Practical Machine Learning Week-4 Assignment

Asmaa abdlhady

july 29, 2019

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 Descriptions

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
library(knitr)
## Warning: package 'knitr' was built under R version 3.4.4
library(caret)
## Warning: package 'caret' was built under R version 3.4.4
## Loading required package: lattice
## Warning: package 'lattice' was built under R version 3.4.4
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 3.4.4
library(rpart)
## Warning: package 'rpart' was built under R version 3.4.4
```

```r
library(rpart.plot)
## Warning: package 'rpart.plot' 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 5.2.0 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 3.4.4
## 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
library(RColorBrewer)
## Warning: package 'RColorBrewer' was built under R version 3.4.4
library(RGtk2)
## Warning: package 'RGtk2' was built under R version 3.4.4
library(gbm)
## Warning: package 'gbm' was built under R version 3.4.4
## Loaded gbm 2.1.4
```

Loading Data

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

```r
training_data<- read.csv(url(train_url))
testing_data<- read.csv(url(test_url))
dim(training_data)
## [1] 19622   160
        dim(testing_data)
## [1]  20 160
```

Data Cleansing

```r
###Removing Variables which are having Nearly Zero Variance.
nzv <- nearZeroVar(training_data)

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

dim(train_data)
## [1] 19622   100
dim(test_data)
## [1]  20 100
###Removing NA Values of Variables.
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
dim(test_data)
## [1] 20 59
###Removing the first 7 Variables which are Non-Numeric.
train_data<- train_data[, 8:59]
test_data<- test_data[, 8:59]
dim(train_data)
## [1] 19622    52
dim(test_data)
```

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

Data Partioning

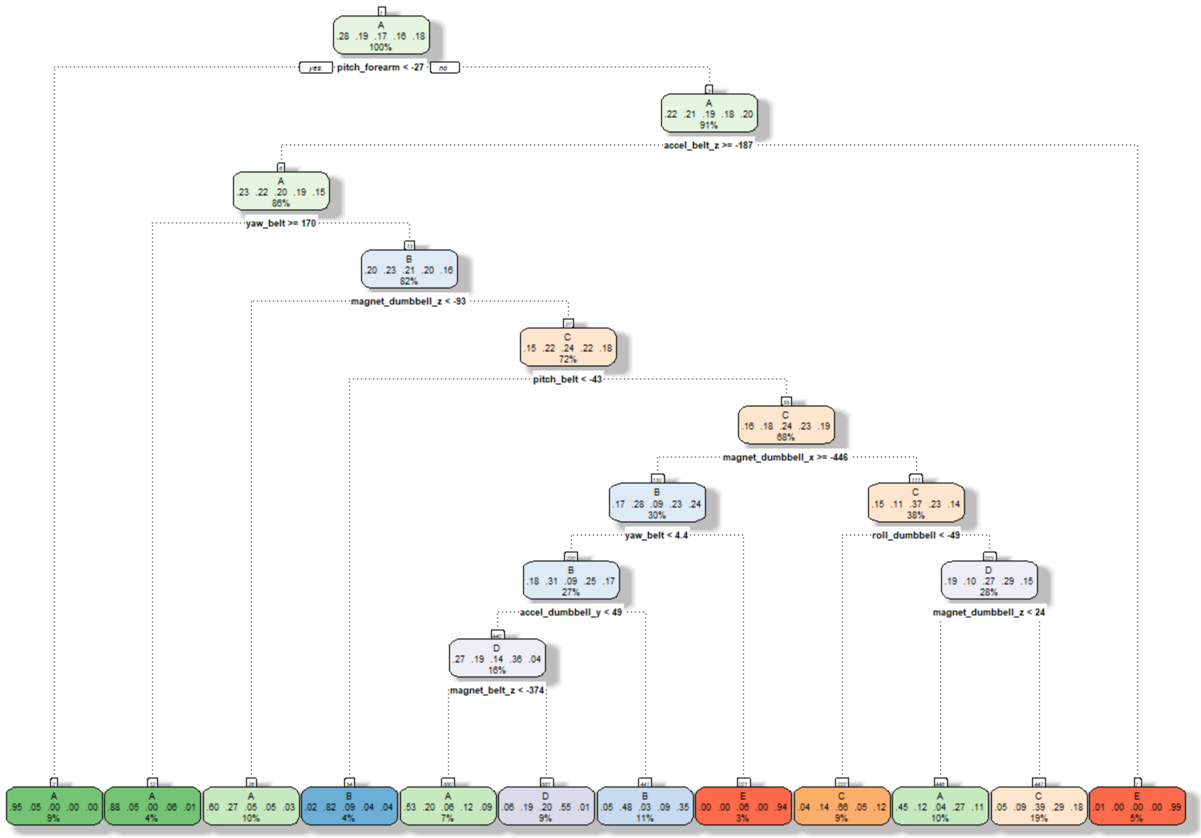
In this we will seggregate our **train_data** in two parts "**training**"(60% of data) and "**testing**"(40% of data)/ Validateion set.

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

Construct the Model using Cross Validation-

Decision Tree Model and Prediction

```
###Fit the model and plot
library(rattle)
DT_model<- train(classe ~. , data=training, method= "rpart")
fancyRpartPlot(DT_model$finalModel)
```



Rattle 2018-Dec-23 00:17:06 rekha

```
###Prediction
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 2019  463   95  353  165
##          B   53  700   54   93  283
##          C  118  217 1050  442  390
##          D   40  137  157  398    7
##          E    2    1   12    0  597
##
```

```
## Overall Statistics
##
##                Accuracy : 0.6072
##                  95% CI : (0.5963, 0.618)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4961
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9046  0.46113   0.7675  0.30949  0.41401
## Specificity            0.8083  0.92367   0.8199  0.94802  0.99766
## Pos Pred Value         0.6523  0.59172   0.4736  0.53857  0.97549
## Neg Pred Value         0.9552  0.87723   0.9435  0.87505  0.88319
## Prevalence             0.2845  0.19347   0.1744  0.16391  0.18379
## Detection Rate         0.2573  0.08922   0.1338  0.05073  0.07609
## Detection Prevalence   0.3945  0.15078   0.2826  0.09419  0.07800
## Balanced Accuracy      0.8565  0.69240   0.7937  0.62875  0.70583
```

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

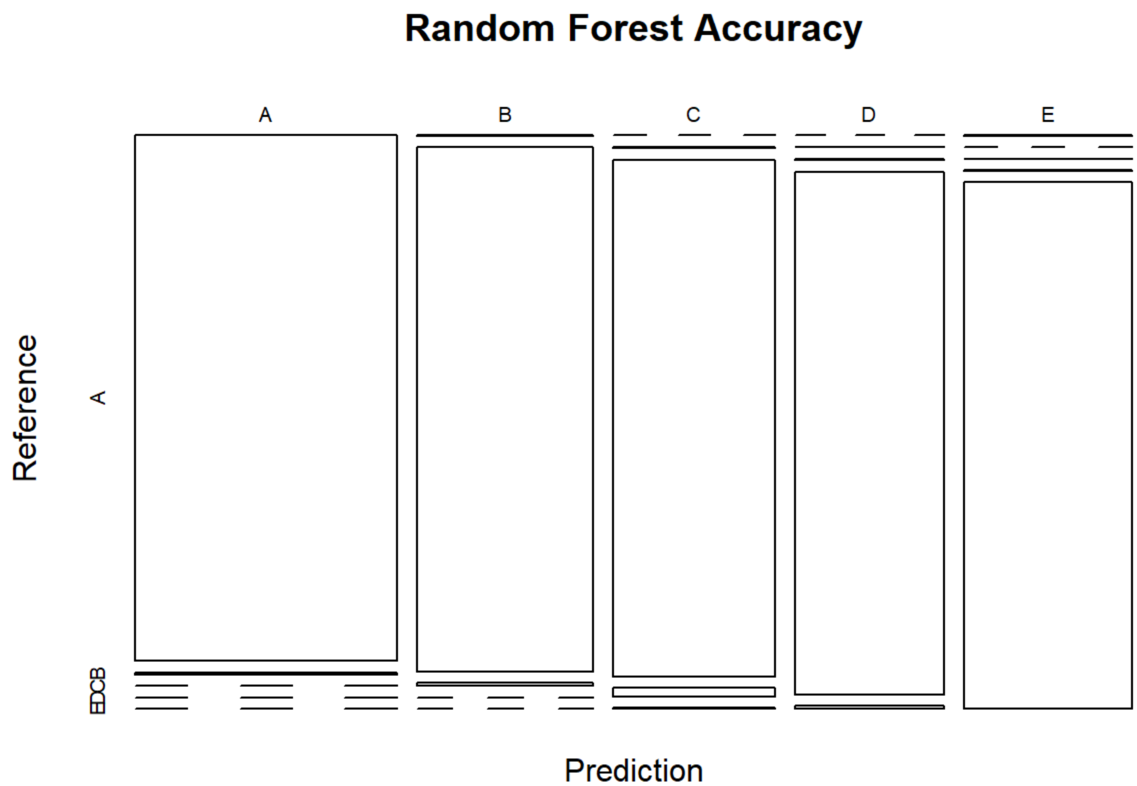
Random Forest Model and Prediction

```
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 2229    9    0    0    0
##          B    2 1504    8    0    0
##          C    0    4 1355   23    2
##          D    0    1    4 1262    7
##          E    1    0    1    1 1433
##
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9897, 0.9938)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9898
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9908   0.9905   0.9813   0.9938
## Specificity            0.9984   0.9984   0.9955   0.9982   0.9995
## Pos Pred Value         0.9960   0.9934   0.9790   0.9906   0.9979
## Neg Pred Value         0.9995   0.9978   0.9980   0.9963   0.9986
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1917   0.1727   0.1608   0.1826
## Detection Prevalence   0.2852   0.1930   0.1764   0.1624   0.1830
## Balanced Accuracy      0.9985   0.9946   0.9930   0.9898   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 2195   42    0    1    3
##          B   23 1425   38    5   13
##          C   10   45 1314   38   15
##          D    3    1   11 1237   27
##          E    1    5    5    5 1384
##
## Overall Statistics
##
##                Accuracy : 0.9629
##                  95% CI : (0.9585, 0.967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9531
##  Mcnemar's Test P-Value : 4.77e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9834   0.9387   0.9605   0.9619   0.9598
## Specificity            0.9918   0.9875   0.9833   0.9936   0.9975
## Pos Pred Value         0.9795   0.9475   0.9241   0.9672   0.9886
## Neg Pred Value         0.9934   0.9853   0.9916   0.9925   0.9910
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2798   0.1816   0.1675   0.1577   0.1764
## Detection Prevalence   0.2856   0.1917   0.1812   0.1630   0.1784
## Balanced Accuracy      0.9876   0.9631   0.9719   0.9777   0.9786
```

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 level. we are compairing the both model which is more ac
curate.

RF_cm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9919704      0.9898426      0.9897382      0.9938245      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
gbm_cm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.629110e-01   9.530844e-01   9.584902e-01   9.669836e-01   2.844762e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   4.770201e-09
```

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
```