Practical Machine Learning – Assignment

Gururaj

7/23/2020

Executive Summary

This assignment will analyse the data collected from wearable like Jawbone Up, Nike FuelBand, and Fitbit. Input data for this analysis comes from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. These participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. As part of this assignment, I will try to create prediction models using cross validation and calculating the sample error.

Load relevant libraries

Loading relevant R libraries. Assuming these libraries are already installed

```
library(caret);
```

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

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

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

```
## Loading required package: rpart
```

```
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:ggplot2':
##
##     margin
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
library(scales)
```

Data Source

Training Data is downloaded from : https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
Test Data is downloaded from : https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Data Extraction

Down load train & test data from the above sources.

```r
trainingURL = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingURL  = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainData   = read.csv(url(trainingURL))
testData    = read.csv(url(testingURL))
dim(trainData)
```

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

```r
dim(testData)
```

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

Data Pre-Processing

Remove variables with the following conditions:
1. Having greater than 95% of NA
2. Having Nearly Zero Variance
3. Having no relevance in the analysis

```r
#1

naCol       = sapply(trainData,function(x)mean(is.na(x)))>0.95
trainData = trainData[,naCol==FALSE]
testData  = testData[,naCol==FALSE]

#2
trainNZV   = nearZeroVar(trainData)
trainData  = trainData[,-trainNZV]
testNZV    = nearZeroVar(testData)
testData   = testData[,-testNZV]

#3
trainData = trainData[,-c(1:7)]
testData  = testData[,-c(1:7)]
dim(trainData)
```

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

```
dim(testData)
```

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

Data partition

```
inTrain     = createDataPartition(trainData$classe,p=3/4,list=FALSE)
trainingSet = trainData[inTrain,]
testingSet  = trainData[-inTrain,]
dim(trainingSet)
```

```
## [1] 14718    52
```

```
dim(testingSet)
```

```
## [1] 4904   52
```

Create prediction models

Using Decision Tree model and Random forest models

1. Random Forest Model

```
set.seed(100000)
rfModelFit         = train(classe ~.,data=trainingSet, method="rf",ntree=10)
rfPredictionFit    = predict(rfModelFit,trainingSet)
rfConfusionMatrix = confusionMatrix(rfPredictionFit,trainingSet$classe)
rfConfusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    1    0    0    0
##          B    0 2847    0    0    0
##          C    0    0 2567    2    0
##          D    0    0    0 2410    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 0.9998
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9997
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9996   1.0000   0.9992   1.0000
## Specificity            0.9999   1.0000   0.9998   1.0000   1.0000
## Pos Pred Value         0.9998   1.0000   0.9992   1.0000   1.0000
## Neg Pred Value         1.0000   0.9999   1.0000   0.9998   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1934   0.1744   0.1637   0.1839
## Detection Prevalence   0.2844   0.1934   0.1745   0.1637   0.1839
## Balanced Accuracy      1.0000   0.9998   0.9999   0.9996   1.0000
```

2. Decision Tree Model

```r
set.seed(100000)
dtModelFit        = train(classe ~.,data=trainingSet,method="rpart")
dtPredictionFit   = predict(dtModelFit,trainingSet)
dtConfusionMatrix = confusionMatrix(dtPredictionFit,trainingSet$classe)
dtConfusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3819 1185 1184 1078  648
##          B   67  966   78  423  545
##          C  209  298 1135  329  370
##          D   86  399  170  582  383
##          E    4    0    0    0  760
##
## Overall Statistics
##
##                Accuracy : 0.4934
##                  95% CI : (0.4853, 0.5015)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3373
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9125  0.33919  0.44215  0.24129  0.28086
## Specificity            0.6112  0.90623  0.90075  0.91565  0.99967
## Pos Pred Value         0.4826  0.46465  0.48484  0.35926  0.99476
```

```
## Neg Pred Value         0.9462  0.85110  0.88430  0.86028  0.86054
## Prevalence             0.2843  0.19350  0.17441  0.16388  0.18386
## Detection Rate         0.2595  0.06563  0.07712  0.03954  0.05164
## Detection Prevalence   0.5377  0.14126  0.15906  0.11007  0.05191
## Balanced Accuracy      0.7619  0.62271  0.67145  0.57847  0.64026
```

3. Linear Discriminant Analysis Model

```r
set.seed(100000)

ldaModelFit = train(classe ~ ., data=trainingSet, method = "lda")
ldaPredictionFit   = predict(ldaModelFit,trainingSet)
ldaConfusionMatrix = confusionMatrix(ldaPredictionFit,trainingSet$classe)
ldaConfusionMatrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##            A 3451  453  269  134  126
##            B   90 1818  253  135  469
##            C  314  324 1658  285  264
##            D  315  120  334 1714  286
##            E   15  133   53  144 1561
##
## Overall Statistics
##
##                Accuracy : 0.6932
##                  95% CI : (0.6856, 0.7006)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6114
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8246   0.6383   0.6459   0.7106   0.5769
## Specificity            0.9068   0.9202   0.9023   0.9143   0.9713
## Pos Pred Value         0.7785   0.6575   0.5828   0.6190   0.8190
## Neg Pred Value         0.9286   0.9138   0.9234   0.9416   0.9106
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2345   0.1235   0.1127   0.1165   0.1061
## Detection Prevalence   0.3012   0.1879   0.1933   0.1881   0.1295
## Balanced Accuracy      0.8657   0.7793   0.7741   0.8124   0.7741
```

Results from above analysis

```
## [1] "Rain Forest Model Accuracy: 99.98%"
```

```
## [1] "Decision Tree Model Accuracy: 49.34%"
```

```
## [1] "Linear Discriminant Analysis Model Accuracy: 69.32%"
```

Conclusion

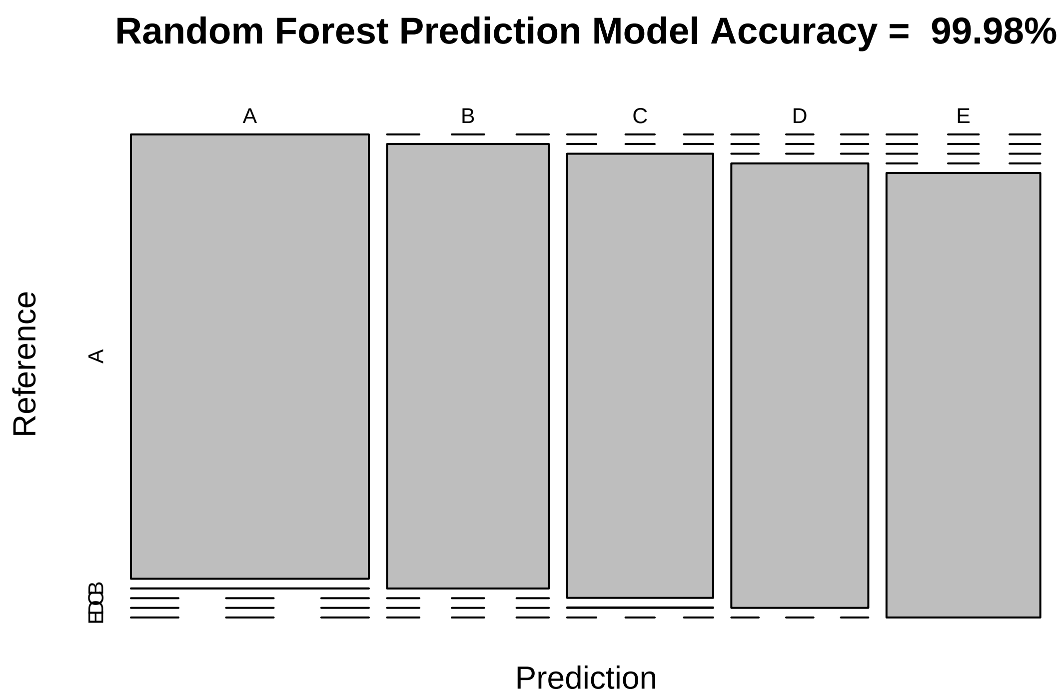
Based on analysis of above prediction models, Random Forest Model is best fitted in terms of highest accuracy and lowest sample error. Due to this reason, applying RF model to predict test/validation data

```
testPrediction = predict(rfModelFit,newdata = testData)
testPrediction
```

```
##  [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
```

Appendix

Plot-1 : Random Forest Prediction Model



Plot-2 : Decision Tree Prediction Model