Activity Recognition Using Predictive Analytics

Pooja Tandon

Introduction

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, our 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

Data Preprocessing

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Load Libraries required:

```r
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.6.2
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.2

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.6.2
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.6.2
```

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.2

## 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.
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.2

## 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
```

```r
library(RColorBrewer)
library(RGtk2)
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.6.2

## Loaded gbm 2.1.5
```

Loading Data

```r
train_url<- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url<- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training_data<- read.csv(url(train_url))
testing_data<- read.csv(url(test_url))
dim(training_data)
```

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

```r
dim(testing_data)
```

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

##Data Cleansing Removing Variables which are having Nearly Zero Variance.

```r
nzv <- nearZeroVar(training_data)

train_data <- training_data[,-nzv]
test_data <- testing_data[,-nzv]

dim(train_data)
```

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

```r
dim(test_data)
```

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

```r
na_val_col <- sapply(train_data, function(x) mean(is.na(x))) > 0.95
train_data <- train_data[,na_val_col == FALSE]
test_data <- test_data[,na_val_col == FALSE]

dim(train_data)
```

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

```r
dim(test_data)
```

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

```r
train_data<- train_data[, 8:59]
test_data<- test_data[, 8:59]
dim(train_data)
```

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

```r
dim(test_data)
```

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

Data Partioning

In this we will seggregate our train_data in two parts “training”(60% of data) and “testing”(40% of data)/ Validateion set.

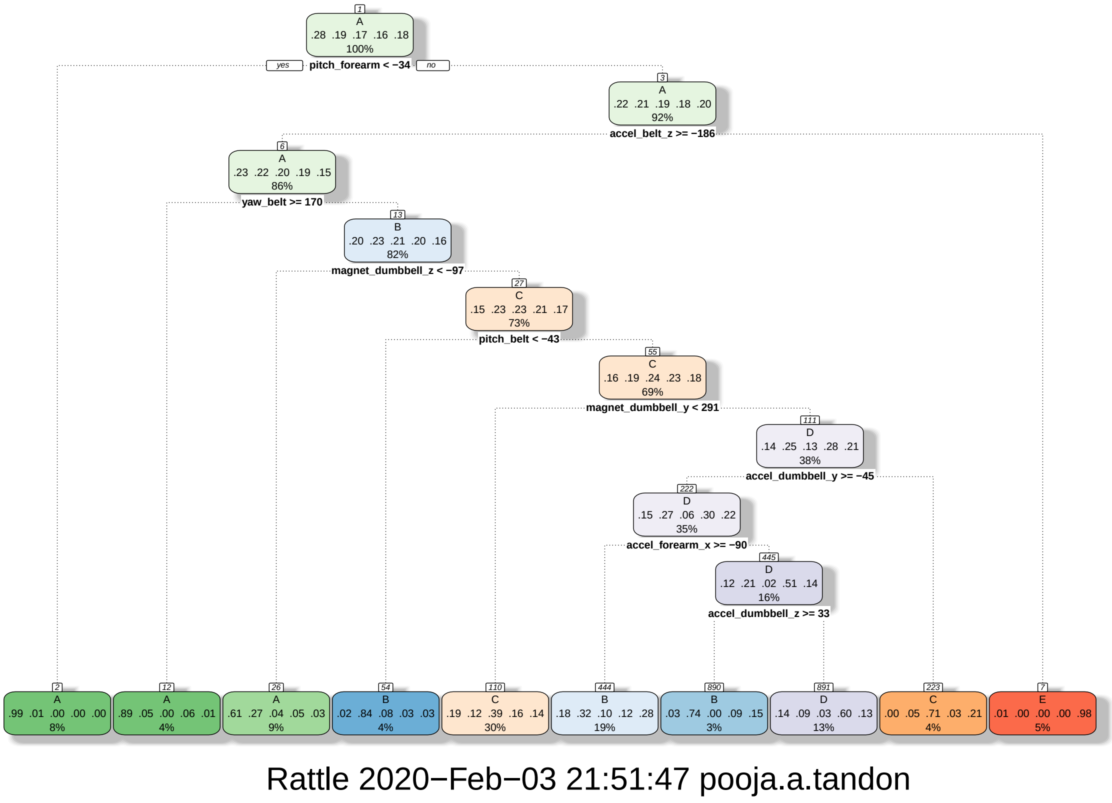
```r
inTrain<- createDataPartition(train_data$classe, p=0.6, list=FALSE)
inTrain<- createDataPartition(train_data$classe, p=0.6, list=FALSE)
training<- train_data[inTrain,]
testing<- train_data[-inTrain,]
dim(training)
```

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

Construct the Model using Cross Validation-

Decision Tree Model and Prediction

```r
library(rattle)
DT_model<- train(classe ~. , data=training, method= "rpart")
fancyRpartPlot(DT_model$finalModel)
```



```r
set.seed(21243)
DT_prediction<- predict(DT_model, testing)
confusionMatrix(DT_prediction, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1322  228   37   58   18
##          B  306  894  170  206  453
##          C  460  278 1118  386  406
##          D  137  118   43  636  161
##          E    7    0    0    0  404
##
## Overall Statistics
##
##                Accuracy : 0.5575
##                  95% CI : (0.5464, 0.5685)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4457
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5923   0.5889   0.8173  0.49456  0.28017
## Specificity            0.9393   0.8206   0.7638  0.93003  0.99891
## Pos Pred Value         0.7949   0.4406   0.4222  0.58082  0.98297
## Neg Pred Value         0.8528   0.8927   0.9519  0.90372  0.86039
## Prevalence             0.2845   0.1935   0.1744  0.16391  0.18379
## Detection Rate         0.1685   0.1139   0.1425  0.08106  0.05149
## Detection Prevalence   0.2120   0.2586   0.3375  0.13956  0.05238
## Balanced Accuracy      0.7658   0.7048   0.7905  0.71229  0.63954
```

From the Decision Tree Model we see the prediction accuracy is 57% which is not upto satisfactory level.

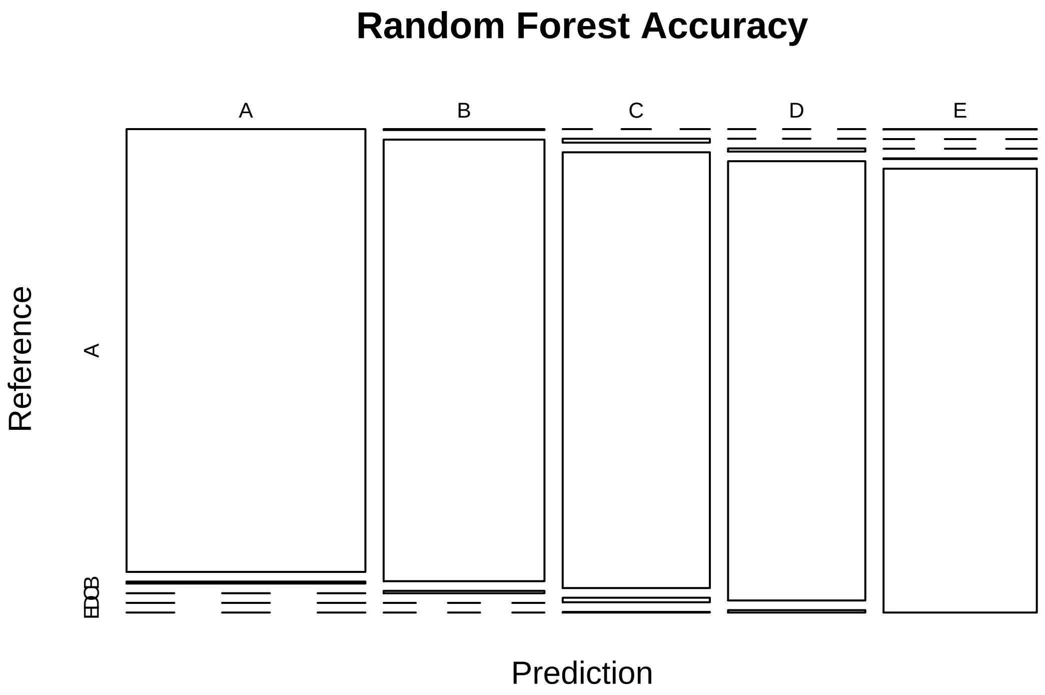
Random Forest Model and Prediction

```r
set.seed(26817)
###Fit the model
RF_model<- train(classe ~. , data=training, method= "rf", ntree=100)
###Prediction
RF_prediction<- predict(RF_model, testing)
RF_cm<-confusionMatrix(RF_prediction, testing$classe)
RF_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228   10    0    0    0
##          B    3 1496    8    0    0
##          C    0   12 1351   14    2
##          D    0    0    9 1270    7
##          E    1    0    0    2 1433
```

```
##
## 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            0.9982   0.9855   0.9876   0.9876   0.9938
## Specificity            0.9982   0.9983   0.9957   0.9976   0.9995
## Pos Pred Value         0.9955   0.9927   0.9797   0.9876   0.9979
## Neg Pred Value         0.9993   0.9965   0.9974   0.9976   0.9986
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1907   0.1722   0.1619   0.1826
## Detection Prevalence   0.2852   0.1921   0.1758   0.1639   0.1830
## Balanced Accuracy      0.9982   0.9919   0.9916   0.9926   0.9966
```

```
###plot
plot(RF_cm$table, col=RF_cm$byClass, main="Random Forest Accuracy")
```



From the Random Forest Model we see the prediction accuracy is 99% which is close to perfect accuracy level.

Gradient Boosting Model and Prediction

```
set.seed(25621)
gbm_model<- train(classe~., data=training, method="gbm", verbose= FALSE)
gbm_model$finalmodel
```

```
## NULL
```

```
###Prediction

gbm_prediction<- predict(gbm_model, testing)
gbm_cm<-confusionMatrix(gbm_prediction, testing$classe)
gbm_cm
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2197   49    0    1    5
##          B   27 1418   40   10   21
##          C    4   46 1307   32   14
##          D    3    0   18 1231   23
##          E    1    5    3   12 1379
## 
## Overall Statistics
## 
##                Accuracy : 0.96
##                  95% CI : (0.9554, 0.9642)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9494
## 
##  Mcnemar's Test P-Value : 4.442e-07
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9843   0.9341   0.9554   0.9572   0.9563
## Specificity            0.9902   0.9845   0.9852   0.9933   0.9967
## Pos Pred Value         0.9756   0.9354   0.9316   0.9655   0.9850
## Neg Pred Value         0.9937   0.9842   0.9905   0.9916   0.9902
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2800   0.1807   0.1666   0.1569   0.1758
## Detection Prevalence   0.2870   0.1932   0.1788   0.1625   0.1784
## Balanced Accuracy      0.9873   0.9593   0.9703   0.9753   0.9765
```

From the Gradient Boosting Model we see the prediction accuracy is 96% which is satisfied.

```
##we have taken Random Forest and Gradient Boosting Model because it reach to satisfied prediction leve
RF_cm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9913332      0.9890369      0.9890255      0.9932638      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
gbm_cm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.599796e-01   9.493642e-01   9.554046e-01   9.642087e-01   2.844762e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   4.441811e-07
```

Conclusion

we conclude that, Random Forest is more accurate than Gradient Boosting Model at upto 99% of accuracy level

Prediction - using Random Forest MOdel on testing data.

```
prediction_test<- predict(RF_model, test_data)
prediction_test
```

```
##  [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
```