Practical Machine Learning Project - Prediction

Kassem Saleh

May 26, 2016

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.

```
require(caret)
require(corrplot)
require(rpart)
require(rpart.plot)
require(Rtsne)
require(xgboost)
require(stats)
require(knitr)
require(ggplot2)
knitr::opts_chunk$set(cache=TRUE, echo = TRUE)
```

Download the Data if they don't exist

```
trainingUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv"
testingUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.
csv"
trainingFile <- "./pml-training.csv"
testingFile  <- "./pml-testing.csv"

if (!file.exists(trainingFile)) {
  download.file(trainingUrl, destfile=trainingFile, method="curl")
}
if (!file.exists(testingFile)) {
  download.file(testingUrl, destfile=testingFile, method="curl")
}
```

Reading the data into data frames

```
trainingRaw <- read.csv("./pml-training.csv")
testingRaw <- read.csv("./pml-testing.csv")
dim(trainingRaw)
```

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

```
dim(testingRaw)
```

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

Cleaning the data

removing missing values and meaningless variables.

```
sum(complete.cases(trainingRaw))
```

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

First, remove columns containing missing values.

```
trainingRaw <- trainingRaw[, colSums(is.na(trainingRaw)) == 0]
testingRaw <- testingRaw[, colSums(is.na(testingRaw)) == 0]
```

Next, we remove columns that do not contribute to the accelerometer measurements.

```
classe <- trainingRaw$classe
trainingRemove <- grepl("^X|timestamp|window", names(trainingRaw))
trainingRaw <- trainingRaw[, !trainingRemove]
trainingCleaned <- trainingRaw[, sapply(trainingRaw, is.numeric)]
trainingCleaned$classe <- classe
testingRemove <- grepl("^X|timestamp|window", names(testingRaw))
testingRaw <- testingRaw[, !testingRemove]
testingCleaned <- testingRaw[, sapply(testingRaw, is.numeric)]
```

Partition the data

Then, we can partition 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(trainingCleaned$classe, p=0.70, list=F)
trainingData <- trainingCleaned[inTrain, ]
testingData <- trainingCleaned[-inTrain, ]
```

Data Modeling

We fit a predictive model for activity recognition using the randomForest algorithm. We 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)
```

```
## 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
```

```
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, 10989, 10991, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9901727  0.9875673
##   27    0.9917015  0.9895017
##   52    0.9840572  0.9798282
##
## 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, testingData)
confusionMatrix(testingData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    1
##          B    5 1131    3    0    0
##          C    0    0 1021    5    0
##          D    0    0   13  949    2
##          E    0    0    1    6 1075
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9915, 0.9957)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   1.0000   0.9836   0.9885   0.9972
## Specificity            0.9998   0.9983   0.9990   0.9970   0.9985
## Pos Pred Value         0.9994   0.9930   0.9951   0.9844   0.9935
## Neg Pred Value         0.9988   1.0000   0.9965   0.9978   0.9994
## Prevalence             0.2851   0.1922   0.1764   0.1631   0.1832
## Detection Rate         0.2843   0.1922   0.1735   0.1613   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9984   0.9992   0.9913   0.9927   0.9979
```

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

```
##  Accuracy     Kappa
## 0.9938828 0.9922620
```

```
outOfSampleError <- 1 - as.numeric(confusionMatrix(testingData$classe, predictR
f)$overall[1])
outOfSampleError
```

```
## [1] 0.006117247
```

So, the estimated accuracy of the model is 99.38% and the estimated out-of-sample error is 0.61%.

Test Data Set prediction

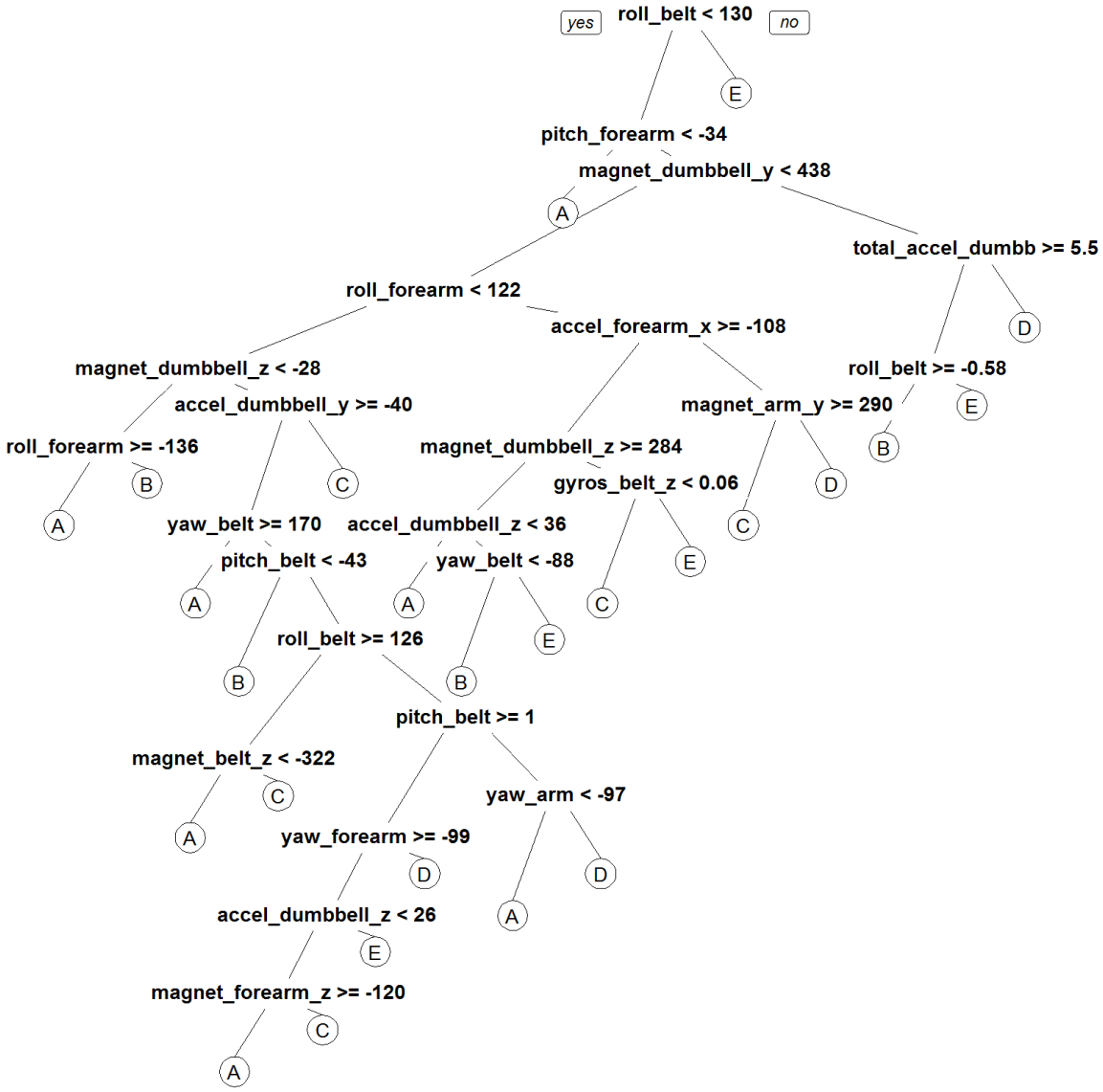
Now, we apply the model to the original testing data set downloaded from the data source.

```
predictionResults <- predict(modelRf, testingCleaned[, -length(names(testingCle
aned))])
predictionResults
```

```
##  [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
```

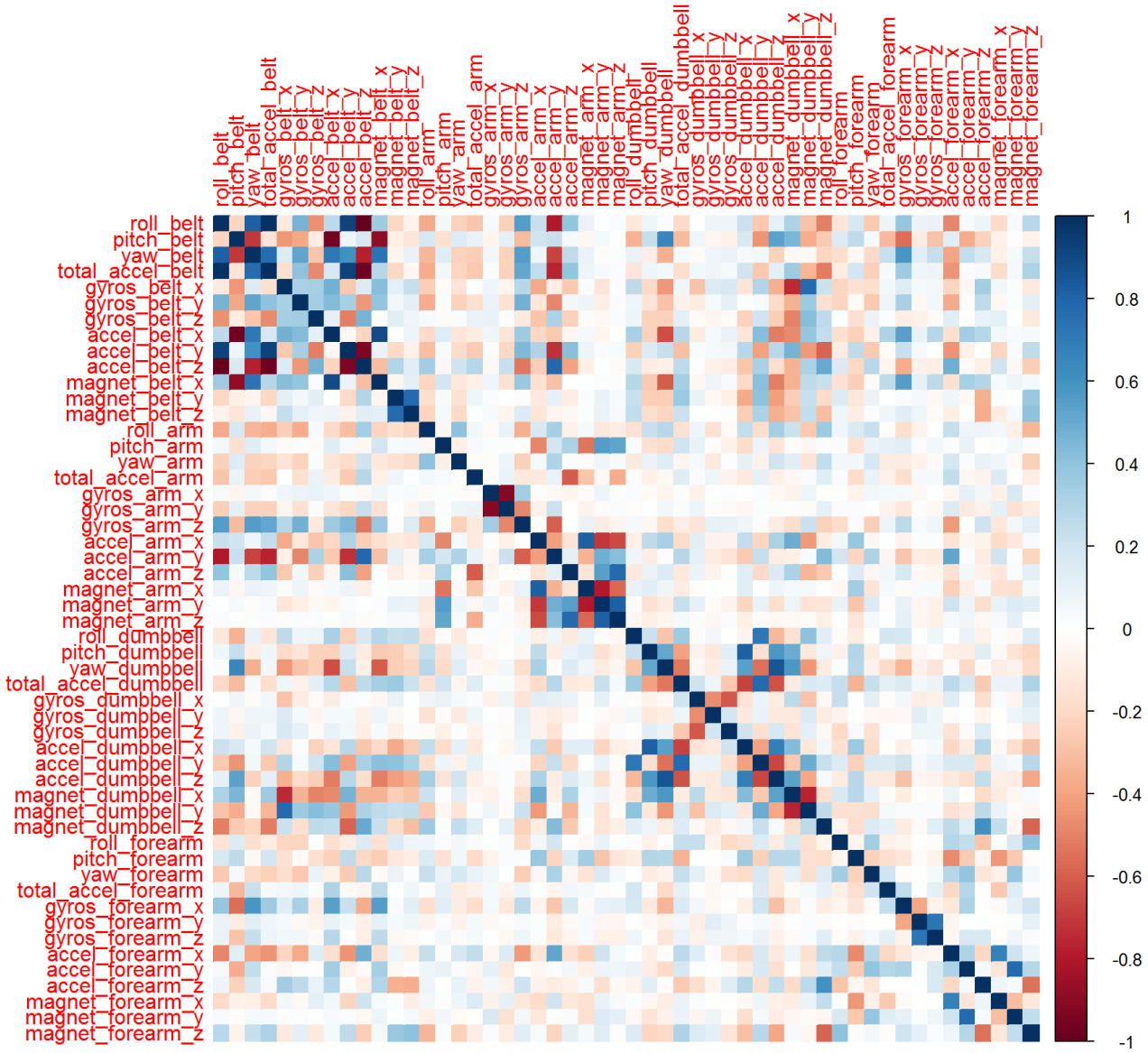
Visualization of Decision Tree

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



Visualization of Correlation Matrix

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


roll_belt
pitch_belt
yaw_belt
total_accel_belt
gyros_belt_x
gyros_belt_y
gyros_belt_z
accel_belt_x
accel_belt_y
accel_belt_z
magnet_belt_x
magnet_belt_y
magnet_belt_z
roll_arm
pitch_arm
yaw_arm
total_accel_arm
gyros_arm_x
gyros_arm_y
gyros_arm_z
accel_arm_x
accel_arm_y
accel_arm_z
magnet_arm_x
magnet_arm_y
magnet_arm_z
roll_dumbbell
pitch_dumbbell
yaw_dumbbell
total_accel_dumbbell
gyros_dumbbell_x
gyros_dumbbell_y
gyros_dumbbell_z
accel_dumbbell_x
accel_dumbbell_y
accel_dumbbell_z
magnet_dumbbell_x
magnet_dumbbell_y
magnet_dumbbell_z
roll_forearm
pitch_forearm
yaw_forearm
total_accel_forearm
gyros_forearm_x
gyros_forearm_y
gyros_forearm_z
accel_forearm_x
accel_forearm_y
accel_forearm_z
magnet_forearm_x
magnet_forearm_y
magnet_forearm_z
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1