Practical Machine Learning Course Project Wwriteup

Maumita Niyogi

Sunday, October 25, 2015

Summary

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, our goal is 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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The goal of our project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. We may use any of the other variables to predict with. We have to create a report describing 1) how we built our model 2) how we used cross validation 3) what we think the expected out of sample error is 4) why we made the choices we did.

Data Preprocessing

```
library(caret)
```

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

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

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(corrplot)
```

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Source: http://groupware.les.inf.puc-rio.br/har.

Read Data

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

```r
trainRaw <- read.csv("pm1training.csv")
testRaw <- read.csv("pm1testing.csv")
dim(trainRaw)
```

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

```r
dim(testRaw)
```

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

The training data set contains 19622 observations and 160 variables. 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

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

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

First, we remove columns that contain NA missing values.

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

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

```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)]
dim(trainCleaned)
```

```
## [1] 19622    53
```

```r
dim(testCleaned)
```

```
## [1] 20 53
```

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

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.

```r
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 . We will use 5-fold cross validation when applying the algorithm.

```r
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, 10989, 10991, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9910462  0.9886729  0.001301269  0.001647776
##   27    0.9914102  0.9891334  0.001717547  0.002174708
##   52    0.9850037  0.9810264  0.002718384  0.003439965
##
## 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.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    1
##          B    7 1128    4    0    0
##          C    0    0 1021    5    0
##          D    0    0   14  949    1
##          E    0    0    1    7 1074
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9906, 0.995)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9991   0.9817   0.9875   0.9981
## Specificity            0.9995   0.9977   0.9990   0.9970   0.9983
## Pos Pred Value         0.9988   0.9903   0.9951   0.9844   0.9926
## Neg Pred Value         0.9983   0.9998   0.9961   0.9976   0.9996
## Prevalence             0.2853   0.1918   0.1767   0.1633   0.1828
## Detection Rate         0.2841   0.1917   0.1735   0.1613   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9977   0.9984   0.9903   0.9922   0.9982
```

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

```
##  Accuracy     Kappa
## 0.9930331 0.9911872
```

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

```
## [1] 0.006966865
```

So, the estimated accuracy of the model is 99.42% and the estimated out-of-sample error is 0.70%.

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.

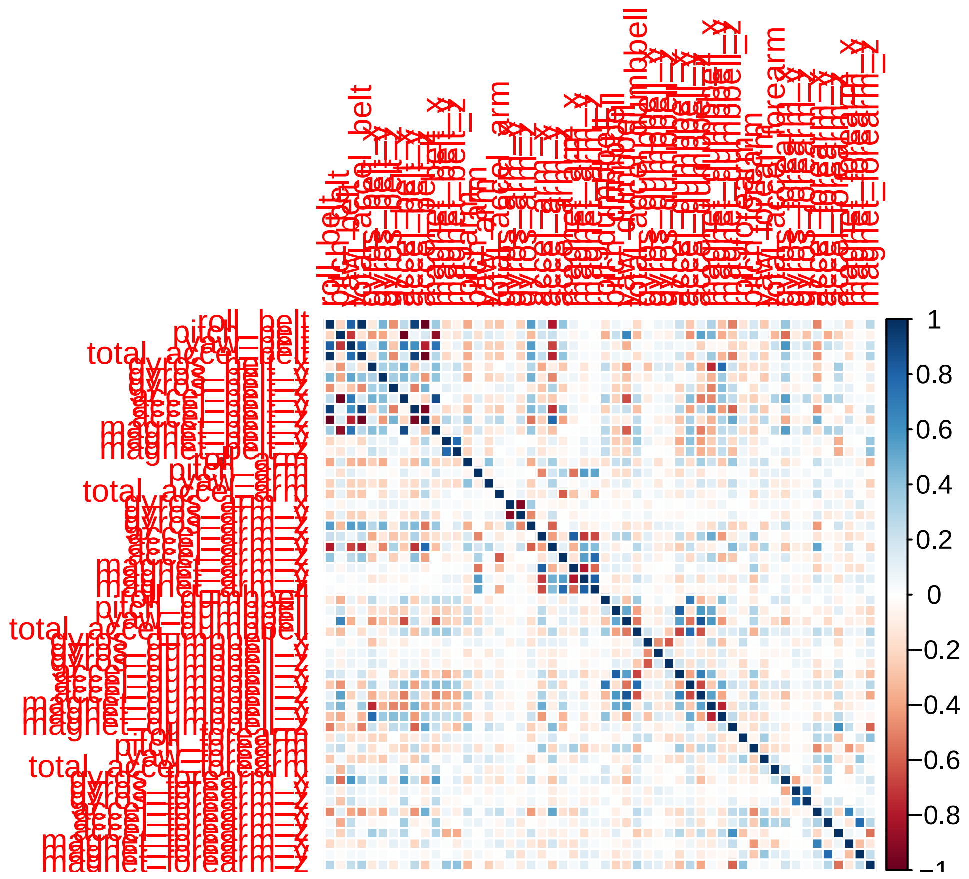
```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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

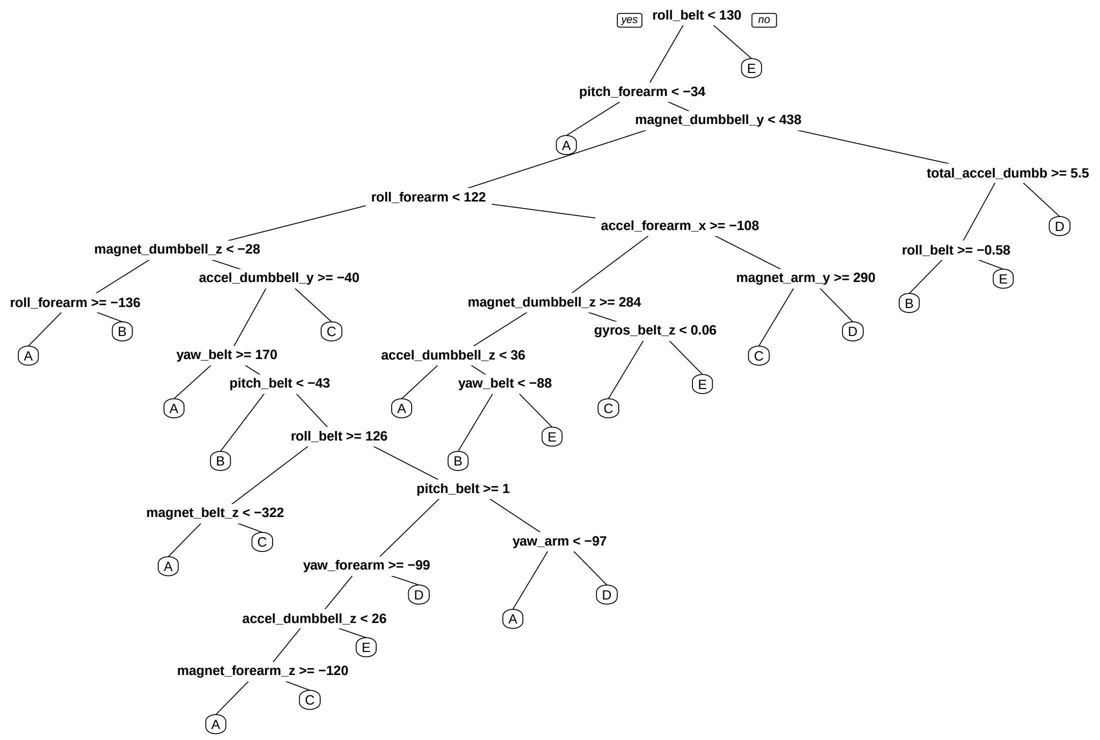
Correlation Matrix Visualization

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



Decision Tree Visualization

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


yes
roll_belt < 130
no
E
pitch_forearm < −34
A
magnet_dumbbell_y < 438
roll_forearm < 122
accel_forearm_x >= −108
total_accel_dumbb >= 5.5
D
magnet_dumbbell_z < −28
accel_dumbbell_y >= −40
roll_forearm >= −136
A
B
C
yaw_belt >= 170
A
pitch_belt < −43
B
roll_belt >= 126
magnet_belt_z < −322
A
C
pitch_belt >= 1
yaw_forearm >= −99
D
accel_dumbbell_z < 26
E
magnet_forearm_z >= −120
A
C
yaw_arm < −97
A
D
magnet_dumbbell_z >= 284
accel_dumbbell_z < 36
A
yaw_belt < −88
B
E
gyros_belt_z < 0.06
C
E
magnet_arm_y >= 290
C
D
roll_belt >= −0.58
B
E