Practical Machine Learning Final Project

Mahmoud Samy

December 19, 2018

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

In this project, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise.

Data Preprocessing

```
library(caret)
```

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

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

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

```
## randomForest 4.6-14
```

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

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

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Download the Data

```
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

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

## [1] 19622   160

dim(testRaw)

## [1]  20 160
```

from previous Output 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 that we will predict.

Clean the data

In this step, we will clean the data and Removing the Observation that containing missing values as well as removing variables that i don't care about.

```
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 remove some columns that contain measurements not useful for us .

```
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 next steps.

```
set.seed(22123)
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. We will use 5-fold cross validation when using 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: 10989, 10990, 10989, 10989, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9915548  0.9893164
##   27    0.9905359  0.9880269
##   52    0.9845668  0.9804751
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

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 1672    2    0    0    0
##          B    4 1133    2    0    0
##          C    0    7 1018    1    0
##          D    0    0   10  954    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2848
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9921   0.9883   0.9979   1.0000
## Specificity            0.9995   0.9987   0.9984   0.9980   0.9998
## Pos Pred Value         0.9988   0.9947   0.9922   0.9896   0.9991
## Neg Pred Value         0.9991   0.9981   0.9975   0.9996   1.0000
## Prevalence             0.2848   0.1941   0.1750   0.1624   0.1837
## Detection Rate         0.2841   0.1925   0.1730   0.1621   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9986   0.9954   0.9934   0.9979   0.9999

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

##  Accuracy     Kappa
## 0.9954121 0.9941964

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

## [1] 0.004587935
```

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.

```
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
```

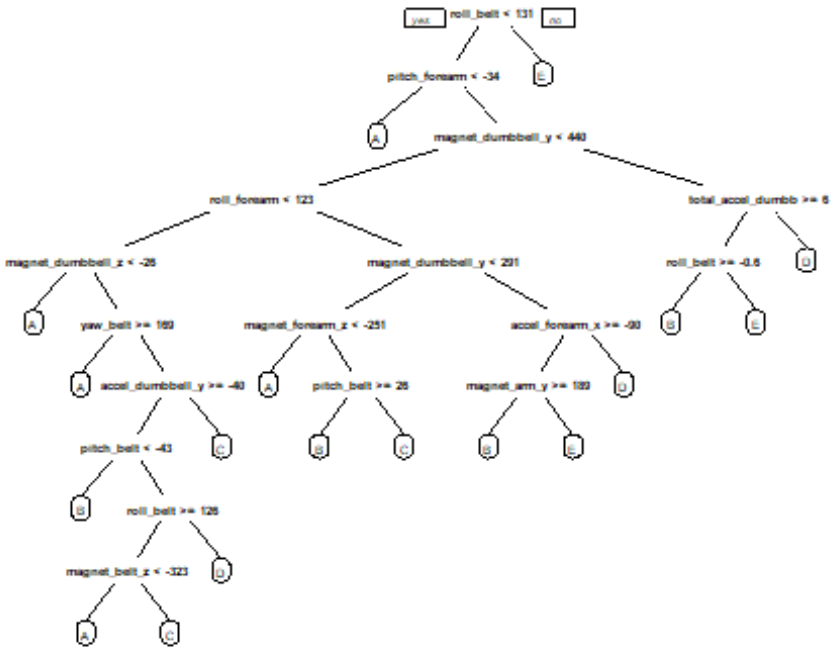
Our Figures

1. Correlation Matrix Visualization

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

2. Decision Tree Visualization

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


yes
roll_belt < 131
no
pitch_forearm < -34
E
A
magnet_dumbbell_y < 440
roll_forearm < 123
total_accel_dumbb >= 6
magnet_dumbbell_z < -26
magnet_dumbbell_y < 291
roll_belt >= -0.6
D
A
yaw_belt >= 169
magnet_forearm_z < -251
accel_forearm_x >= -90
B
E
A
accel_dumbbell_y >= -40
A
pitch_belt >= 26
magnet_arm_y >= 199
D
pitch_belt < -43
C
B
C
B
E
B
roll_belt >= 126
magnet_belt_z < -323
D
A
C