Practical Machine Learning: Prediction Project

Author: Surya Gurung
19th November, 2016

Executive Summary

In this project, I analysed the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The five different ways are classified as exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). These classifications are recored under variable ‘classe’. The goal of this project is to build a model to predict the manner (one of the 5 ways) in which they did the exercise using the training data and use the prediction model to predict 20 different test cases in test data. The training data for this project are downloaded from this site (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) and test data are downloaded from here. (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

First, I cleaned both data sets by removing variables with more than 70% ‘NA’ values and near zero variance. I also removed first two columns since they are irrelevent variables. The train data set is divided into two subsets. The bigger subset is the training set which is used to build the prediction models. The smaller subset is used as a validation set to find the accuracy of the prediction models before it is used for the test data set. I built a decision tree model using rpart function with training data subset and used the model to predict on the test subset. Similary, I built a random forest prediction model with 3-fold cross-validation to select optimal parameters for the model. The accuracy of decision tree model is 0.8646 where as the random forest prediction model has accuracy of 0.9997. So, I chose the random forest model to predict on the test data set.

Loading and Preprocessing the data

Loading necessary libraries and the project data:

```
options(warn = -1)
library(ggplot2)
library(caret)
```

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

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

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

```
## Loading required package: rpart
```

```
# sets working director as '~/rprojects'. If it doesn't exit, it creates one.
if (file.exists('~/rprojects')){
    setwd('~/rprojects')
}else {
    dir.create('~/rprojects')
    setwd('~/rprojects')
}

trainingFile = "pmlTraining.csv"
testFile = "pmlTesting.csv"

# if the data is not downloaded yet, downloads the csv file from given url.
if (!file.exists(trainingFile)){
    fileURL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv'
    download.file(fileURL, destfile = trainingFile, method='curl')
}
if (!file.exists(testFile)){
    fileURL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testin
g.csv'
    download.file(fileURL, destfile = testFile, method='curl')
}

trainData <- read.csv('pmlTraining.csv', na.strings = 'NA')
testData <- read.csv('pmlTesting.csv', na.strings = 'NA')
```

Preprocessing the training and testing data:

```
names(trainData) # results not shown here
head(trainData)  # results not shown here
```

By inspecting the variables as above, first two variables (‘X’ and ‘user_name’) seems irrelavent. So, lets get ride of these two columns from both data sets.

```
trainData <- trainData[ , -(1:2)]
testData <- testData[ , -(1:2)]
```

Some of the varialbles have lots of ‘NA’ value. So, lets get rid of the variables with more than 70% of its values as ‘NA’. Also using nearZeroVar function, lets get ride of columns with near zero variance.

```
cleanNACols <- sapply(trainData, function(naCol) mean(is.na(naCol)))
trainData <- trainData[ ,(cleanNACols > 0.70) == FALSE] # removes cols with mor
e than 70% NAs value
testData <- testData[ ,(cleanNACols > 0.70) == FALSE] # removes cols with more
than 70% NAs value

preObj <- nearZeroVar(trainData, saveMetrics = TRUE)
trainData <- trainData[ ,preObj$nzv == FALSE]
testData <- testData[,preObj$nzv == FALSE]
```

Building the Models

Now, lets build two prediction models:

- Predicting with Decision Tree
- Predicting with Random Forest

First, lets partition the training data set, **trainData**, into a training set, **trainingSet**, with 70% and a validation set, **cvTestingSet**, with 30% of data so that I can cross-validate the model built on the trainingSet before I can use the model to predict on test data, **testData**.

```
set.seed(331)

inTrain <- createDataPartition(y = trainData$classe, p = 0.7, list = FALSE)
trainingSet <- trainData[inTrain, ]
cvTestingSet <- trainData[-inTrain, ]
```

Predicting with Decision Tree:

```
dtModelFit<-rpart(classe ~ ., data = trainingSet, method = 'class')
pred <- predict(dtModelFit, cvTestingSet, type = 'class')
confusionMatrix(pred,cvTestingSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1616   39    6    2    0
##          B   35  942   62   48    0
##          C   23  151  941  150   45
##          D    0    7    8  606   54
##          E    0    0    9  158  983
##
## Overall Statistics
##
##                Accuracy : 0.8646
##                  95% CI : (0.8556, 0.8732)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8286
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9654   0.8270   0.9172   0.6286   0.9085
## Specificity            0.9888   0.9694   0.9241   0.9860   0.9652
## Pos Pred Value         0.9717   0.8666   0.7183   0.8978   0.8548
## Neg Pred Value         0.9863   0.9589   0.9814   0.9313   0.9791
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2746   0.1601   0.1599   0.1030   0.1670
## Detection Prevalence   0.2826   0.1847   0.2226   0.1147   0.1954
## Balanced Accuracy      0.9771   0.8982   0.9206   0.8073   0.9369
```

Predicting with Random Forest:

```
tControl <- trainControl(method = 'cv', number = 3)
rfModelFit <- train(classe ~ ., data = trainingSet, method = 'rf', trControl =
tControl)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rfModelFit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 38
##
##         OOB estimate of  error rate: 0.07%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3906    0    0    0    0 0.0000000000
## B    1 2657    0    0    0 0.0003762227
## C    0    2 2392    2    0 0.0016694491
## D    0    0    1 2250    1 0.0008880995
## E    0    0    0    3 2522 0.0011881188
```

```
pred1 <- predict(rfModelFit, cvTestingSet)
confusionMatrix(pred1,cvTestingSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    2    0    0
##          C    0    0 1024    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9988, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9996
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9981   1.0000   1.0000
## Specificity            1.0000   0.9996   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   0.9982   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9996   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1740   0.1638   0.1839
## Detection Prevalence   0.2845   0.1939   0.1740   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9998   0.9990   1.0000   1.0000
```

From above confusion matrices, we can see the accuracy of decision tree model is 0.8646 where as the random forest prediction model has accuracy of 0.9997. The plotings of decision tree (Figure 1) and random forest error (Figure 2) are listed in Index at the end of this report. So, I am using the random forest model to predict on the test data set.

Predicting on Test Data

Since the random forest prediction model gave way better accuracy, I am using it to predict on the test data set, **testData**. But before doing this, I need to train the model on full training data set, **trainData**, for better accuracy in predicting with **testData**. So, the random forest model is trained again using the full and cleaned train data set, **trainData**.

```
modelFit <- train(classe ~ ., data = trainData, method = 'rf', trControl = tCon
trol)
modelFit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 38
##
##         OOB estimate of  error rate: 0.05%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5580    0    0    0    0 0.0000000000
## B    2 3795    0    0    0 0.0005267316
## C    0    4 3418    0    0 0.0011689071
## D    0    0    1 3214    1 0.0006218905
## E    0    0    0    1 3606 0.0002772387
```

```
pred2 <- predict(modelFit, testData)
#confusionMatrix(pred2,testData$classe)
```

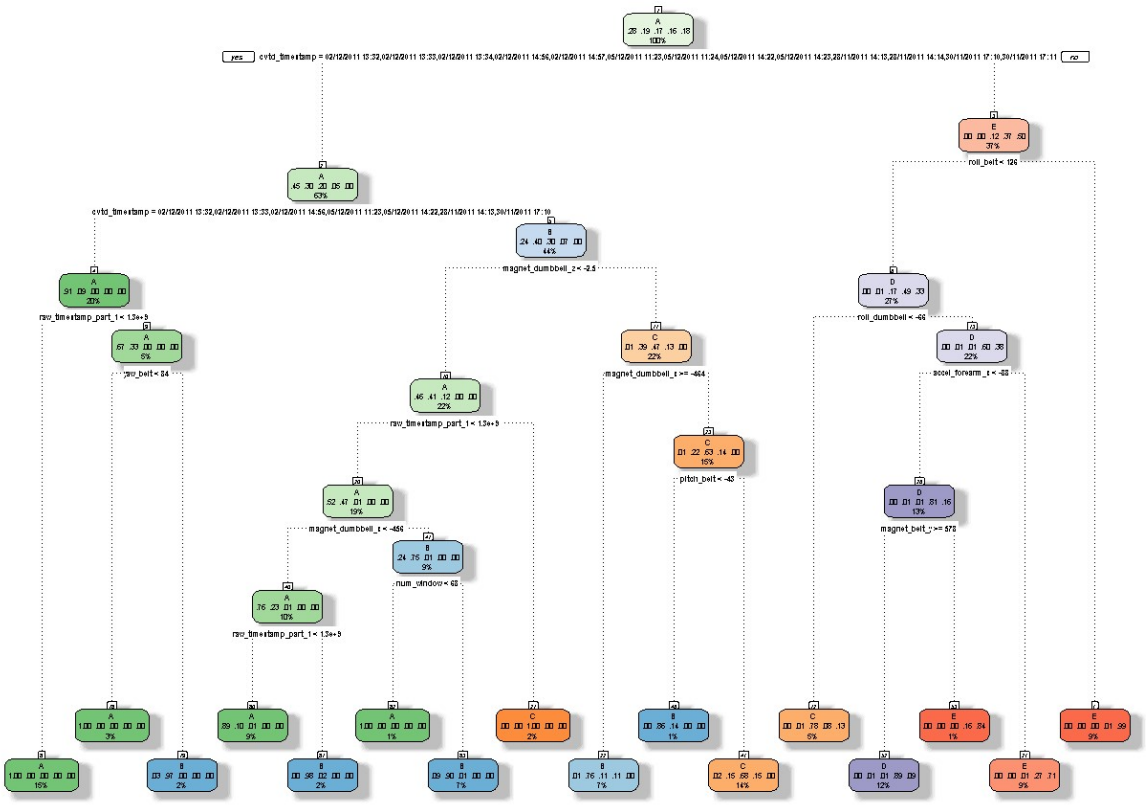
Function to generate a text file of the prediction:

```
generatePredictionFile <- function(prediction){
    for (i in 1:length(prediction)){
        fName <- paste0('problemId_', i, '.txt')
        write.table(prediction[i], file = fName, quote = FALSE, row.names = FAL
SE, col.names = FALSE)
    }
}
generatePredictionFile(pred2)
```

Index

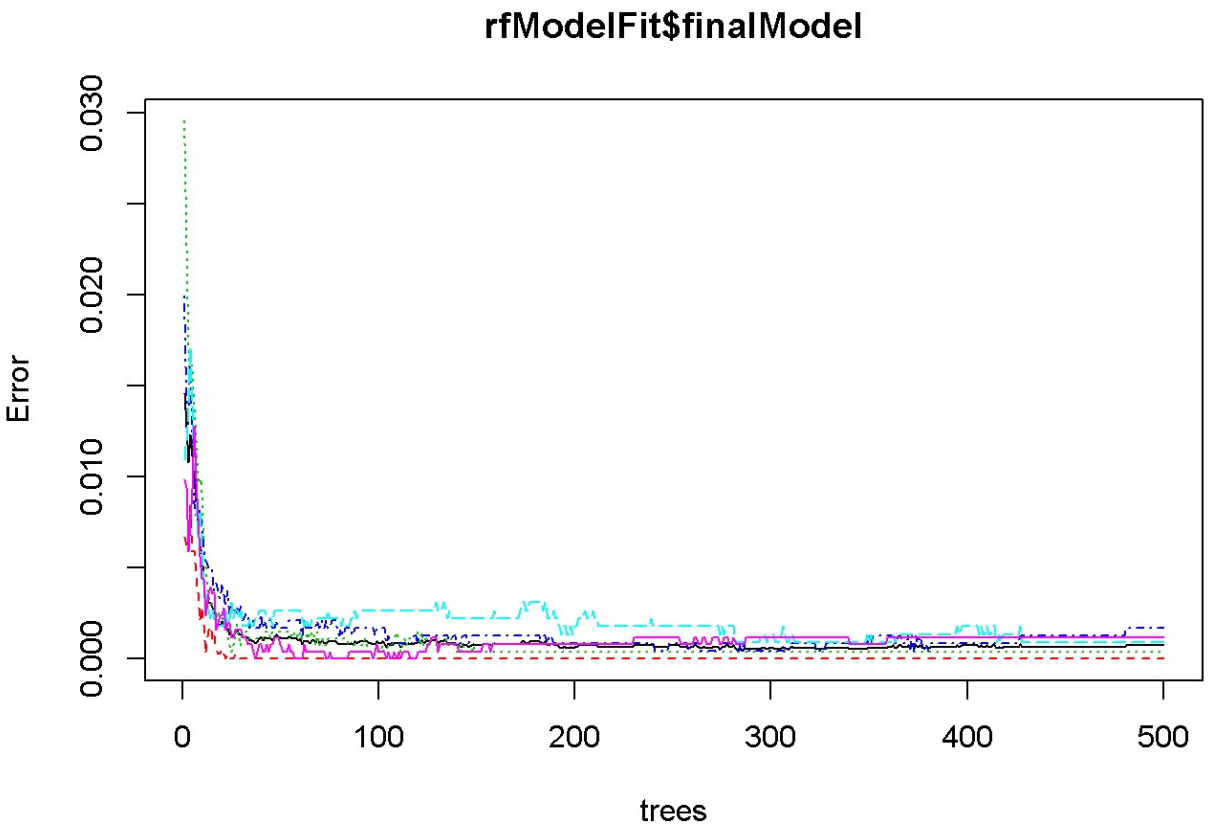
(Figure 1)

```
fancyRpartPlot(dtModelFit)
```



(Figure 2)

```
plot(rfModelFit$finalModel)
```


rfModelFit$finalModel
Error
0.000
0.010
0.020
0.030
0
100
200
300
400
500
trees