Practical Machine Learning - Weight Lifting Exercise Data

Hin Siong Chong

October 19, 2017

1. Executive Summary

In this course project, I will develop a machine-learning algorithm to analyze the Weight Lifting Exercise Dataset collected from accelerometers of 6 particupants to predict the manner in which they did the exercise. The dataset is available from http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

2. Data loading and processing

2.1 Loading packages and data

```r
## load packages
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.4.2

## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.2

## randomForest 4.6-12

## 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
## load data and treat empty values as NA
testing_raw <- read.csv("pml-testing.csv", na.strings=c("NA", ""), header=TRUE)
training_raw <- read.csv("pml-training.csv", na.strings=c("NA", ""), header=TRUE)
```

```
## explore data
dim(training_raw)
```

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

```
sum(complete.cases(training_raw))
```

```
## [1] 406
```

```
dim(testing_raw)
```

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

```
sum(complete.cases(testing_raw))
```

```
## [1] 0
```

```
## check identical column names of both training and testing files
colnames_training <- colnames(training_raw)
colnames_testing <- colnames(testing_raw)
all.equal(colnames_training[1:length(colnames_training)-1], colnames_testing[1:length(colnames_testing)-
```

```
## [1] TRUE
```

2.2 Data cleansing

Datasets are filtered and subsetted for data analysis

```
## remove columns with NA values
training_raw <- training_raw[, colSums(is.na(training_raw)) == 0]
testing_raw <- testing_raw[, colSums(is.na(testing_raw)) == 0]

## remove columns not needed for measurements
classe <- training_raw$classe
# for training set
training_remove <- grepl("^X|timestamp|window", names(training_raw))
training_raw <- training_raw[, !training_remove]
training_clean <- training_raw[, sapply(training_raw, is.numeric)]
training_clean$classe <- classe
# for testing set
testing_remove <- grepl("^X|timestamp|window", names(testing_raw))
testing_raw <- testing_raw[, !testing_remove]
testing_clean <- testing_raw[, sapply(testing_raw, is.numeric)]
```

2.3 Splitting the training set

Cleaned training dataset is split into a training subset (comprising 60% of original entries) and a testing subset (40% of original entries).

```
set.seed(1122) ## for reproducibility
in_train <- createDataPartition(training_clean$classe, p=0.60, list=FALSE)
my_train <- training_clean[in_train, ]
my_test <- training_clean[-in_train, ]
dim(my_train); dim(my_test)
```

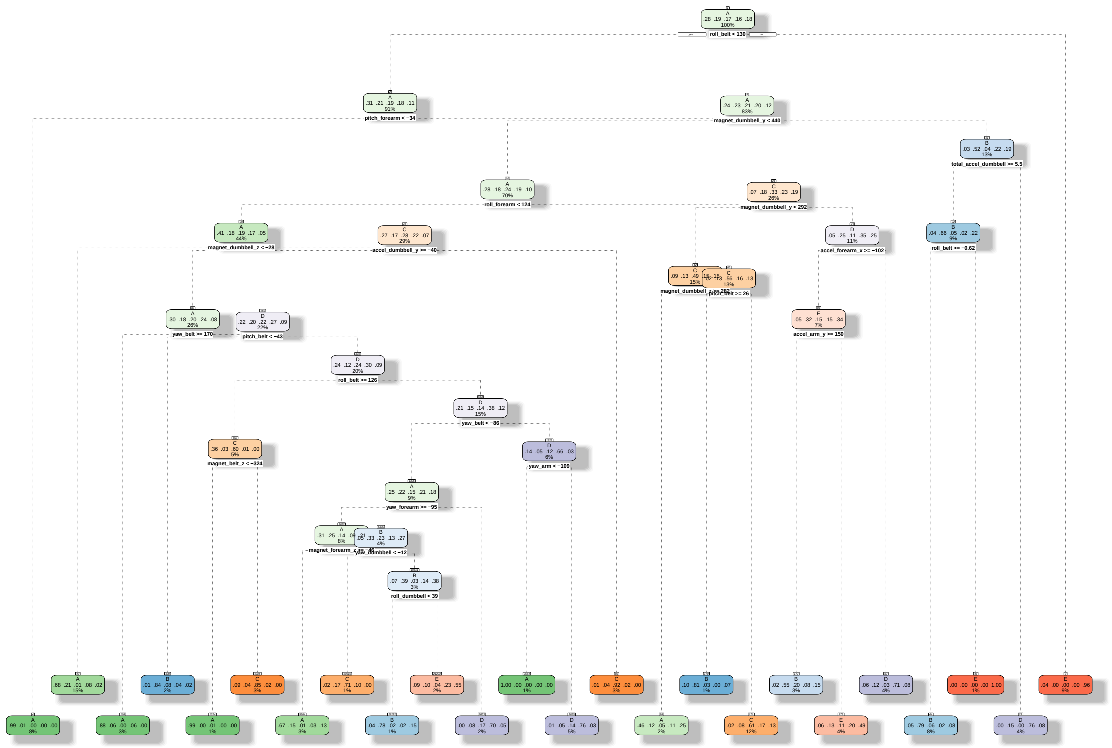
```
## [1] 11776    53
## [1] 7846   53
```

3. Model selection

3.1 rpart Model

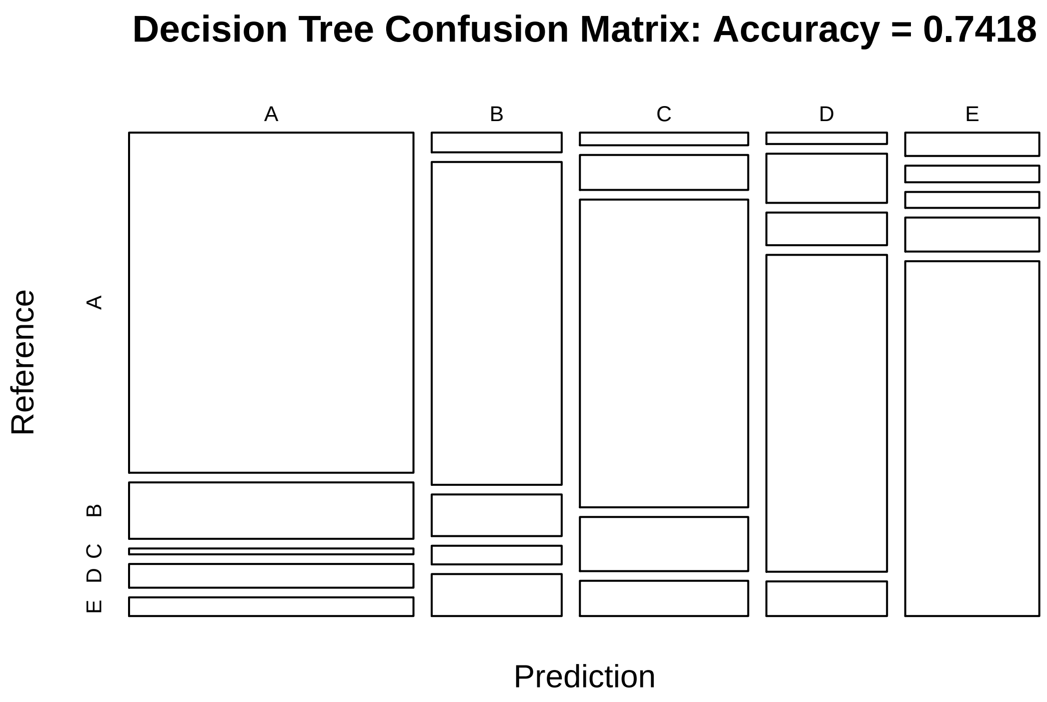
```r
set.seed(1122)
modFit_Trees <- rpart(classe~., method="class", data=my_train)
fancyRpartPlot(modFit_Trees)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2017−Oct−19 17:00:58 HS

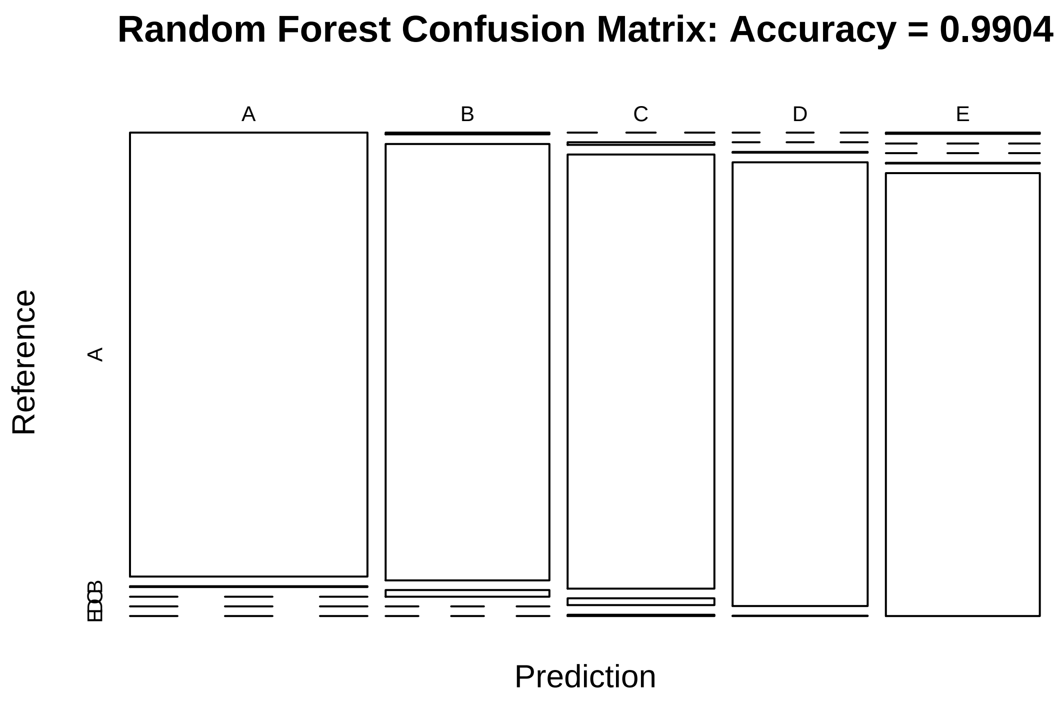
```r
predict_Trees <- predict(modFit_Trees, my_test, type = "class")
CM_Trees <- confusionMatrix(predict_Trees, my_test$classe)
plot(CM_Trees$table, col = CM_Trees$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy =",
```



In testing this model on the testing subset, it is revealed to have an accurary of 74%. This is the most accurate for Class A and the least accurate for Class B. Out of sample error is 100-74% = 26%.

3.2 Random Forest Model

```
set.seed(1122)
modFit_RF <- randomForest(classe ~ ., data=my_train)
predict_RF <- predict(modFit_RF, my_test, type = "class")
CM_RF <- confusionMatrix(predict_RF, my_test$classe)
plot(CM_RF$table, col = CM_Trees$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", ro
```



In testing this model on the testing subset, it is revealed to have a high accurary of 99%. This is the least accurate for Class C with a 98.8% accuracy. Out of sample error is 100-99% = 1%.

3.3 Generalized Boosted Regression

```
set.seed(1122)
modFit_GBR <- train(classe ~ ., data=my_train, method = "gbm", trControl = trainControl(method = "repea
```

```
## Warning: package 'gbm' was built under R version 3.4.2
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
predict_GBR <- predict(modFit_GBR, newdata=my_test)
accuracy_GBR <- confusionMatrix(predict_GBR, my_test$classe)
accuracy_GBR
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2190   50    0    0    3
##          B   31 1414   55    3   19
##          C    5   51 1295   61    4
##          D    6    2   14 1213   18
##          E    0    1    4    9 1398
##
## Overall Statistics
##
##                Accuracy : 0.9572
##                  95% CI : (0.9525, 0.9615)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9458
##  Mcnemar's Test P-Value : 1.367e-10
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9812   0.9315   0.9466   0.9432   0.9695
## Specificity            0.9906   0.9829   0.9813   0.9939   0.9978
## Pos Pred Value         0.9764   0.9290   0.9145   0.9681   0.9901
## Neg Pred Value         0.9925   0.9836   0.9886   0.9889   0.9932
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2791   0.1802   0.1651   0.1546   0.1782
## Detection Prevalence   0.2859   0.1940   0.1805   0.1597   0.1800
## Balanced Accuracy      0.9859   0.9572   0.9640   0.9686   0.9837
```

In testing this model on the testing subset, it is revealed to have a accurary of 95.7%. This is the least accurate for Class B. Out of sample error is 100-95.7% = 4.3%.

4. Predicting results on the test data

Based on the model selection, Random Forest model provides the highest accuracy over that of Decision Trees or Generalized Boosted Regression. Thus, Random Forest model is used as a final model to test the test data.

```r
final_result <- predict(modFit_RF, testing_clean, type = "class")
final_result
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

5. Conclusion

Random Forest is a better model for prediction of exercise quality in this project over other models such as rpart and generalized boosted regression. It produces an accuracy of 99% with an out of sample error of 1%.