PML Assignment

Kuldeep Singh Meena

August 26, 2019

Executive Summary

Based on a dataset provide by HAR http://groupware.les.inf.puc-rio.br/har we will try to train a predictive model to predict what exercise was performed using a dataset with 159 features

Our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participant They were asked to perform barbell lifts correctly and incorrectly in 5 different ways to build a predictive model. Referring to the research project HAR published on http://groupware.les.inf.puc-rio.br/har, five ways are exactly categoried by five classes which are 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). In a word, we will use the data from accelerometers to predict the class with some predictive models. 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).

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

Download the Data

```r
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainFile <- "./data/pml-training.csv"
testFile  <- "./data/pml-testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile=trainFile, method="curl")
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile=testFile, method="curl")
}
```

Read the Data

After downloading the data from the data source, we can read the two csv files into two data frames.

```r
trainRaw <- read.csv("./data/pml-training.csv")
testRaw <- read.csv("./data/pml-testing.csv")
dim(trainRaw)
```

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

```
dim(testRaw)
```

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

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables. The "classe" variable in the training set is the outcome to predict.

Clean the data

In this step, we will clean the data and get rid of observations with missing values as well as some meaningless variables.

```
sum(complete.cases(trainRaw))
```

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

First, we remove columns that contain NA missing values.

```
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
```

Next, we get rid of some columns that do not contribute much to the accelerometer measurements.

```
classe <- trainRaw$classe
trainRemove <- grepl("^X|timestamp|window", names(trainRaw))
trainRaw <- trainRaw[, !trainRemove]
trainCleaned <- trainRaw[, sapply(trainRaw, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testRaw))
testRaw <- testRaw[, !testRemove]
testCleaned <- testRaw[, sapply(testRaw, is.numeric)]
```

Now, the cleaned training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables. The "classe" variable is still in the cleaned training set.

Slice the data

Then, we can split the cleaned training set into a pure training data set (70%) and a validation data set (30%). We will use the validation data set to conduct cross validation in future steps.

```
set.seed(22519) # For reproducibile purpose
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Data Modeling

We fit a predictive model for activity recognition using **Random Forest** algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general. We will use **5-fold cross validation** when applying the algorithm.

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlRf, ntree=250)
modelRf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10988, 10989, 10989, 10991, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9912654  0.9889499
##   27    0.9916291  0.9894104
##   52    0.9842766  0.9801110
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Then, we estimate the performance of the model on the validation data set.

```
predictRf <- predict(modelRf, testData)
confusionMatrix(testData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    2    3    0    0
##          B    5 1130    3    1    0
##          C    0    4 1019    3    0
##          D    0    0   10  954    0
##          E    0    0    4    2 1076
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.9913, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9947   0.9808   0.9938   1.0000
## Specificity            0.9988   0.9981   0.9986   0.9980   0.9988
## Pos Pred Value         0.9970   0.9921   0.9932   0.9896   0.9945
## Neg Pred Value         0.9988   0.9987   0.9959   0.9988   1.0000
## Prevalence             0.2845   0.1930   0.1766   0.1631   0.1828
## Detection Rate         0.2836   0.1920   0.1732   0.1621   0.1828
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9979   0.9964   0.9897   0.9959   0.9994
```

```
accuracy <- postResample(predictRf, testData$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9937128 0.9920477
```

```
oose <- 1 - as.numeric(confusionMatrix(testData$classe, predictRf)$overall[1])
oose
```

```
## [1] 0.006287171
```

So, the estimated accuracy of the model is 99.42% and the estimated out-of-sample error is 0.58%.

Predicting for Test Data Set

Now, we apply the model to the original testing data set downloaded from the data source. We remove the `problem_id` column first.

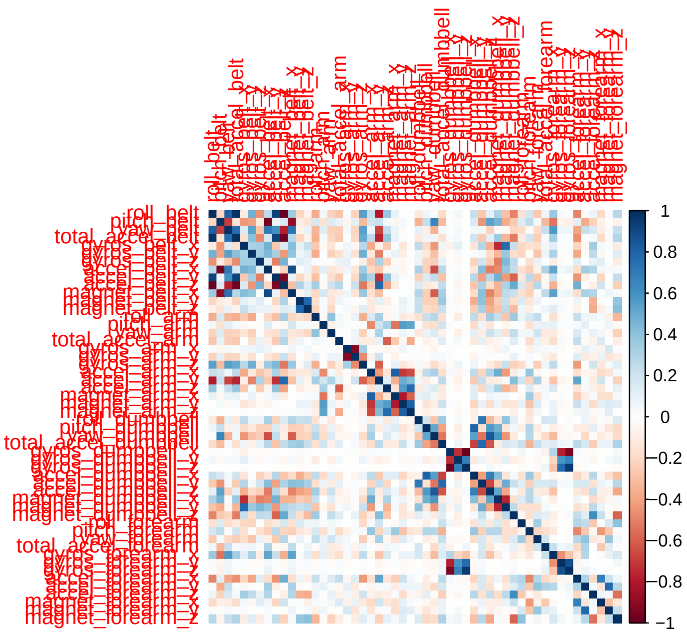
```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

Appendix: Figures

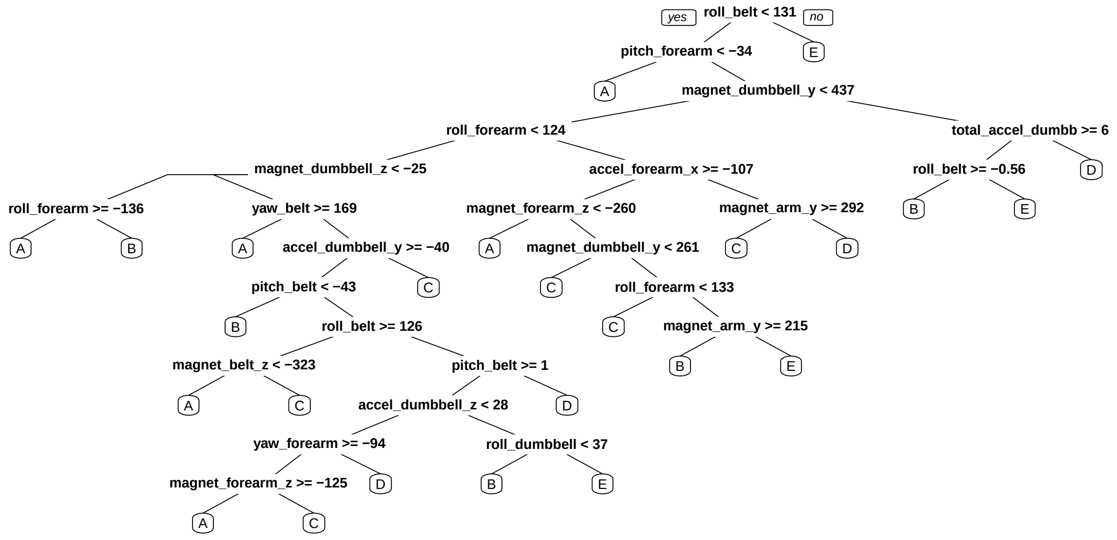
1. Correlation Matrix Visualization

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



2. Decision Tree Visualization

```r
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel) # fast plot
```


yes
roll_belt < 131
no
pitch_forearm < −34
E
A
magnet_dumbbell_y < 437
roll_forearm < 124
total_accel_dumbb >= 6
magnet_dumbbell_z < −25
accel_forearm_x >= −107
roll_belt >= −0.56
D
roll_forearm >= −136
yaw_belt >= 169
magnet_forearm_z < −260
magnet_arm_y >= 292
B
E
A
B
A
accel_dumbbell_y >= −40
A
magnet_dumbbell_y < 261
C
D
pitch_belt < −43
C
C
roll_forearm < 133
B
roll_belt >= 126
C
magnet_arm_y >= 215
magnet_belt_z < −323
pitch_belt >= 1
B
E
A
C
accel_dumbbell_z < 28
D
yaw_forearm >= −94
roll_dumbbell < 37
magnet_forearm_z >= −125
D
B
E
A
C