Practical Machine Learning Project

Gabriel Nóbrega

Monday, November 16, 2015

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

The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Libraries

Here we are going to load the necessary libraries for the project.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

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

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

```
## Warning: package 'rpart.plot' was built under R version 3.1.3
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.1.3
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.1.3
```

Downloading the data

Here we are going to download the datasets.

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

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

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

```
dim(testRaw)
```

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

Cleaning and dividing the data

First, lets check how many complete cases we have

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

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

Here we will remove the NAs columns

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

Here we will cut off some variables that are not relevant

```
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)]
```

In this part, we will divide the training set into a 70% pure training and a 30% validation set.

```
set.seed(34568)
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Modeling the data

We will use a prediction model based on the Random Forest method

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlRf, ntree=25
0)
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: 10990, 10988, 10990, 10990, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9903908  0.9878432  0.0008397278  0.001062062
##   27    0.9904635  0.9879362  0.0015965126  0.002019572
##   52    0.9852948  0.9813971  0.0038849205  0.004916946
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Here we can estimate the performance of the model on the validation set

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

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    1    0    0
##          B    5 1131    3    0    0
##          C    0    6 1015    5    0
##          D    0    0    4  958    2
##          E    0    0    0    3 1079
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9949          
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.285           
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9936          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9938   0.9922   0.9917   0.9981
## Specificity            0.9995   0.9983   0.9977   0.9988   0.9994
## Pos Pred Value         0.9988   0.9930   0.9893   0.9938   0.9972
## Neg Pred Value         0.9988   0.9985   0.9984   0.9984   0.9996
## Prevalence             0.2850   0.1934   0.1738   0.1641   0.1837
## Detection Rate         0.2841   0.1922   0.1725   0.1628   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9983   0.9961   0.9950   0.9952   0.9988
```

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

```
##  Accuracy     Kappa 
## 0.9949023 0.9935517
```

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

```
## [1] 0.005097706
```

Modeling the data

Applying the model to the original test set

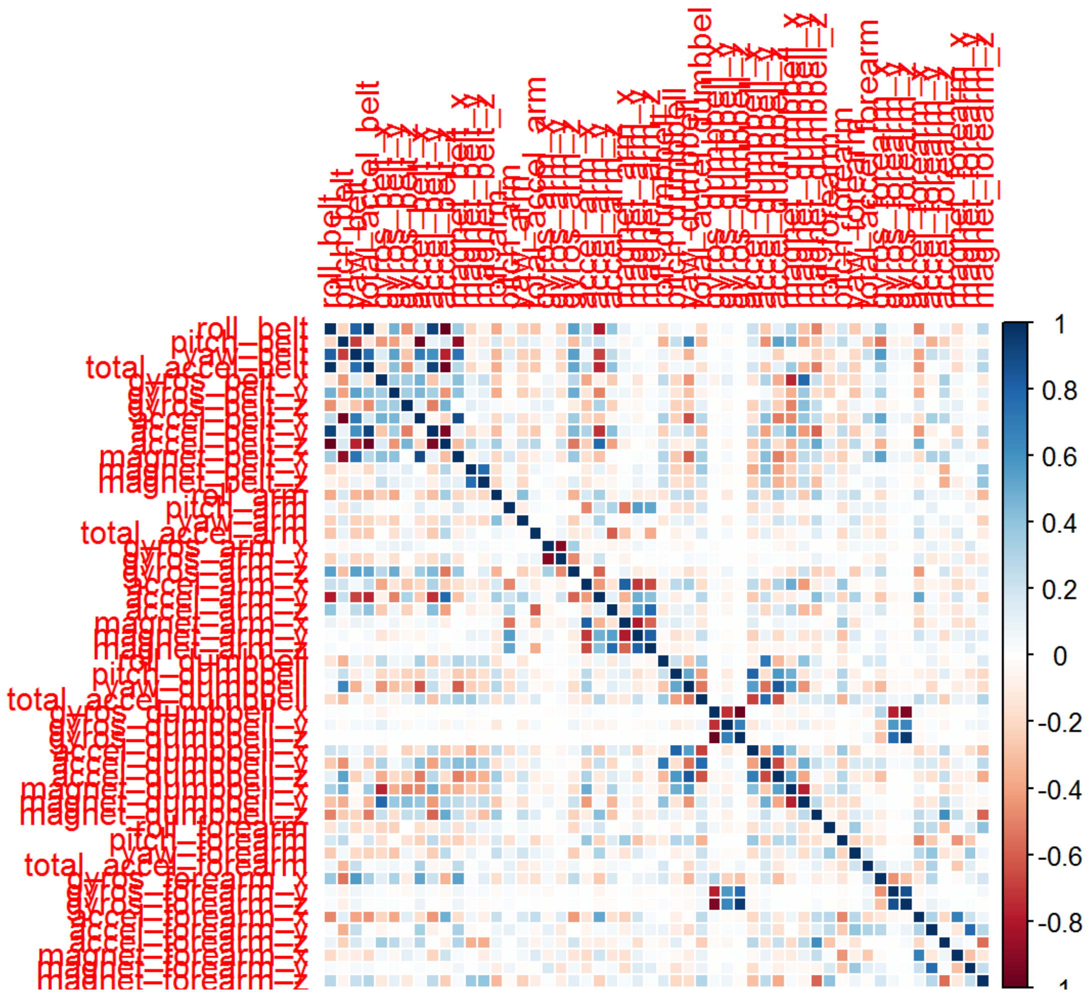
```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

Figures

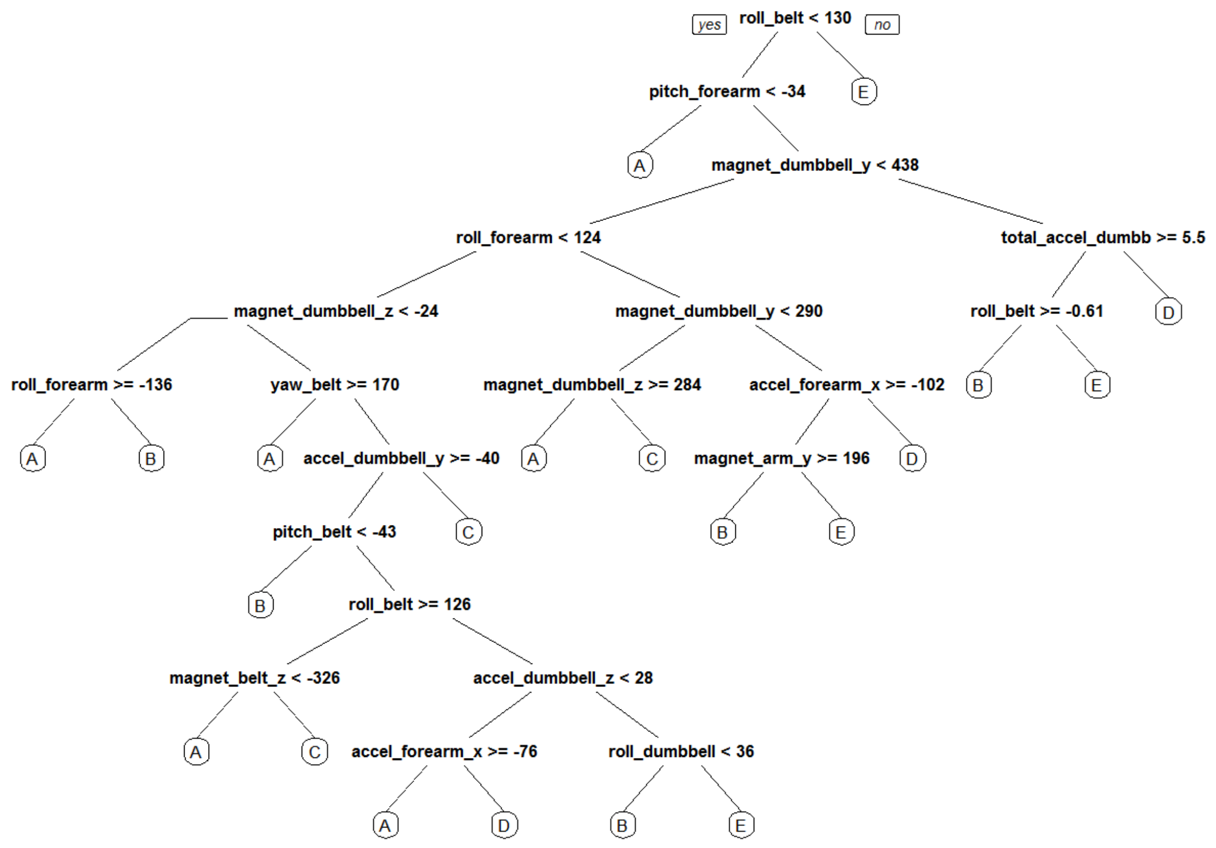
Correlation Matrix Visualization

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



Tree Visualization

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


yes
roll_belt < 130
no
pitch_forearm < -34
E
A
magnet_dumbbell_y < 438
roll_forearm < 124
total_accel_dumbb >= 5.5
magnet_dumbbell_z < -24
magnet_dumbbell_y < 290
roll_belt >= -0.61
D
roll_forearm >= -136
yaw_belt >= 170
magnet_dumbbell_z >= 284
accel_forearm_x >= -102
B
E
A
B
A
accel_dumbbell_y >= -40
A
C
magnet_arm_y >= 196
D
pitch_belt < -43
C
B
E
B
roll_belt >= 126
magnet_belt_z < -326
accel_dumbbell_z < 28
A
C
accel_forearm_x >= -76
roll_dumbbell < 36
A
D
B
E