Practical Machine Learning Assignment

Ashutosh

22/03/2020

#PRACTICAL MACHINE LEARNING ASSIGNMENT

##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. In this project, my goal will be 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

##Data The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

##“Loading the Data”

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.5.3
```

```
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.5.2
```

```
library(RGtk2)
```

```
## Warning: package 'RGtk2' was built under R version 3.5.3
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.5.3
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
## 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
```

```
raw_training_data<-read.csv("training.csv")
raw_testing_data<-read.csv("testing.csv")
dim(raw_training_data)
```

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

```
dim(raw_testing_data)
```

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

##Data cleansing and ordering

```
non_zero<-nearZeroVar(raw_training_data)
training_data<-raw_training_data[,-non_zero]
testing_data<-raw_testing_data[,-non_zero]
dim(training_data)
```

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

```
dim(testing_data)
```

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

```
na_value<-sapply(training_data, function(x) mean(is.na(x))) > 0.95
training_data <-training_data[,na_value == FALSE]
testing_data<-testing_data[,na_value == FALSE]
dim(training_data)
```

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

```
dim(testing_data)
```

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

###Removing the first 7 non numeric variables from the data

```
training_data<-training_data[,8:59]
testing_data<-testing_data[,8:59]
dim(training_data)
```

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

```
dim(testing_data)
```

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

```
colnames(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(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"
```

##Separating the data into Testing and Training sets I will be partitioning our 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<-caret::createDataPartition(training_data$classe, p=0.6,list=FALSE)
train<-training_data[inTrain,]
test<-training_data[-inTrain,]
dim(train)
```

```
## [1] 11776    52
```

```
dim(test)
```

```
## [1] 7846   52
```

##Decison Tree Model Prediction

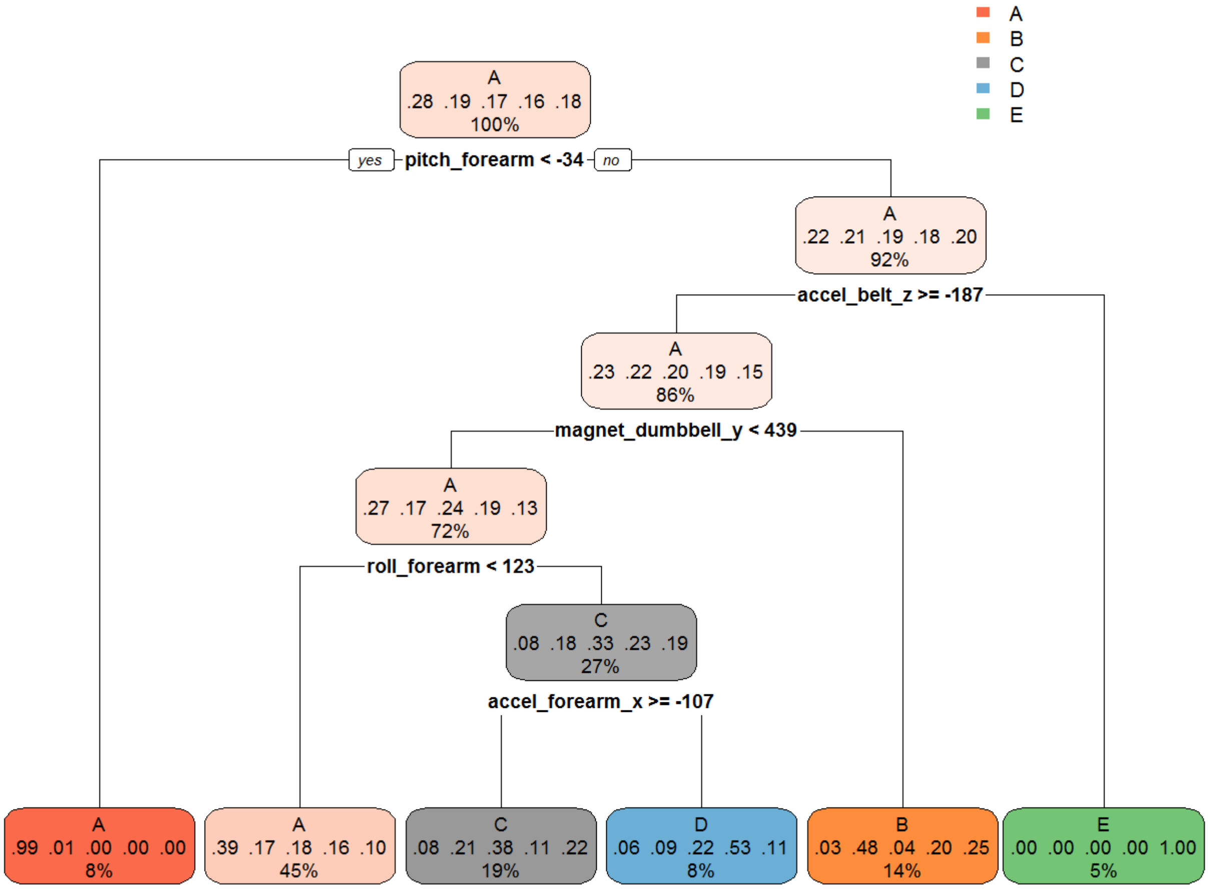
```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.5.3
```

```
DT_modfit<-train(classe ~ ., data = train, method="rpart")
DT_prediction <- predict(DT_modfit, test)
confusionMatrix(DT_prediction, test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2032  638  629  564  331
##          B   34  493   50  232  291
##          C  131  333  539  169  362
##          D   31   53  150  321   66
##          E    4    1    0    0  392
## 
## Overall Statistics
## 
##                Accuracy : 0.4814
##                  95% CI : (0.4703, 0.4925)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.3216
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9104  0.32477   0.3940  0.24961  0.27184
## Specificity            0.6149  0.90408   0.8464  0.95427  0.99922
## Pos Pred Value         0.4845  0.44818   0.3514  0.51691  0.98741
## Neg Pred Value         0.9452  0.84806   0.8687  0.86644  0.85904
## Prevalence             0.2845  0.19347   0.1744  0.16391  0.18379
## Detection Rate         0.2590  0.06283   0.0687  0.04091  0.04996
## Detection Prevalence   0.5345  0.14020   0.1955  0.07915  0.05060
## Balanced Accuracy      0.7626  0.61442   0.6202  0.60194  0.63553
```

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

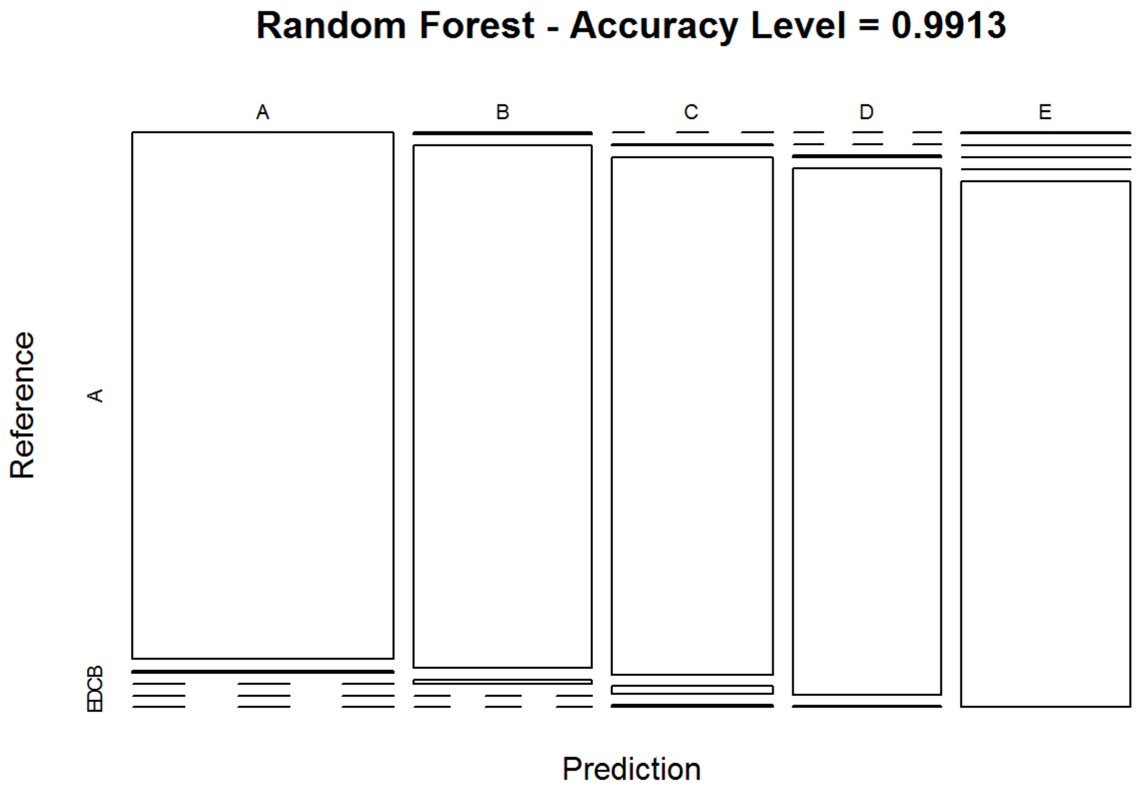


##Random Forest Model Prediction

```
RF_modfit<-train(classe ~ ., data = train, method = "rf", ntree = 100)
RF_prediction<-predict(RF_modfit, test)
RF_pred_conf<-confusionMatrix(RF_prediction, test$classe)
RF_pred_conf
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2226    9    0    0    0
##          B    4 1503   13    0    0
##          C    0    4 1351   20    5
##          D    0    0    3 1264    3
##          E    2    2    1    2 1434
## 
## Overall Statistics
##                                          
##                Accuracy : 0.9913         
##                  95% CI : (0.989, 0.9933)
##     No Information Rate : 0.2845         
##     P-Value [Acc > NIR] : < 2.2e-16      
##                                          
##                   Kappa : 0.989          
##                                          
##  Mcnemar's Test P-Value : NA             
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9973   0.9901   0.9876   0.9829   0.9945
## Specificity            0.9984   0.9973   0.9955   0.9991   0.9989
## Pos Pred Value         0.9960   0.9888   0.9790   0.9953   0.9951
## Neg Pred Value         0.9989   0.9976   0.9974   0.9967   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2837   0.1916   0.1722   0.1611   0.1828
## Detection Prevalence   0.2849   0.1937   0.1759   0.1619   0.1837
## Balanced Accuracy      0.9979   0.9937   0.9915   0.9910   0.9967
```

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



##Conclusion

The Random Forest model has significantly more accuracy than decision tree model hence we will be selecting Random Forest model for final prediction from testing_data .

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