Practical Machine Learning Course Project Report

Rafael Sulino

11/08/2020

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, your goal will 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. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

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
```

Downloading 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"
```

```r
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile=trainFile, method="libcurl")
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile=testFile, method="libcurl")
}
```

Reading Data

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
```

```r
dim(testRaw)
```

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

Training dataset:

- 19622 observations
- 160 variables

Testing dataset:

- 20 observations
- 160 variables

Cleaning data

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

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

Removing missing values.

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

Removing some columns (not used).

```r
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)]
```

Training dataset (cleaned):

- 19622 observations
- 53 variables

Testing dataset (cleaned):

- 20 observations
- 54 variables

Spliting Data

- Training = 70%
- Validation = 30%

```
set.seed(1234)
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Modeling Data

Random Forest, 5-fold cross validation.

```
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.
```

Model performance.

```
testdData <- testData[complete.cases(testData),]
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    5    0    0    0
##          B    2 1130    4    0    0
##          C    3    3 1019   10    4
##          D    0    1    3  954    2
##          E    0    0    0    0 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   0.9921   0.9932   0.9896   0.9945
## Specificity            0.9988   0.9987   0.9959   0.9988   1.0000
## Pos Pred Value         0.9970   0.9947   0.9808   0.9937   1.0000
## Neg Pred Value         0.9988   0.9981   0.9986   0.9980   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1920   0.1732   0.1621   0.1828
## Detection Prevalence   0.2845   0.1930   0.1766   0.1631   0.1828
## Balanced Accuracy      0.9979   0.9954   0.9945   0.9942   0.9972
```

```r
accuracy <- postResample(predictRf, as.factor(testData$classe))
accuracy
```

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

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

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

Predicting for Test dataset

Apply the model to the original testing dataset.

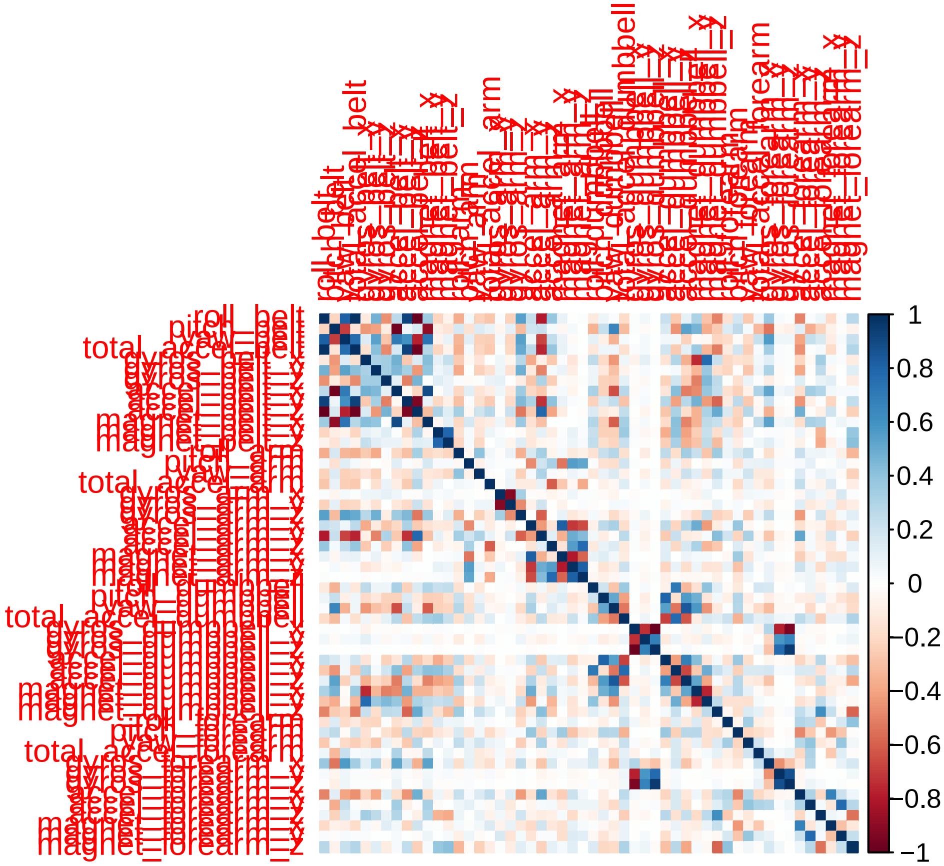
```r
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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

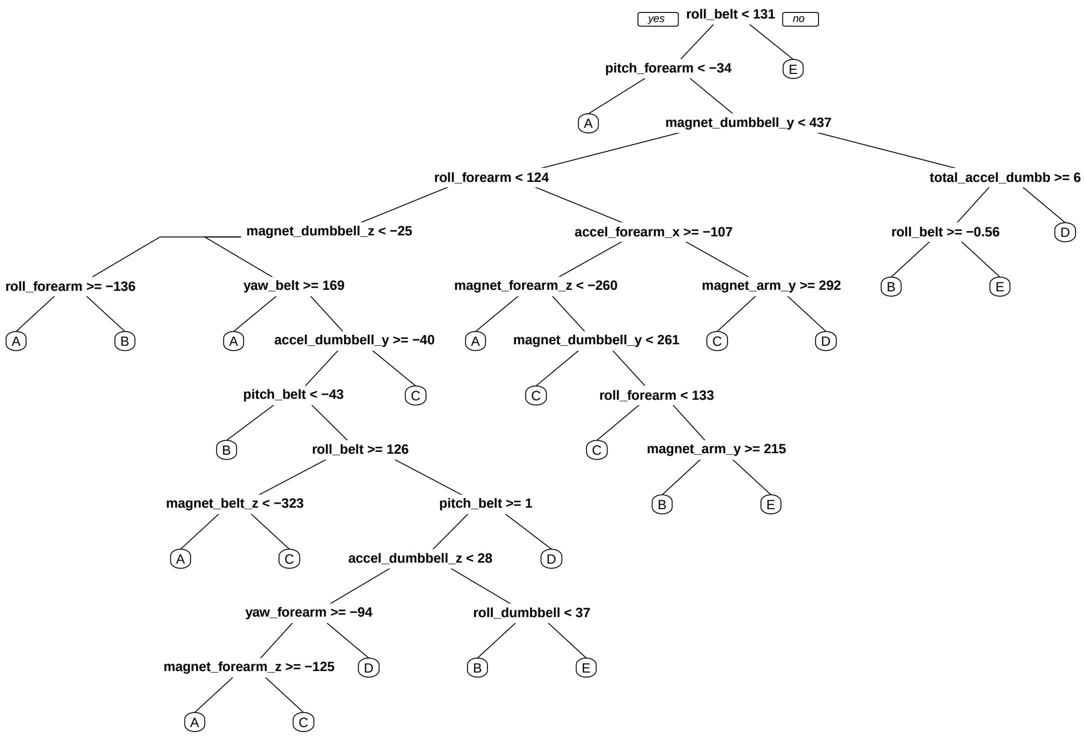
1. Correlation Matrix

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



2. Decision Tree

```
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