Weight Lifting Exercise Execution Performance

Mohit Arora

Summary

In this project, our goal is 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. Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: 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).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Data Processing

Loading data and keeping only those variables which actually make a difference in the prediction. Such as removing the variables which mostly have NA's in it and also those variables which do not have much variance and also which do not have effect on the prediction.

```
#Loading required packages
library(caret)
```

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

```
library(randomForest)
```

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

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.2.2
```

```
## Loading required package: RGtk2
```

```
## Warning: package 'RGtk2' was built under R version 3.2.2
```

```
## 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.
```

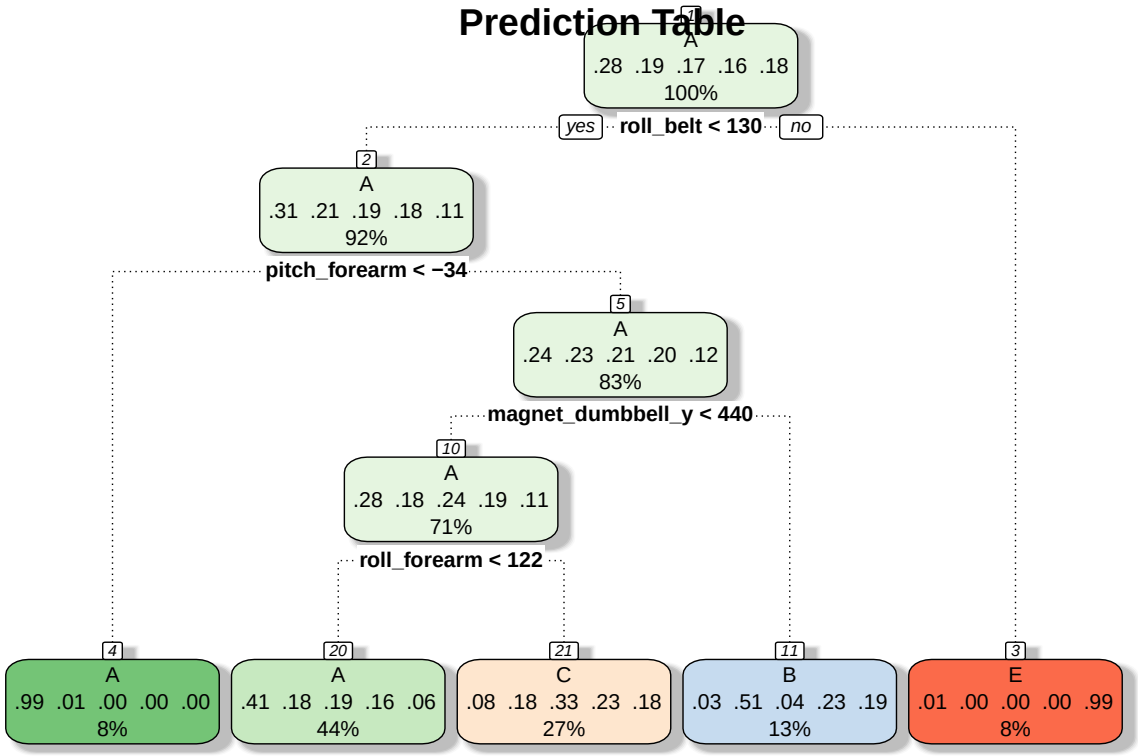
```r
#Reading the data
trainingData <- read.csv("pml-training.csv", na.strings = "NA")
testingData <- read.csv("pml-testing.csv", na.strings = "NA")

#Removing the columns which has more than 95% as NA's in it.
trainingData <- trainingData[, colSums(is.na(trainingData)) <=   0.95 * nrow(trainingData)]

#Removing near zoer variance variables from the data.
nsv <- nearZeroVar(trainingData)
trainingData <- trainingData[, -nsv]

#Removing variables which donot have effect on prediction such as time related variables.
trainingData <- trainingData[, -c(1:6)]

#Including only those variables in testing data which are present in training data.
testingData <- testingData[, which(names(testingData) %in% colnames(trainingData))]

#Creating a partition of data to get validation data set as well with 80% in training and 20% in valida
part <- createDataPartition(trainingData$classe, p = 0.8, list = FALSE)
trainingSubset <- trainingData[part, ]
validationSubset <- trainingData[-part, ]
```

First Prediction Model: Using Decision Tree

```r
#Using Rpart model
rpartModel <- train(classe ~ ., data = trainingSubset, method = "rpart")
```

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

```r
#Using rattle library to create prediction tree
fancyRpartPlot(rpartModel$finalModel, main = "Prediction Table")
```




```
rpartPrediction <- predict(rpartModel, validationSubset)
confusionMatrix(rpartPrediction, validationSubset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 999 301 304 300  99
##          B  24 267  26  98 100
##          C  93 191 354 245 209
##          D   0   0   0   0   0
##          E   0   0   0   0 313
##
## Overall Statistics
##
##                Accuracy : 0.4927
##                  95% CI : (0.477, 0.5085)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3379
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8952  0.35178  0.51754   0.0000  0.43412
## Specificity            0.6423  0.92162  0.77215   1.0000  1.00000
## Pos Pred Value         0.4988  0.51845  0.32418      NaN  1.00000
## Neg Pred Value         0.9391  0.85563  0.88343   0.8361  0.88698
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2547  0.06806  0.09024   0.0000  0.07979
## Detection Prevalence   0.5106  0.13128  0.27836   0.0000  0.07979
## Balanced Accuracy      0.7687  0.63670  0.64485   0.5000  0.71706
```

Second Prediction Model : Using Random Forest Model

```
#Using Rf model
rfModel <- randomForest(classe ~ ., data = trainingSubset, method = "class")

#Prediction on Validation Set
rfValidationPrediction <- predict(rfModel, validationSubset, type = "class")
confusionMatrix(rfValidationPrediction, validationSubset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    4    0    0    0
##          B    1  751    0    0    0
##          C    0    4  683    8    1
##          D    0    0    1  633    2
##          E    0    0    0    2  718
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9912, 0.9963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9926
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9895   0.9985   0.9844   0.9958
## Specificity            0.9986   0.9997   0.9960   0.9991   0.9994
## Pos Pred Value         0.9964   0.9987   0.9813   0.9953   0.9972
## Neg Pred Value         0.9996   0.9975   0.9997   0.9970   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1914   0.1741   0.1614   0.1830
## Detection Prevalence   0.2852   0.1917   0.1774   0.1621   0.1835
## Balanced Accuracy      0.9988   0.9946   0.9973   0.9918   0.9976
```

As we can clearly see that Random forest model has much better accuracy as compared to Rpart model

which use decision tree for prediction. RF model gave 99% accuracy for validation set as compared to only 50% by Rpart model.

```
#Prediction on Testing data set
rfTestPrediction <- predict(rfModel, testingData, type = "class")
rfTestPrediction
```

```
##  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
```

Submission

```
#Writing files for submission
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(rfTestPrediction)
```