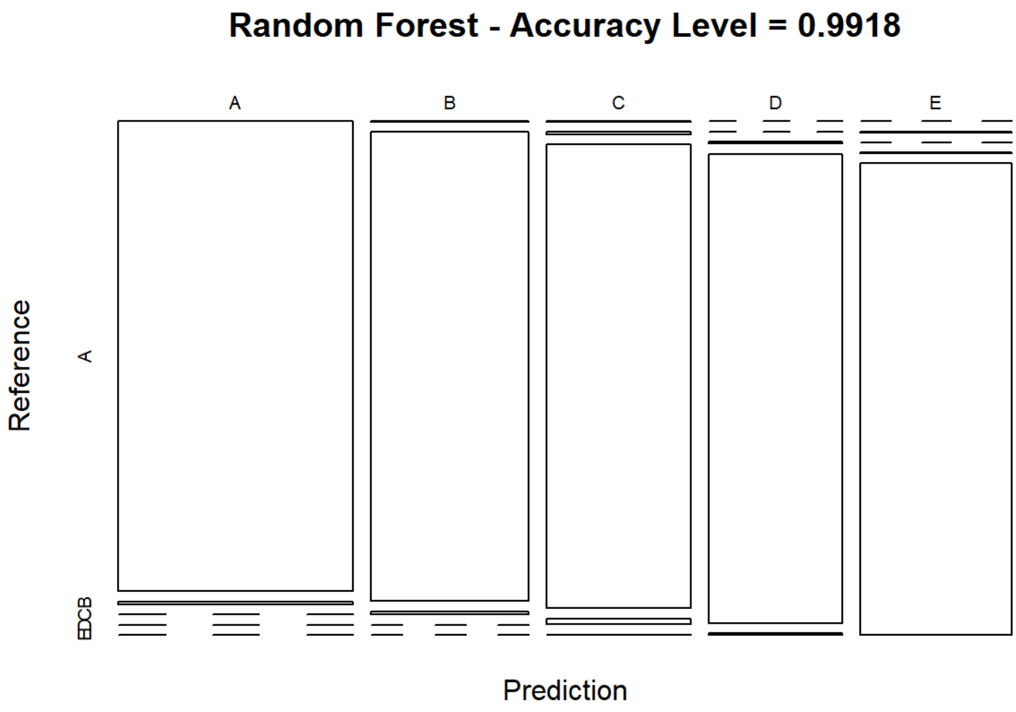
```
RF_prediction <- predict(RF_modfit, testing)
RF_pred_conf <- confusionMatrix(RF_prediction, testing$classe)
RF_pred_conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230   14    0    0    0
##          B    1 1495   10    0    0
##          C    1    7 1353   17    1
##          D    0    0    5 1268    5
##          E    0    2    0    1 1436
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9918
##                  95% CI : (0.9896, 0.9937)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9897
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9848   0.9890   0.9860   0.9958
## Specificity            0.9975   0.9983   0.9960   0.9985   0.9995
## Pos Pred Value         0.9938   0.9927   0.9811   0.9922   0.9979
## Neg Pred Value         0.9996   0.9964   0.9977   0.9973   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1905   0.1724   0.1616   0.1830
## Detection Prevalence   0.2860   0.1919   0.1758   0.1629   0.1834
## Balanced Accuracy      0.9983   0.9916   0.9925   0.9922   0.9977
```

Here is the plot

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



From the Confusion Matrix, we can clearly see that the prediction accuracy of Random Forest model is 99% which is satisfactory.

** 6. Gradient Boosting Model **

```
GBM_modfit <- train(classe ~ ., data = training, method = "gbm", verbose = FALSE)
GBM_modfit$finalModel
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 51 predictors of which 42 had non-zero influence.
```

Prediction in terms of GBM Model

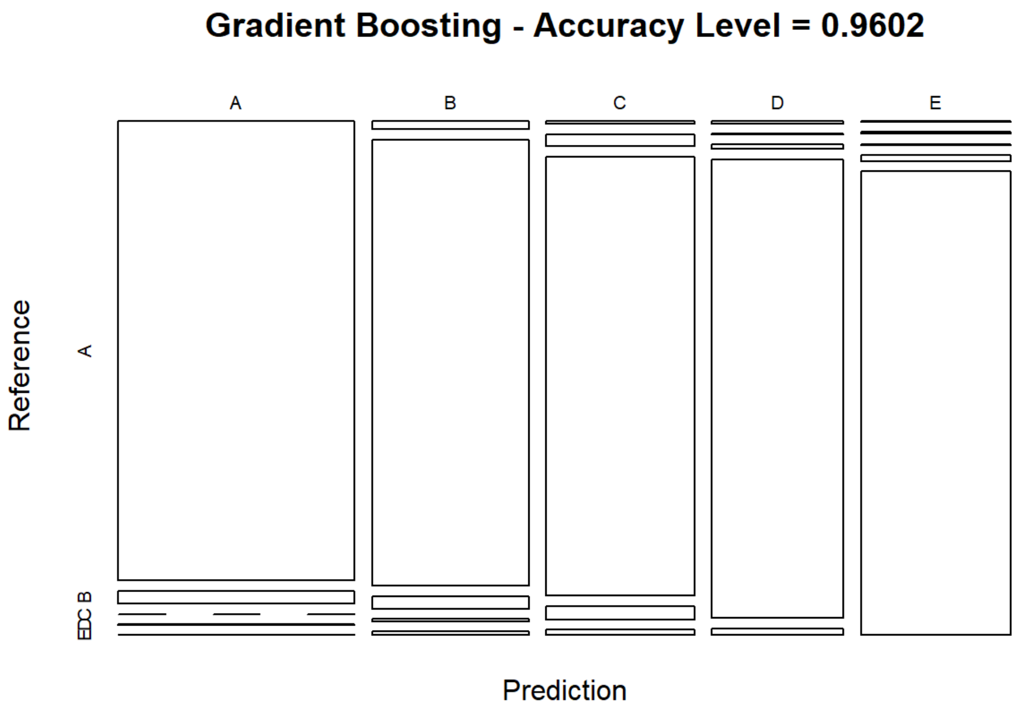
```
#GBM_prediction <- predict(GBM_modfit, testing, type = "class", n.trees = 5, type = link)
GBM_prediction <- predict(GBM_modfit, testing)


GBM_pred_conf <- confusionMatrix(GBM_prediction, testing$classe)
GBM_pred_conf
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2191   61    0    2    1
##          B   26 1414   40    7   11
##          C    8   36 1313   40   16
##          D    6    1   12 1220   18
##          E    1    6    3   17 1396
##
## Overall Statistics
##
##                Accuracy : 0.9602
##                  95% CI : (0.9557, 0.9645)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9497
##  Mcnemar's Test P-Value : 4.337e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9816   0.9315   0.9598   0.9487   0.9681
## Specificity            0.9886   0.9867   0.9846   0.9944   0.9958
## Pos Pred Value         0.9716   0.9439   0.9292   0.9706   0.9810
## Neg Pred Value         0.9927   0.9836   0.9915   0.9900   0.9928
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2793   0.1802   0.1673   0.1555   0.1779
## Detection Prevalence   0.2874   0.1909   0.1801   0.1602   0.1814
## Balanced Accuracy      0.9851   0.9591   0.9722   0.9715   0.9819
```

Here is the plot

```
plot(GBM_pred_conf$table, col = GBM_pred_conf$byClass,
     main = paste("Gradient Boosting - Accuracy Level =",
                  round(GBM_pred_conf$overall['Accuracy'], 4)))
```



From Gradient Boost Model, the prediction accuracy is 95% which is satisfactory.

** Now we need to see how each model has predicted the validation dataset across the classifications. ** We are not considering Decision Tree model as it didn't reach the satisfactory prediction accuracy level. SO only Random Forest and Gradient Boosting methods are being compared.

```
RF_pred_conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNu
ll
##      0.9918430      0.9896806      0.9895954      0.9937126      0.28447
62
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
GBM_pred_conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNu
ll
##   9.602345e-01   9.496836e-01   9.556727e-01   9.644503e-01   2.844762e-
01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   4.336741e-08
```

** Conclusion **

After checking the **Overall Statistics data**, the Random Forest model has definitely more accuracy than GBM. Hence we will be selecting **Random Forest model** for final prediction from **org_testing_data** .

** Final Prediction- Applying selected model on the Test Data **

```
Final_RF_prediction <- predict(RF_modfit, org_testing_data )
Final_RF_prediction
```

```
##  [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
```