Predictive Machine Learning

Socrates

Sunday, August 23, 2015

Executive Summary

This project is submitted as partial fulfilment of requirements of Practical Machine Learning course - part of Data Science curriculum. In this project, a training data set containing the key features of the personal activities of subjects measured using wearable devices such as *Jawbone Up, Nike FuelBand,* and *Fitbit* are provided to train and build the model. A test data set containing the key features of different subjects are provided to classify and determine the “Class” of each subjects, based on the model determined using training data set.

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 behaviour, 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, your 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.

Data Source

To eliminate he network dependency between Workstation where the analysis will be performed and the data source (https://d396qusza40orc.cloudfront.net/predmachlearn/), the training and test data sets are downloaded and made available locally.

Training data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) and test data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv) set files are downdloded and made available in (C:-Practical Machine Learning) folder.

Load Data

First set working directory to load data into the system and then load the training data. Since it is .CSV file the files were opened and inspected in Excel to feel the data. This was required to feel the data structure and the types of values they contain. This, in fact, provided many important insights about the data.

Set working Directory

```
setwd("C:/courses/8-Practical Machine Learning/Project")
```

Replace special non-usable values to NA

Since the data contains “#DIV/0!” as values in many of the fields, it is required to consider them as “NA” while loading.

```
training <- read.csv("pml-training.csv", na.strings=c("NA", "#DIV/0!") )
```

Remove columns containing NA

There are many columns that contain just “NA” without contributing much to the problem. So the columns that has only actual values are considered for this analysis.

```
training <- training[, colSums(is.na(training)) == 0]
```

Cleanup unusable columns

The data inspection in Excel also revealed that the first 7 columns are not directly participating in determining the “classe” of the model and they are more of informatory in nature. So it is decided to eliminate those columns before creating the model from dataset.

```
training <- training[, 8:length(colnames(training))]


colLen <- length(colnames(training))
colLen
```

```
## [1] 53
```

Load Data libraries

Load the libraries required for developing the model and performing analysis.

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(caret)
```

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

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

Split data into Training and Test sets

Since the test data set provided just contains 20 rows and they are mainly intended to answer the Project question, it is required to split the training data set into “training” and “test” data sets with 70-30 ratio.

```
inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
trainingData <- training[inTrain, ]
testingData <- training[-inTrain, ]


dim(trainingData)
## [1] 13737    53
dim(testingData)
## [1] 5885   53
```

Perform Predictions - Using Classification Tree

Now that cleanedup data is ready for analysis and for building models. Let us first use “Classification Tree” algorithm to train and build the model and test them.

Set the seed for reproducibility

```
set.seed(34344)
```

Train the data and print model results

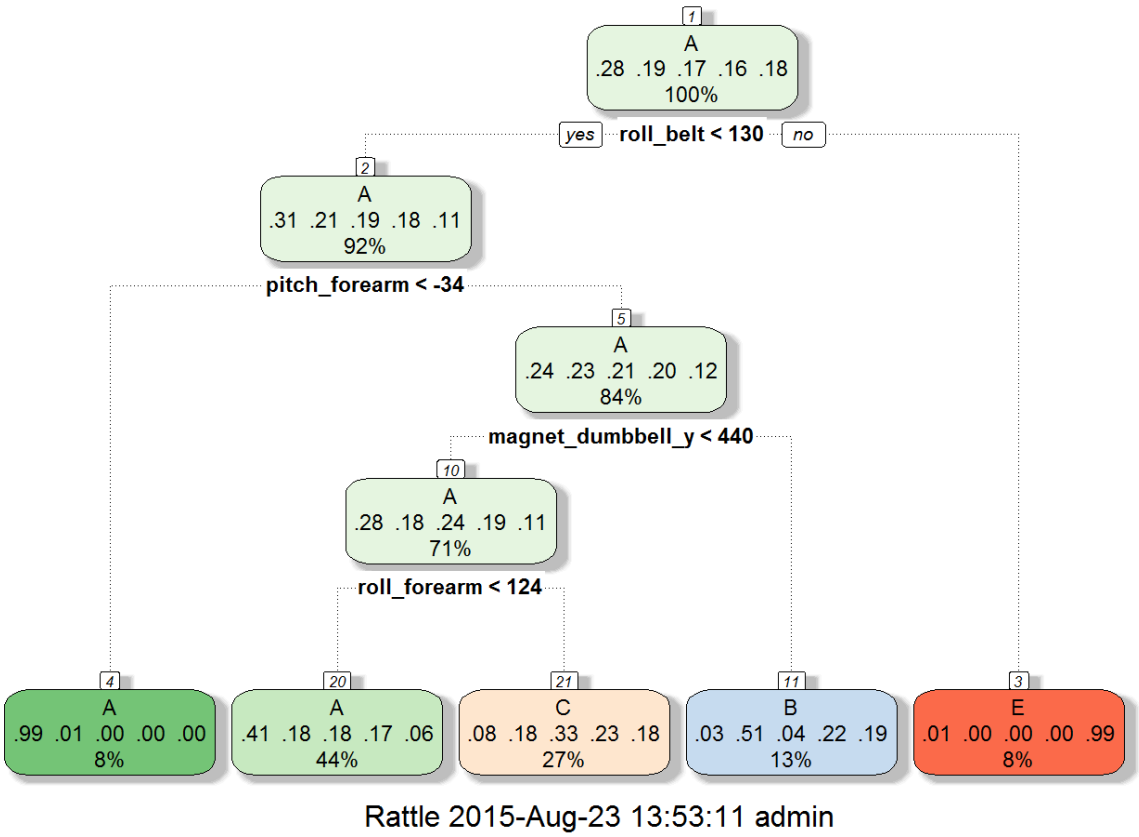
Using the training data set created above, train the data and develop Classification tree model. To create CT model, use `'method=rpart'`

```
modFitCT <- train(classe ~ ., data=training, method="rpart")
modFitCT
## CART
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
##
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0357  0.508     0.3606  0.0167       0.0278
##   0.0600  0.421     0.2181  0.0652       0.1087
##   0.1152  0.341     0.0883  0.0377       0.0563
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03567868.
```

Print Decision tree Model

```
fancyRpartPlot(modFitCT$finalModel)
```



Predict the data

Based on the model created above, use 'predict' function to predict the outcome of Testing Data to analyze the results and measure the accuracy of model.

```
predCT <- predict(modFitCT, newdata=testingData)
```

Create Confusion Matrix

Print the accuracy of model tested on testing data set

```
confusionMatrix(predCT, testingData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1518  481  460  436  147
##          B   26  375   37  169  150
##          C  125  283  529  359  284
##          D    0    0    0    0    0
##          E    5    0    0    0  501
##
## Overall Statistics
##
##                Accuracy : 0.4967
##                  95% CI : (0.4838, 0.5095)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3425
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9068  0.32924  0.51559   0.0000  0.46303
## Specificity            0.6381  0.91951  0.78370   1.0000  0.99896
## Pos Pred Value         0.4990  0.49538  0.33481      NaN  0.99012
## Neg Pred Value         0.9451  0.85101  0.88455   0.8362  0.89199
```

```
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2579  0.06372  0.08989   0.0000  0.08513
## Detection Prevalence   0.5169  0.12863  0.26848   0.0000  0.08598
## Balanced Accuracy      0.7725  0.62437  0.64965   0.5000  0.73100
```

Perform Predictions - Using Random Forest

The second part of this analysis is to build a model using “Random Forest” algorithm to train and build the model and test them.

Set the seed for reproducibility

```
set.seed(34344)
```

Train the data and print model results

Using the training data set created above, train the data and develop Random Forest model. To create RF model, use `'method=rf'`

```
modFitRF <- train(classe ~ ., data=training, method="rf")
## Loading required package: randomForest
## Warning: package 'randomForest' was built under R version 3.1.3
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
print(modFitRF, digits=3)
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
##
## Resampling results across tuning parameters:
##
```

```
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.993     0.991  0.00170      0.00215
##   27    0.993     0.991  0.00134      0.00169
##   52    0.985     0.981  0.00354      0.00446
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Predict the data

Based on the model created above, use 'predict' function to predict the outcome of Testing Data to analyze the results and measure the accuracy of model.

```
predRF <- predict(modFitRF, newdata=testingData)
```

Create Confusion Matrix

Print the accuracy of model tested on testing data set

```
confusionMatrix(predRF, testingData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    0 1026    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Out of sample error

The calculated accuracy from Random Forest algorithm: 1, that is 100%
The out-of-sample error (Error rate on new data test): 1 - 1 = 0.0, that is 0%

So the model derived from Random Forest is 100% accurate!!!

Conclusion

Comparing the results of both Classification Tree and Random Forest, it is more evident that Random Forest produces more accurate result than Classification Tree. So let us use the model built by Random forest to determine the outcome of test data provided.

```
# Load the testing data and apply the same clean-up process performed on trai
ning data set
testing <- read.csv("pml-testing.csv", na.strings=c("NA", "#DIV/0!") )
testing <- testing[, colSums(is.na(testing)) == 0]
testing <- testing[, 8:length(colnames(testing))]


predict(modFitRF, newdata=testing)
##  [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
```