Prediction Assignment

Sara Regina Ferreira de Faria

Aug 29, 2018

Introduction

The goal of this report is to find out the class of the exercise using the input data. In other words, we have to figure out how good the person is doing the activity.

The data has 5 classes: * **Class A** - exactly according to the specification

- **Class B** - throwing the elbows to the front
- **Class C** - lifting the dumbbell only halfway
- **Class D** - lowering the dumbbell only halfway
- **Class E** - throwing the hips to the front

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Load data

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",'pml-training.csv')

training <- read.csv('pml-training.csv')

download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",'pml-testing.csv')

testing <- read.csv('pml-testing.csv')
```

Explore data

Get how many data do we have:

```
dim(training)
```

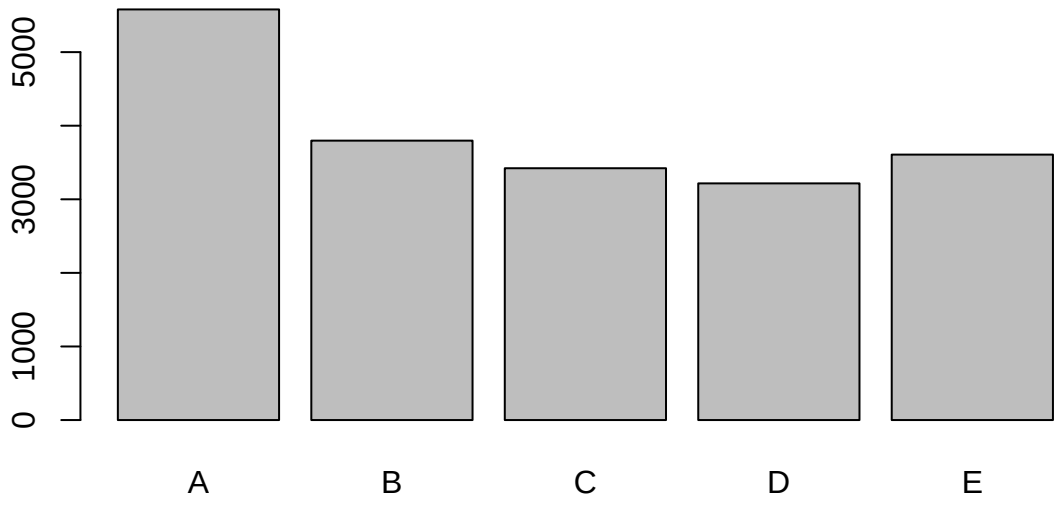
```
## [1] 19622   160
```

```
dim(testing)
```

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

Get an idea of quantity of each class:

```
plot(training$classe)
```



Choose variables

First of all we have to remove those variables that has more NAs or empty values than feasible values.

```
nas = sapply(training, function(x) sum(is.na(x)))
training = training[,nas == 0]
testing = testing[,nas == 0]

empty = sapply(training, function(x) sum(x == ''))
training = training[,empty == 0]
testing = testing[,empty == 0]
```

Then we remove the variables that has no direct impact in the classification of the exercise:

```
names(training[,1:7])
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

```
training = training[,8:60]
testing = testing[,8:60]
```

Train the model

To train the model, we will devide the training data into two sets: one for train the model and the other one to test it.

A Decision Tree and a Random Forest will be used to fit two models.

```
# load libraries
library(caret)
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

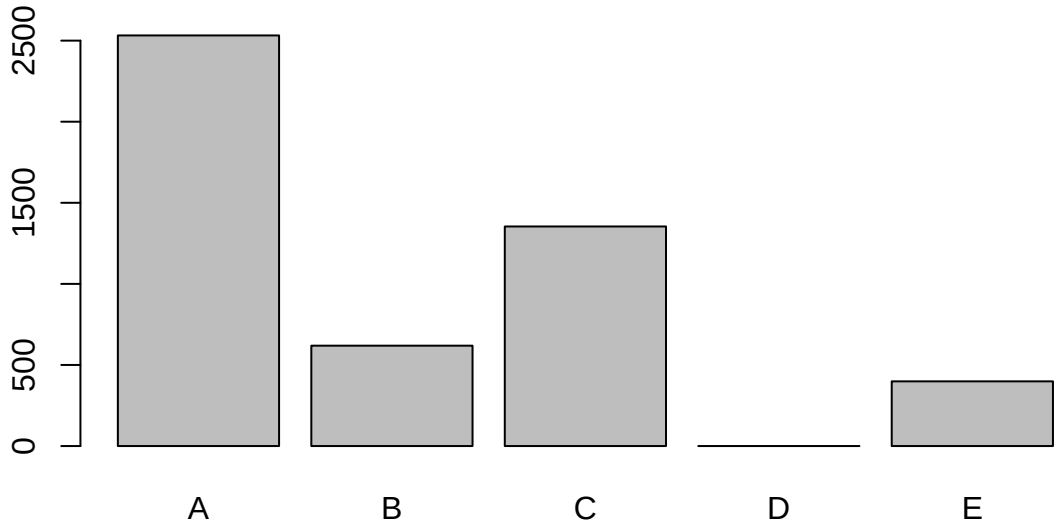
```
##
## Attaching package: 'randomForest'
```

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

```
set.seed(43672)

#Create training and testing set from testing data
train <- createDataPartition(y = training$classe,p = .75, list=F)
trainingSet <- training[train,]
testingSet <- training[-train,]

# train the models
modelRpart <- train(classe ~ . , data=trainingSet, method="rpart")
modelRF <- randomForest(classe ~ .,   data=trainingSet, do.trace=F)
```

Test the rpart model

```
# make predictions
predTest <- predict(modelRpart,testingSet)

# summarize results
plot(predTest)
```



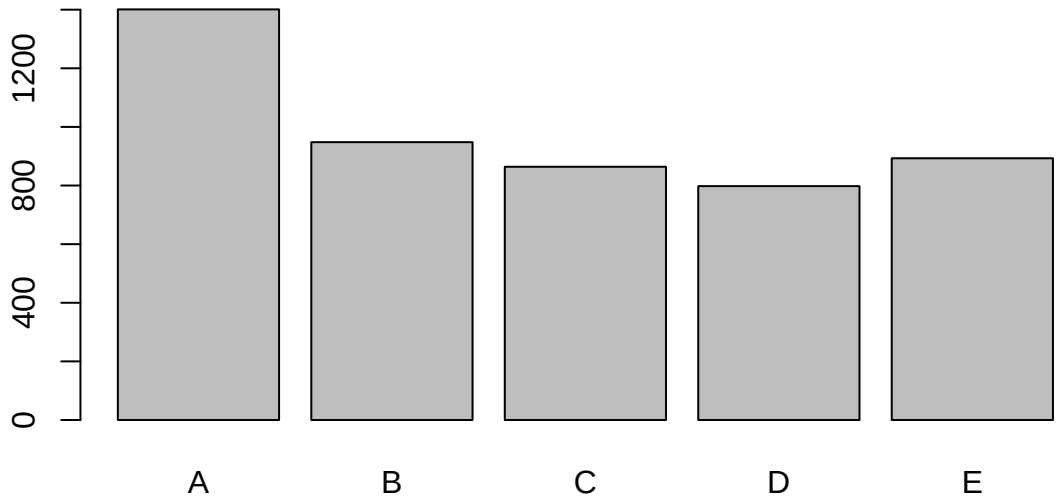

```
# print Confusion Matrix
confusionMatrix(predTest,testingSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1257  401  383  355  136
##          B   18  300   28  148  125
##          C  115  248  444  301  246
##          D    0    0    0    0    0
##          E    5    0    0    0  394
##
## Overall Statistics
##
##                Accuracy : 0.4884
##                  95% CI : (0.4743, 0.5025)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3318
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9011  0.31612  0.51930   0.0000  0.43729
## Specificity            0.6366  0.91934  0.77525   1.0000  0.99875
## Pos Pred Value         0.4964  0.48465  0.32792      NaN  0.98747
```

```
## Neg Pred Value         0.9418  0.84854  0.88423   0.8361  0.88746
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2563  0.06117  0.09054   0.0000  0.08034
## Detection Prevalence   0.5163  0.12622  0.27610   0.0000  0.08136
## Balanced Accuracy      0.7689  0.61773  0.64728   0.5000  0.71802
```

Test the random forest model

```
# make predictions
predTest <- predict(modelRF,testingSet)

# summarize results
plot(predTest)
```




```
# print Confusion Matrix
confusionMatrix(predTest,testingSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    6    0    0    0
##          B    0  942    6    0    0
##          C    0    1  849   14    0
##          D    0    0    0  790    8
##          E    0    0    0    0  893
##
## Overall Statistics
```

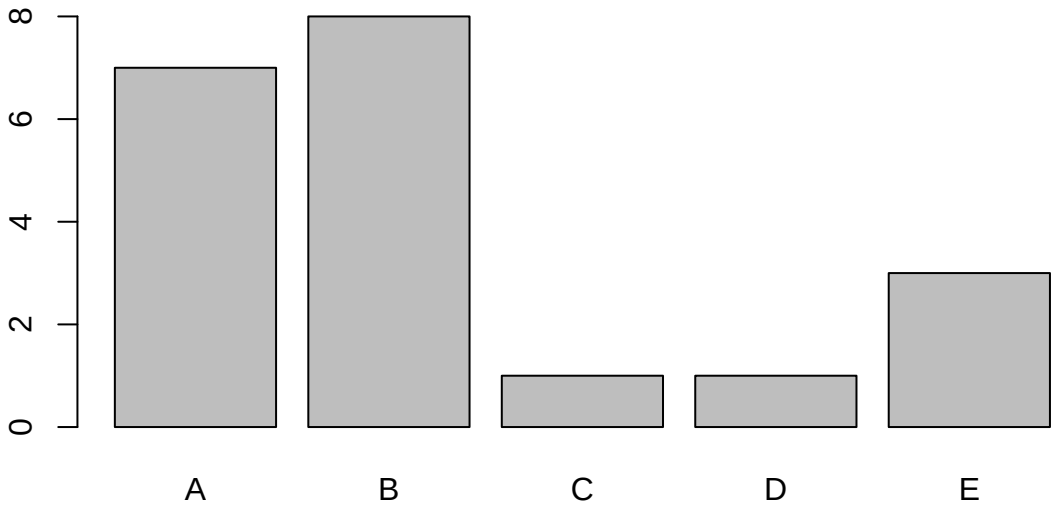
```
##
##                Accuracy : 0.9929
##                  95% CI : (0.9901, 0.995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9926   0.9930   0.9826   0.9911
## Specificity            0.9983   0.9985   0.9963   0.9980   1.0000
## Pos Pred Value         0.9957   0.9937   0.9826   0.9900   1.0000
## Neg Pred Value         1.0000   0.9982   0.9985   0.9966   0.9980
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1921   0.1731   0.1611   0.1821
## Detection Prevalence   0.2857   0.1933   0.1762   0.1627   0.1821
## Balanced Accuracy      0.9991   0.9956   0.9946   0.9903   0.9956
```

Compare models

Comparing the two confusion matrix and the model statistics, the Random Forest has a better result than the decision tree (repart). The Random Forest model has an accuracy of 99.29% against only 48.84% in decision tree model.

Apply the Random Forest model to validation set

```
# make predictions
predValidation <- predict(modelRF,testing)

# summarize results
plot(predValidation)
```




```
# print results
print(predValidation)
```

```
##  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
```

Excutive Summary

For this project, the goal was to predict the class of each exercise (A, B, C, D or E). First of all, the data variables that were mostly NAs or empty were removed. Than the variables that were not meaningfull were removed, like ID or the name of the user.

To train the model, a Decision Tree and a Random Forest were used in 75% of the training data. Applaying the models to the test data, the Random Forest had a better result with an accuracy of 99%.

Then the Random Forest model was used with validation data and the results will be evaluated in another assignment.