Practical Machine Learning Course Project

Paulo Cardoso GitHub

March 26, 2016

Introduction

Using devices such as JawboneUp, NikeFuelBand, and Fitbitit 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, the 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: http://groupware.les.inf.puc-rio.br/har

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

Set up

This first stage the Set UP is characterized by the preparation of the environment and data. Therefore the following steps will be performed:

- Loading of the libraries that will be used in this project
- Loading the train and test dataset
- Data Cleaning
- Data Slicing

```
# Loading Libraries
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.
##
## Attaching package: 'randomForest'
```

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

```r
library(corrplot)
library(data.table)
```

Loading the data.

```r
# Loading Data
train = read.csv("pml-training.csv")
test = read.csv("pml-testing.csv")

# Dimensions
dim(train)
```

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

```r
dim(test)
```

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

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 to predict.

```r
# Data Cleaning
# The First step is the removal od the columns that contain NA values
trainNA <- train[, colSums(is.na(train)) == 0]
testNA <- test[, colSums(is.na(test)) == 0]
# The Second step is the removal of some columns that do not contribute to the accelerometer measuremen
classe <- trainNA$classe
trainRem <- grepl("^X|timestamp|window", names(trainNA))
trainNA <- trainNA[, !trainRem]
trainClean <- trainNA[, sapply(trainNA, is.numeric)]
trainClean$classe <- classe
testRem <- grepl("^X|timestamp|window", names(testNA))
testNA <- testNA[, !testRem]
testClean <- testNA[, sapply(testNA, is.numeric)]

dim(trainClean)
```

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

```r
dim(testClean)
```

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

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.

```r
# Data Slicing
trainPart <- createDataPartition(trainClean$classe, p=0.66, list=F)
trainData <- trainClean[trainPart, ]
testData <- trainClean[-trainPart, ]
```

Then, we can split the cleaned training set into a pure training data set (66%) and a validation data set (34%). We will use the validation data set to conduct cross validation in future steps.

Data Modeling

Model 1: Random Forest

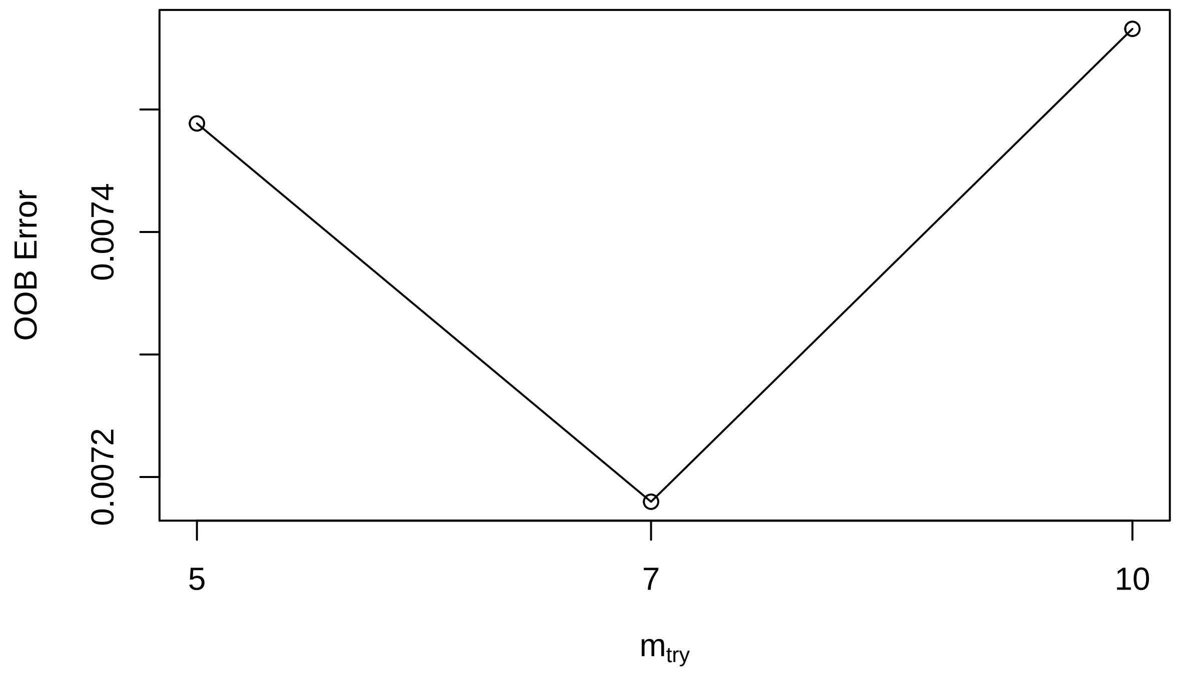
Regarding the data, we will expect Decision Tree and Random Forest to give the best results. We start with Random Forest. First we set a seed to make this project reproducable. We will use the tuneRF function to calculate the optimal mtry and use that in the random forest function.

```r
# Seed for reproducibility
set.seed(12345)

# Finding best mtry
last <- as.numeric(ncol(trainClean))
prior <- last - 1

bestmtry <- tuneRF(trainData[-last],trainData$classe, ntreeTry=100, stepFactor=1.5, improve=0.01, trace
```

```
## mtry = 7  OOB error = 0.72%
## Searching left ...
## mtry = 5     OOB error = 0.75%
## -0.04301075 0.01
## Searching right ...
## mtry = 10    OOB error = 0.76%
## -0.05376344 0.01
```



```
mtry <- bestmtry[as.numeric(which.min(bestmtry[,"OOBError"])),"mtry"]

# Random Forest
trainRF <- randomForest(classe~.,data=trainData, mtry=mtry, ntree=501, keep.forest=TRUE, proximity=TRUE

# Accuracy for trainData
pred1Model1 <- predict(trainRF, newdata=trainData)
confusionMatrix(pred1Model1,trainData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3683    0    0    0    0
##          B    0 2507    0    0    0
##          C    0    0 2259    0    0
##          D    0    0    0 2123    0
##          E    0    0    0    0 2381
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
# Accuracy for testData
pred2Model1 <- predict(trainRF, newdata=testData)
confusionMatrix(pred2Model1,testData$classe)
```

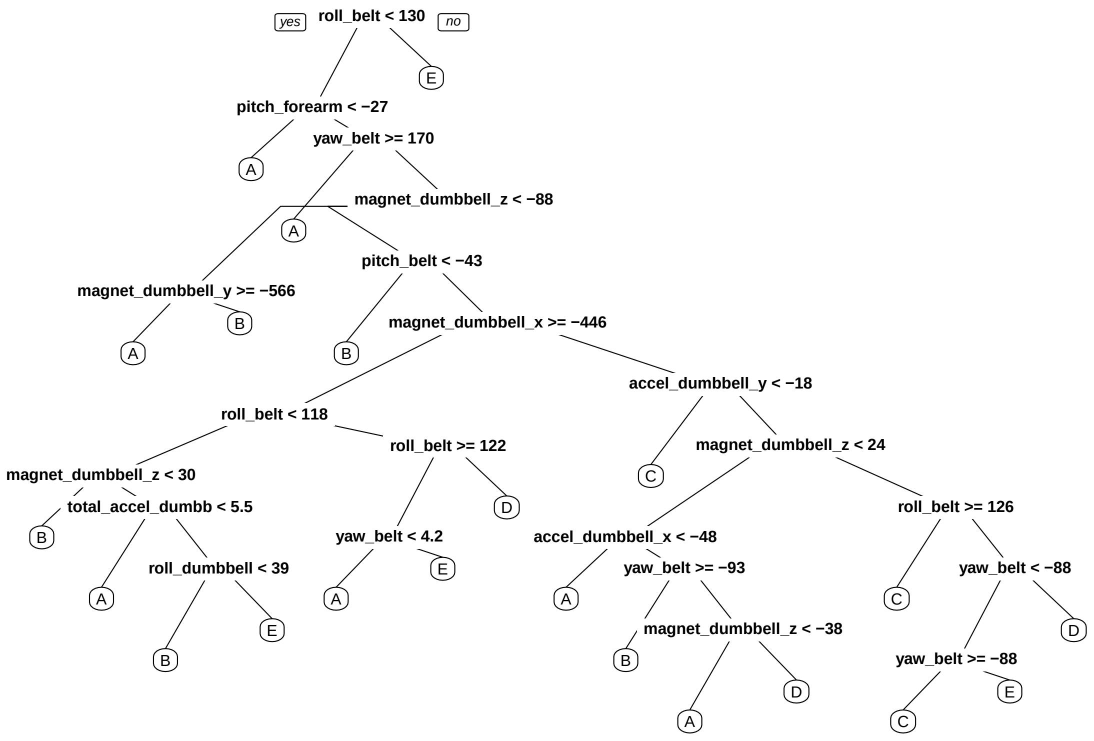
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1896    6    0    0    0
##          B    0 1275    4    0    0
##          C    0    9 1158   11    2
##          D    0    0    1 1080    1
##          E    1    0    0    2 1223
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.9924, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.993
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9995   0.9884   0.9957   0.9881   0.9976
## Specificity            0.9987   0.9993   0.9960   0.9996   0.9994
## Pos Pred Value         0.9968   0.9969   0.9814   0.9982   0.9976
## Neg Pred Value         0.9998   0.9972   0.9991   0.9977   0.9994
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1912   0.1736   0.1619   0.1834
## Detection Prevalence   0.2852   0.1918   0.1769   0.1622   0.1838
## Balanced Accuracy      0.9991   0.9938   0.9959   0.9939   0.9985
```

Here we use Model 1 to predict both the training as the testing set. With the test set, we obtain an accuracy of 0.9946, which seems to be acceptable. However, we will also test the Decision Tree model.

Model 2: Decision Tree

```r
# Model 2: Decision Tree
trainDT <- rpart(classe ~ ., data=trainData, method="class")

# Ploting Tree
prp(trainDT)
```



```r
# Cross Validation
predModel2 <- predict(trainDT, testData, type = "class")
confusionMatrix(predModel2, testData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670  210   46   59   25
##          B   76  762  105   82   66
##          C   27   98  761   47   51
##          D   91   47  198  723  124
##          E   33  173   53  182  960
##
## Overall Statistics
##
```

```
##                Accuracy : 0.7311
##                  95% CI : (0.7203, 0.7418)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6593
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8803   0.5907   0.6543   0.6615   0.7830
## Specificity            0.9288   0.9388   0.9595   0.9175   0.9190
## Pos Pred Value         0.8308   0.6984   0.7734   0.6112   0.6852
## Neg Pred Value         0.9513   0.9053   0.9293   0.9326   0.9495
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2504   0.1143   0.1141   0.1084   0.1439
## Detection Prevalence   0.3014   0.1636   0.1475   0.1774   0.2101
## Balanced Accuracy      0.9045   0.7648   0.8069   0.7895   0.8510
```

As we can see, this model is not improving the performance, having an accuracy of 0.7323. Therefor, we will continue with model 1.

Results

Finally, as the Random Forest model gave us the best result, we will apply that to our validation set and create the documents to submit.

```
# Predict the class of the validation set
result<-predict(trainRF, testClean[, -length(names(testClean))])
result
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```