Practical Machine Learning Assignment

Raymond Chua

26 April 2015

Executive Summary

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, 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

In this assignment, I used both Classification and Regression Tree(CART) and Random Forest to produce the best model. With an accuracy of 98.5% on test data, the Random Forest is eventually selected as the best model.

Loading up the required libraries

```r
library(caTools)
library(caret)
```

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

```r
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```r
library(corrplot)
library(e1071)
library(rpart)
library(rpart.plot)
library(randomForest)
```

Reading the data

```r
setwd("~/Documents/Coursera/Practical Machine Learning")
activityTrain = read.csv("pml-training.csv")
activityTest = read.csv("pml-testing.csv")
```

Cleaning the data

```
#remove columns with NA values
activityTrain = activityTrain[,colSums(is.na(activityTrain)) == 0]
activityTest = activityTest[,colSums(is.na(activityTest)) == 0]

#remove column X, columns with Time Stamps and columns windows
drops = c("X", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new_window", "num_win
activityTrain = activityTrain[,!(names(activityTrain) %in% drops)]
activityTest = activityTest[,!(names(activityTest) %in% drops)]

#convert all columns except classe to numeric
classe <- activityTrain$classe
activityTrain <- activityTrain[, sapply(activityTrain, is.numeric)]
activityTrain$classe <- classe
activityTest <- activityTest[, sapply(activityTest, is.numeric)]
```

Split the activityTrain data into training and testing data sets

```
#Split the data
set.seed(3000)
spl = sample.split(activityTrain$classe, SplitRatio = 0.7)
Train = subset(activityTrain, spl==TRUE)
Test = subset(activityTrain, spl==FALSE)
```

Build a Classification and Regression Model(CART)

A CART is a form of decision tree.

```
cartTree = rpart(classe ~ ., data = Train, method="class", minbucket=100)
```

Make prediction using the CART Model and check for accuracy using the Test data

```
# Make Predictions using the CART Model
PredictCART = predict(cartTree, newdata = Test, type = "class")

# Confusion Matrix using CART Model
confusionMatrixCart = confusionMatrix(Test$classe, PredictCART)
confusionMatrixCart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1462   49   42   72   49
##          B  154  720  121   89   55
##          C   19   83  827   69   29
##          D   62   80  156  611   56
##          E   16   97  128   50  791
##
## Overall Statistics
##
##                Accuracy : 0.7493
##                  95% CI : (0.738, 0.7603)
##     No Information Rate : 0.291
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6827
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8535   0.6997   0.6491   0.6857   0.8071
## Specificity            0.9492   0.9138   0.9566   0.9291   0.9407
## Pos Pred Value         0.8734   0.6321   0.8053   0.6332   0.7311
## Neg Pred Value         0.9404   0.9349   0.9080   0.9431   0.9607
## Prevalence             0.2910   0.1748   0.2164   0.1514   0.1665
## Detection Rate         0.2483   0.1223   0.1405   0.1038   0.1344
## Detection Prevalence   0.2844   0.1935   0.1745   0.1639   0.1838
## Balanced Accuracy      0.9013   0.8067   0.8029   0.8074   0.8739
```

```
#Accuracy using CART Model
AccuracyCART = confusionMatrixCart$overall[1]
AccuracyCART
```

```
##  Accuracy
## 0.7492781
```

```
#Out of Sample Error
errRate = 1 - unname(AccuracyCART)
errRate
```

```
## [1] 0.2507219
```

Next, we will try to build a random forest model to see if we can get a better accuracy on the test data.

Build a Random Forest Model

```
trainForest = randomForest(classe ~ . , data = Train, ntree=200, nodesize=25)
```

Make prediction using the Random Forest Model and check for accuracy using the Test data

```r
# Make Predictions using the Random Forest Model
predictForest = predict(trainForest, newdata = Test)

# Confusion Matrix using Random Forest Model
confusionMatrixRF = confusionMatrix(Test$classe, predictForest)
confusionMatrixRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    2    1    0    0
##          B   13 1121    5    0    0
##          C    0   14 1010    3    0
##          D    0    0   32  933    0
##          E    0    0    5    8 1069
##
## Overall Statistics
##
##                Accuracy : 0.9859
##                  95% CI : (0.9826, 0.9888)
##     No Information Rate : 0.2861
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9822
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9923   0.9859   0.9592   0.9883   1.0000
## Specificity            0.9993   0.9962   0.9965   0.9935   0.9973
## Pos Pred Value         0.9982   0.9842   0.9834   0.9668   0.9880
## Neg Pred Value         0.9969   0.9966   0.9912   0.9978   1.0000
## Prevalence             0.2861   0.1931   0.1789   0.1604   0.1816
## Detection Rate         0.2838   0.1904   0.1716   0.1585   0.1816
## Detection Prevalence   0.2844   0.1935   0.1745   0.1639   0.1838
## Balanced Accuracy      0.9958   0.9911   0.9778   0.9909   0.9987
```

```r
#Accuracy using Random Forest Model
AccuracyRF = confusionMatrixRF$overall[1]
AccuracyRF
```

```
##  Accuracy
## 0.9859011
```

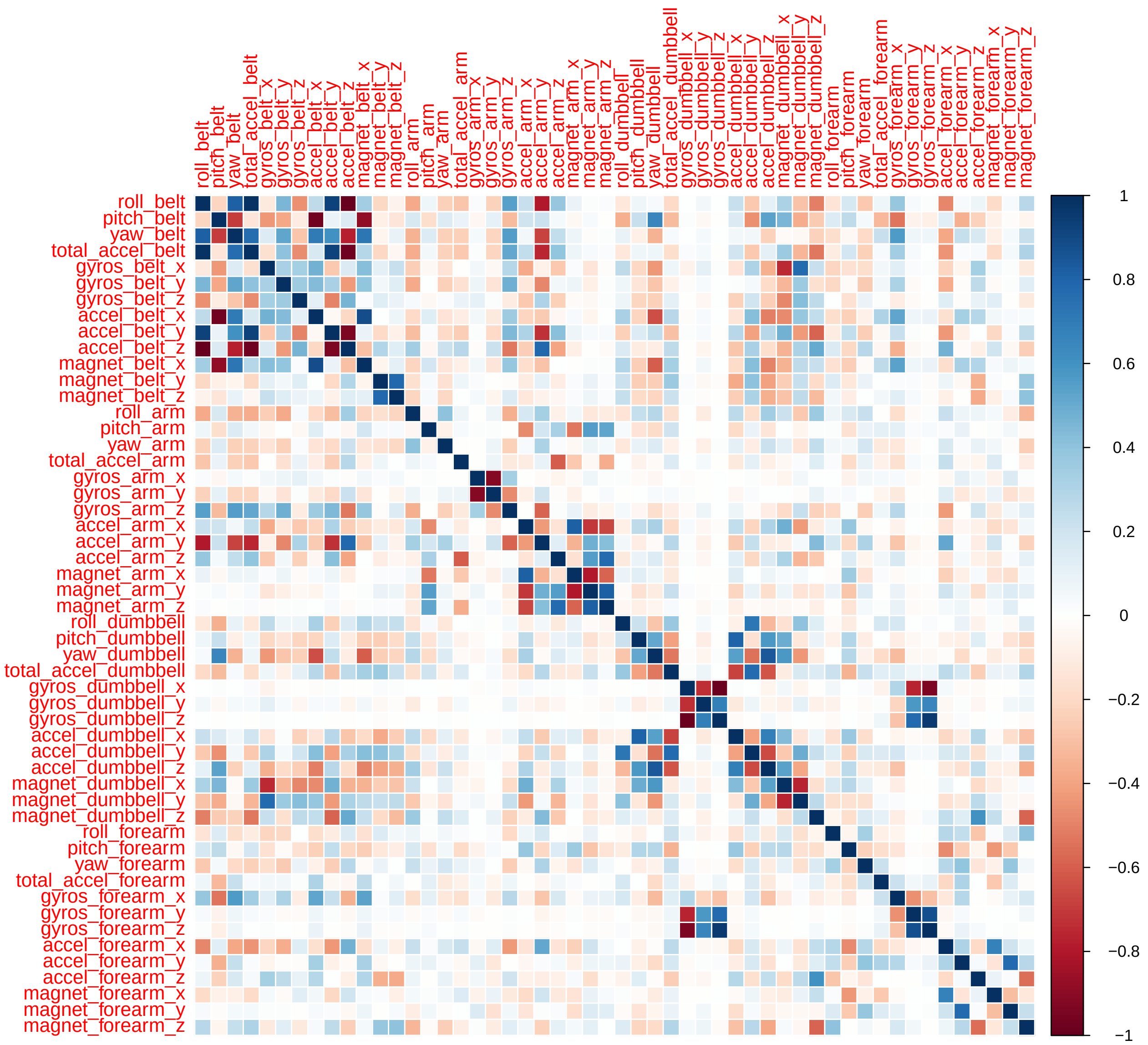
```r
#Out of Sample Error
errRateRF = 1 - unname(AccuracyRF)
errRateRF
```

```
## [1] 0.01409886
```

Appendix : Figures

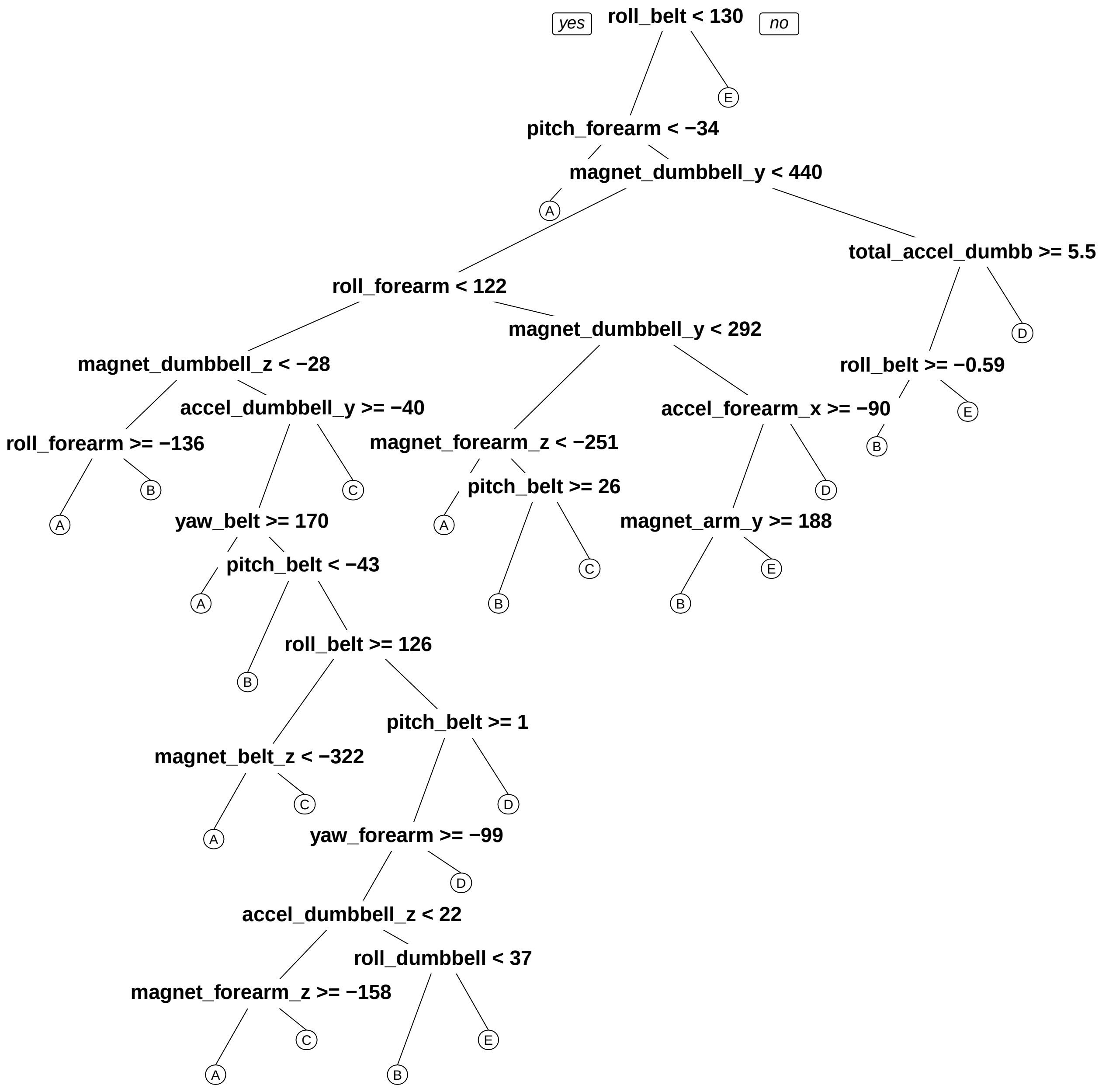
Correlation Matrix Visualization

```
corrPlot <- cor(Train[, -length(names(Train))])
corrplot(corrPlot, method="color")
```



Classification Tree Visualization

```
prp(cartTree, split.cex=1.5)
```


yes
roll_belt < 130
no
E
pitch_forearm < −34
magnet_dumbbell_y < 440
A
total_accel_dumbb >= 5.5
roll_forearm < 122
magnet_dumbbell_y < 292
D
roll_belt >= −0.59
magnet_dumbbell_z < −28
accel_dumbbell_y >= −40
accel_forearm_x >= −90
E
B
magnet_forearm_z < −251
roll_forearm >= −136
B
C
D
A
A
pitch_belt >= 26
yaw_belt >= 170
magnet_arm_y >= 188
pitch_belt < −43
C
E
A
B
B
B
roll_belt >= 126
pitch_belt >= 1
magnet_belt_z < −322
C
D
A
yaw_forearm >= −99
D
accel_dumbbell_z < 22
roll_dumbbell < 37
magnet_forearm_z >= −158
C
E
A
B