Final assigment - Practical Machine Learning

Domingo González

July, 2016

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

Objective

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. 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.

Data

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Loading and preparing Data for predictions.

```r
set.seed(22222)

training_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(training_url), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testing_url), na.strings=c("NA","#DIV/0!",""))
```

```
## The training data set contains 19622 observations for 160 variables and the testing data set contains

dim(training)
```

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

```
dim(testing)
```

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

```
## We can observe that we have a large sample size corresponding to the "training" data, so according to

library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.1

## Loading required package: lattice

## Loading required package: ggplot2
```

```
intrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
training_p <- training[intrain, ]
testing_p <- training[-intrain, ]
dim(training_p)
```

```
## [1] 11776   160
```

```
dim(testing_p)
```

```
## [1] 7846  160
```

```
## Once we have obtained the data for prediction, we need to clean it from variables and observations th

## Cleaning variables which their variances are close to zero
zero_variance <- nearZeroVar(training_p)
training_p <- training_p[ ,-zero_variance]
testing_p <- testing_p[ ,-zero_variance]

## Cleaning variables with most of NAs
most_NAs <- sapply(training_p, function(x) mean(is.na(x))) > 0.6
training_p <- training_p[ , most_NAs==FALSE]
testing_p <- testing_p[ , most_NAs==FALSE]

## Cleaning variables that are not significants in the prediction analysis as the first 5 variables whic
training_p <- training_p[ , -(1:5)]
testing_p <- testing_p[ , -(1:5)]

## Dimensions of our cleaned and final data to use for the prediction analysis
dim(training_p)
```

```
## [1] 11776    54
```

```r
dim(testing_p)
```

```
## [1] 7846   54
```

Prediction Models

Prediction with Decision Trees

```r
library(rpart)
model_Dtree <- rpart(classe ~ ., data=training_p, method="class")

library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.3.1
```

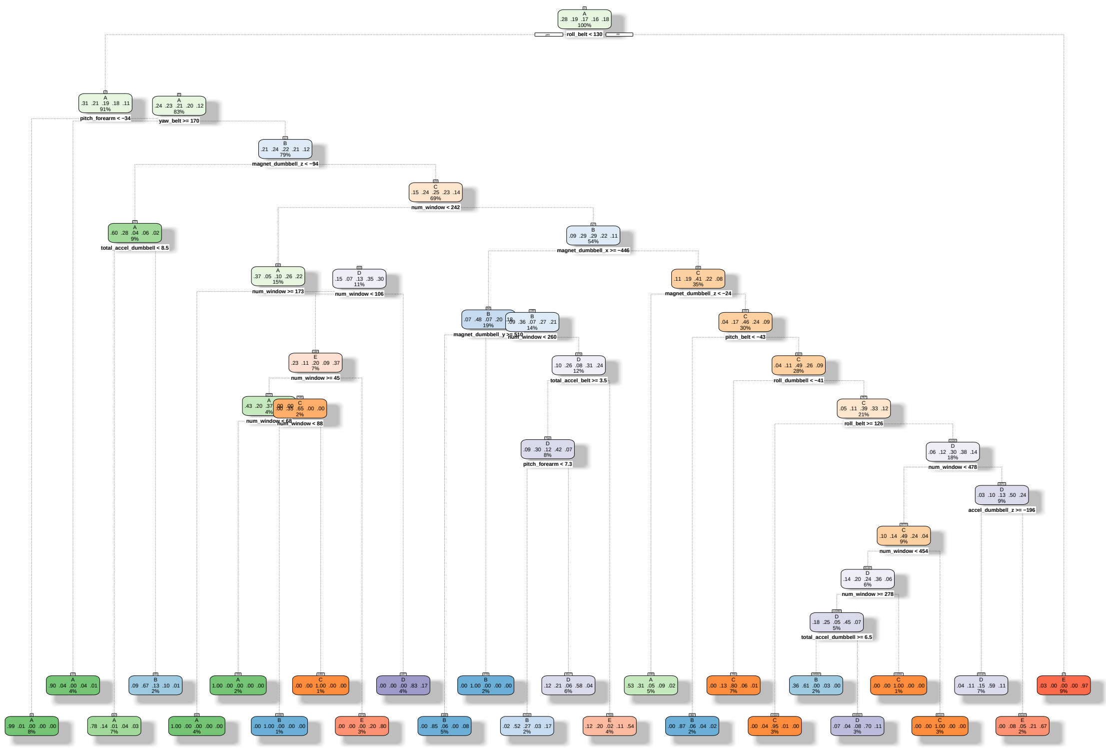
```
## Rattle: A free graphical interface for data mining with R.
## Versión 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.3.1
```

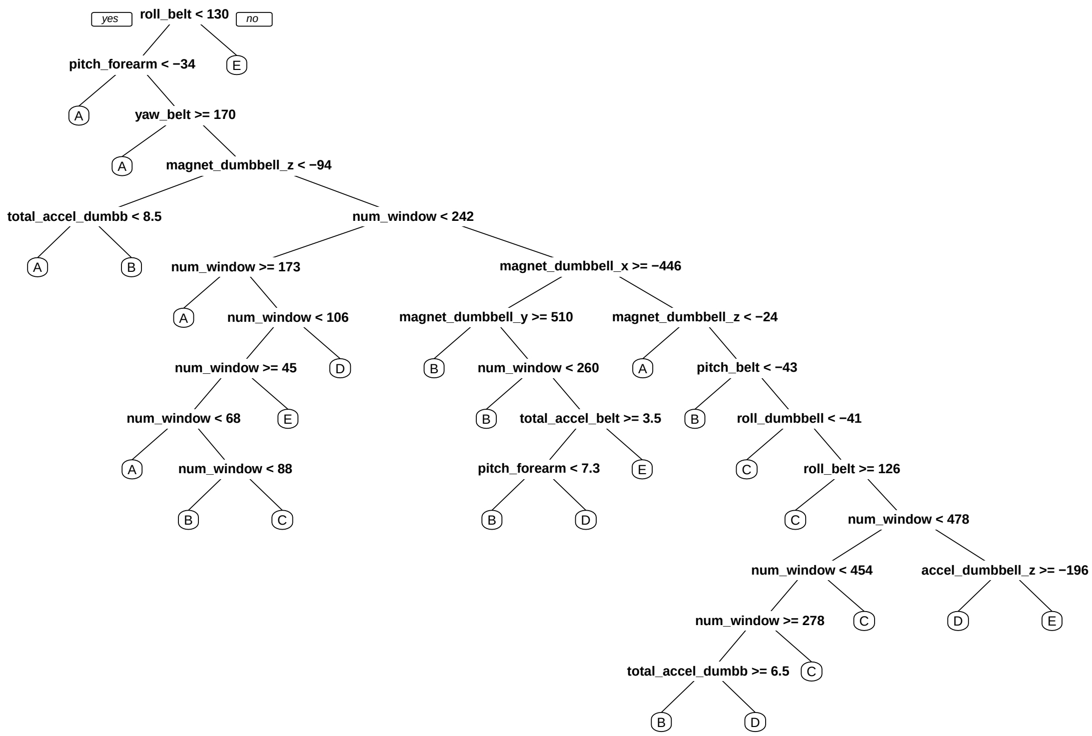
```r
library(RColorBrewer)
## to view the decision tree we can use these two options
fancyRpartPlot(model_Dtree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2016−jul.−21 08:41:39 Dogoz

```r
prp(model_Dtree)
```




```r
## Using the model (model_Dtree) for Prediction:

prediction_Dtree <- predict(model_Dtree, testing_p, type = "class")

## test results from the prediction_Dtree
library(caret)
confusionMatrix(prediction_Dtree, testing_p$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1951  220   29   85   20
##          B   92  992  148   40   66
##          C    0   86 1069   58    6
##          D  106  148  109  939  158
##          E   83   72   13  164 1192
##
## Overall Statistics
##
##                Accuracy : 0.7829
##                  95% CI : (0.7737, 0.792)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.7253
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8741   0.6535   0.7814   0.7302   0.8266
## Specificity            0.9369   0.9453   0.9768   0.9206   0.9482
## Pos Pred Value         0.8464   0.7414   0.8769   0.6432   0.7822
## Neg Pred Value         0.9493   0.9192   0.9549   0.9457   0.9605
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2487   0.1264   0.1362   0.1197   0.1519
## Detection Prevalence   0.2938   0.1705   0.1554   0.1861   0.1942
## Balanced Accuracy      0.9055   0.7994   0.8791   0.8254   0.8874
```

Prediction with random forest

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.1
```

```
## 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
model_rf <- randomForest(classe ~ ., data=training_p)
prediction_rf <- predict(model_rf, testing_p, type = "class")
confusionMatrix(prediction_rf, testing_p$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    6    0    0    0
##          B    2 1511   10    0    0
##          C    0    1 1358    6    0
##          D    0    0    0 1278    1
##          E    0    0    0    2 1441
##
## Overall Statistics
##
##                Accuracy : 0.9964
```

```
##                  95% CI : (0.9948, 0.9976)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9954   0.9927   0.9938   0.9993
## Specificity            0.9989   0.9981   0.9989   0.9998   0.9997
## Pos Pred Value         0.9973   0.9921   0.9949   0.9992   0.9986
## Neg Pred Value         0.9996   0.9989   0.9985   0.9988   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1926   0.1731   0.1629   0.1837
## Detection Prevalence   0.2850   0.1941   0.1740   0.1630   0.1839
## Balanced Accuracy      0.9990   0.9967   0.9958   0.9968   0.9995
```

Prediction with linear discriminant analysis

```
model_lda <- train(classe ~ ., data=training_p, method = "lda")
```

```
## Loading required package: MASS
```

```
prediction_lda <- predict(model_lda, testing_p)
confusionMatrix(testing_p$classe, prediction_lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1834   62  151  172   13
##          B  206  993  197   60   62
##          C  134  144  904  138   48
##          D   77   52  180  924   53
##          E   58  222  137  117  908
##
## Overall Statistics
##
##                Accuracy : 0.709
##                  95% CI : (0.6988, 0.7191)
##     No Information Rate : 0.2943
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6318
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7943   0.6741   0.5762   0.6549   0.8376
## Specificity            0.9281   0.9176   0.9261   0.9437   0.9210
## Pos Pred Value         0.8217   0.6542   0.6608   0.7185   0.6297
## Neg Pred Value         0.9154   0.9241   0.8973   0.9258   0.9725
## Prevalence             0.2943   0.1877   0.2000   0.1798   0.1382
## Detection Rate         0.2337   0.1266   0.1152   0.1178   0.1157
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8612   0.7959   0.7511   0.7993   0.8793
```

As it can be observed the Random Forests prediction is more accurate than the linear discriminat analysis and the decision tree prediction.

Model applied to submission set

We use the random forest prediction with the high accuracy model on the test data set.

```
prediction_validation <- predict(model_rf, newdata=testing)
prediction_results <- data.frame(problem_id=testing$problem_id, predicted=prediction_validation)
prediction_results
```

```
##    problem_id predicted
## 1           1         B
## 2           2         A
## 3           3         B
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
## 9           9         A
## 10         10         A
## 11         11         B
## 12         12         C
## 13         13         B
## 14         14         A
## 15         15         E
## 16         16         E
## 17         17         A
## 18         18         B
## 19         19         B
## 20         20         B
```

Conclusion

Three prediction models were applied, the most accurated model was the random forestFor(0.9949), then the decision tree model (0.7346) and the linear discriminant analysis (0.7146). The random forest prediction model was applied to the test data for validation.