WEIGHT LIFTING EXERCISE DATA ANALYSIS

Korhan Tezel

Overview

Using the data from devices such as Jawbone Up, Nike FuelBand, and Fitbit, which collect data of human body movement, we will try to identify how well the participants perform a barbell lift activity. Participants were asked to perform the same set of exercises correctly and incorrectly with accelerometers placed on the belt, forearm, and dumbbell.

Summary

The data is separated into training and test data sets by 70/30 ratio respectively. After separation, 3 models are used to predict the "classe" variable.

1) Random Forests: ran with 10-fold cross-validation

   This prediction model turned out pretty accurate when it was ran on the test data set.

2) Decision Tree:

   This prediction model is the least accurate one.

3) Gradient Boost with 10-fold cross-validation

   This prediction model is also pretty accurate when used on the test data set.

Getting and Cleaning The Data

Data is downloaded through provided URLs. There are training and test data sets. All the missing values will be converted into NAs.Timestamp and username columns will be removed since they don't add any value to the analysis. Character values in column new_window will be converted 1(for yes) and 0(for no). There are some columns with lots of missing values. These columns will be ignored since missing values makes certain models harder to run.

```
# Reading and processing training data
trainData <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", na.string
trainData_clean <- trainData[-c(1:5)]
trainData_clean$new_window <- ifelse(trainData_clean$new_window =="yes",1,0)
trainData_clean <- trainData_clean[,colSums(is.na(trainData_clean))==0]

# Reading and processing test data
testData <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", na.strings=
testData_clean <- testData[-c(1:5)]
testData_clean$new_window <- ifelse(testData_clean$new_window =="yes",1,0)
testData_clean <- testData_clean[,colSums(is.na(testData_clean)) == 0]
```

Exploratory Data Analysis

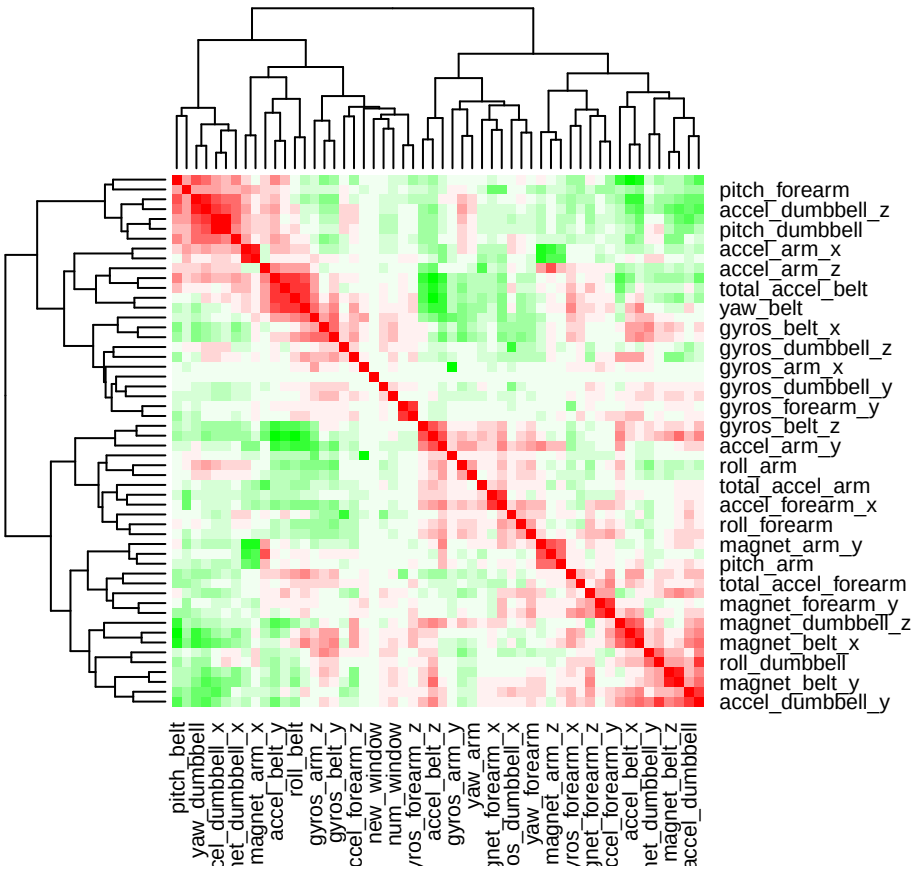
```
dim(trainData_clean)
```

```
## [1] 19622    55
```

```
dim(testData_clean)
```

```
## [1] 20 55
```

```
# Install necessary libraries for correlation plotting
if(!require("corrplot")) install.packages("corrplot"); library("corrplot")
mydata.cor = cor(trainData_clean[,-55], method=c("spearman"))
if(!require("Hmisc")) install.packages("Hmisc"); library("Hmisc")
palette = colorRampPalette(c("green", "white", "red")) (20)
heatmap(x = mydata.cor, col = palette, symm = TRUE)
```



Counting classe frequency.

```
if(!require("plyr")) install.packages("plyr"); library("plyr")
count(trainData_clean, "classe")
```

```
##   classe freq
```

```
## 1      A 5580
## 2      B 3797
## 3      C 3422
## 4      D 3216
## 5      E 3607
```

Partitioning the Data for cross-validation

For cross-validation, the data is separated into training and test sets with ration 70% and 30% respectively.

```
if(!require("ggplot2")) install.packages("ggplot2"); library("ggplot2")
if(!require("caret")) install.packages("caret"); library("caret")
inTraining <- createDataPartition(trainData_clean$classe, p=0.7, list=FALSE)
training <- trainData_clean[inTraining,]
testing <- trainData_clean[-inTraining,]
```

Building a Prediction Model

Random Forests with Cross-validation 10-fold

Building a model with Random Forests is processor exhaustive; therefore, we will be using parallel processing to increase the performance and shorten the length of processing time. We are using 10-fold Cross-Validation.
Step 1: Configure parallel processing

```
library(parallel)
library(doParallel)
cluster <- makeCluster(detectCores() -1)
registerDoParallel(cluster)
```

Step 2: Configure trainControl object

```
fitControl <- trainControl(method = "cv", number = 10, allowParallel = TRUE)
```

Step 3: Develop training model

```
fit <- train(classe ~., method="rf", data=trainData_clean, trControl = fitControl)
```

Step 4: De-register parellel processing cluster

```
stopCluster(cluster)
registerDoSEQ()
```

Using predictive model on the testing data.

```
pred <- predict(fit, testing)
```

Looking at the confusion matrix to see the accuracy of the predictive model on the testing data. As it can be from the testing data, the model accurately predicts the classe variable in the testing data; however, we should be aware of an overfitting issue. We have another data set that we can use to test our model on.

```
conf <- confusionMatrix(pred, as.factor(testing$classe))
conf$table; conf$overall[1]
```

```
##           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
```

```
## Accuracy
##        1
```

Based on the Random Forest model, **the out-of-sample error** is virtually zero and the model predicts the dependent values in the test data perfectly.

Decisision Tree

```
if(!require("rattle")) install.packages("rattle"); library("rattle")
```

```
fit_rpart <- train(classe ~., method="rpart", data=training)
```

```
pred2 <- predict(fit_rpart, testing)
```

According to the confusion matrix, Decision Tree model is not a successful prediction model.

```
conf2 <- confusionMatrix(pred2, as.factor(testing$classe))
conf2$table;conf2$overall[1]
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1505  498  193  334   84
##          B   42  353   37  140  196
##          C  124  288  796  439  173
##          D    0    0    0    0    0
##          E    3    0    0   51  629
```

```
## Accuracy
## 0.557859
```

Based on this model, **the-out-of-sample error** is approximately 0.59 or 59%.

Gradient Boosting with cross-validation

```
fit_gbm <- train(classe ~ ., data=training, method="gbm", verbose=F,
    trControl=trainControl(method="cv", number=2, allowParallel=T))
```

```
pred3 <- predict(fit_gbm, testing)
```

```
conf3 <- confusionMatrix(pred3, as.factor(testing$classe))
conf3$table; conf3$overall[1]
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1671   11    0    0    0
##          B    3 1121    4    3    2
##          C    0    7 1019   10    2
##          D    0    0    3  950   10
##          E    0    0    0    1 1068
```

```
##  Accuracy
## 0.9904843
```

The **out-of-sample** error is pretty small, 2%.

Final Model

Both gradient boosting and Random Forests are a good predictor models for the test data set. If we use these models on the test sample provided with no classification values, here are the predictions by both models.

```
final_pred_gbm <- predict(fit_gbm, testData_clean)
final_pred_rf <- predict(fit, testData_clean)
```

Predictions by gradient boosting:

```
final_pred_gbm
```

```
##  [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
```

Predictions by random forests:

```
final_pred_rf
```

```
##  [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
```