HAR Final Assessment

Practical Machine Learning Project

Mathias Barat

1 - SYNOPSIS

Context

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.

Problem

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.

Goal of the project

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).

Dataset information

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz6ZuaJcLo0

2 - DATASETS

Downloading file:

```
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if (!file.exists("data/training.csv") | !file.exists("data/testing.csv")){
```

```
  download.file(train_url, destfile = "data/training.csv")
  download.file(test_url, destfile = "data/testing.csv")
}

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

Exploratory Data Analysis

```
summary(training$classe)
```

```
##    Length     Class      Mode
##     19622 character character
```

```
unique(training$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

```
training$classe <- factor(training$classe)
```

Cleaning

Suppress all NA in the dataset

```
training <- training[,colSums(is.na(training)) == 0]
testing <- testing[,colSums(is.na(training)) == 0]
```

Suppress useless columns

```
# X, user_name, timestamps, windows
training <- training[,-c(1:7)]
testing <- testing[,-c(1:7)]
```

testing, training and quizz datasets

To avoid errors I rename the datasets and create my own training, testing dataset

```
library(caret)
```

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

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

```
test_quizz <- testing
inTrain <- createDataPartition(training$classe, p = 0.8, list = FALSE)
temp <- training
training <- temp[inTrain,]
testing <- temp[-inTrain,]
rm(temp)
```

final dataset

```
# Training dataset:
dim(training)
```

```
## [1] 15699    53
```

```
# Testing dataset:
dim(testing)
```

```
## [1] 3923   53
```

```
# Testing dataset
dim(test_quizz)
```

```
## [1]  20 153
```

Histogram of Y

```
library(ggplot2)
ggplot(training, aes(x=classe)) + geom_histogram(stat = "count", fill="red")+ labs(title="Classe count
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

3 - Recursive Partition modeling

```
library(ggplot2); library(caret); library(randomForest); library(rpart)
```

```
## 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
```

```
modfit_rpart <- rpart(classe ~ ., data=training, method="class")

predict_rpart <- predict(modfit_rpart, testing, type = "class")
```

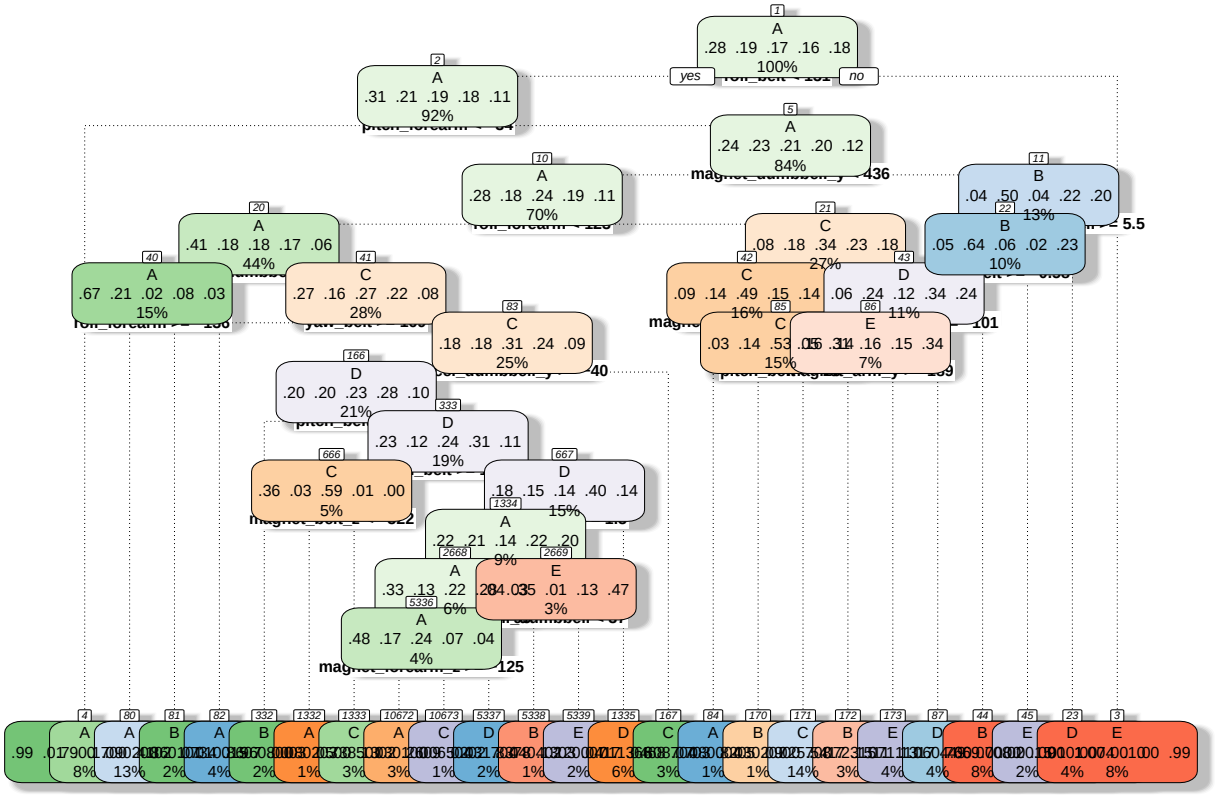
```
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Entrez 'rattle()' pour secouer, faire vibrer, et faire défiler vos données.

##
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':
##
##     importance
```

```
fancyRpartPlot(modfit_rpart, cex= 0.5)
```

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




```
confusionMatrix(predict_rpart, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 971 113  25  35  14
##          B  46 482  62  67  66
##          C  31  78 540 100  94
##          D  48  52  40 396  36
##          E  20  34  17  45 511
##
## Overall Statistics
##
##                Accuracy : 0.7392
##                  95% CI : (0.7252, 0.7529)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6698
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8701   0.6350   0.7895   0.6159   0.7087
## Specificity            0.9334   0.9238   0.9065   0.9463   0.9638
## Pos Pred Value         0.8385   0.6667   0.6406   0.6923   0.8150
## Neg Pred Value         0.9476   0.9134   0.9532   0.9263   0.9363
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2475   0.1229   0.1376   0.1009   0.1303
## Detection Prevalence   0.2952   0.1843   0.2149   0.1458   0.1598
## Balanced Accuracy      0.9017   0.7794   0.8480   0.7811   0.8363
```

---

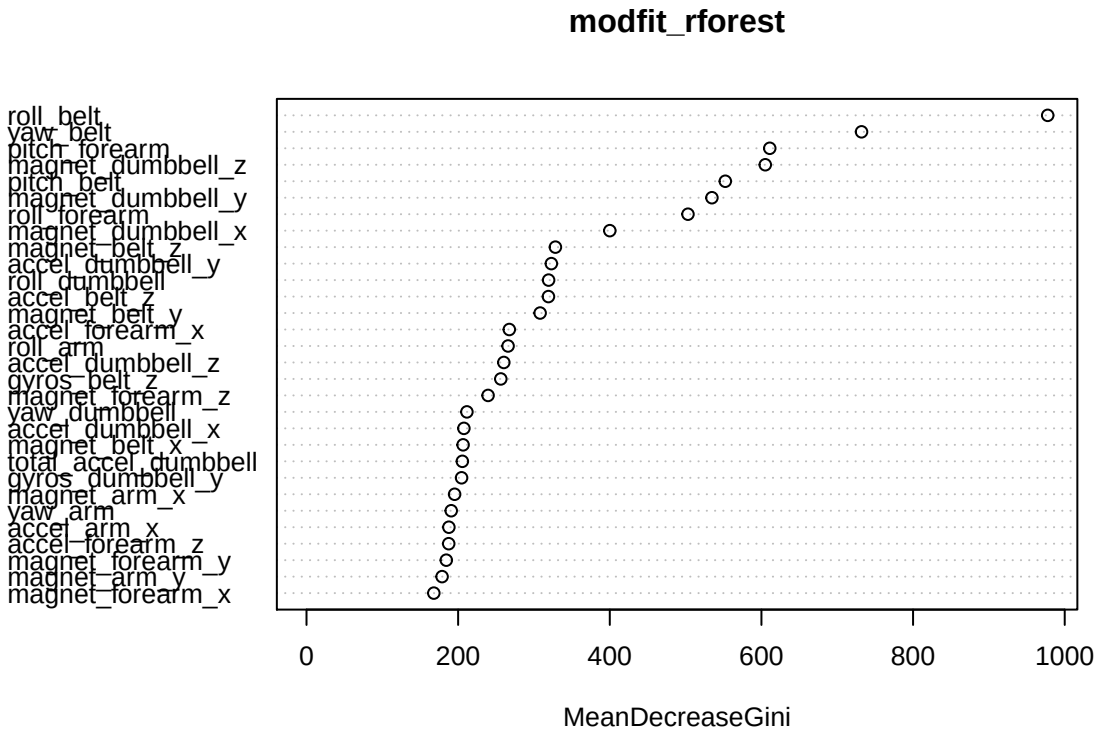
4 - Random Forest Modeling

```r
library(randomForest)
modfit_rforest <- randomForest(classe ~ .,data=training, method = "class", keep.inbag = TRUE)
predict_rforest <- predict(modfit_rforest, testing, type = "class")

modfit_rforest
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, method = "class",      keep.inbag = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.43%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4461    3    0    0    0 0.000672043
## B   13 3020    5    0    0 0.005924951
```

```
## C    0   15 2721    2    0 0.006208912
## D    0    0   21 2551    1 0.008550330
## E    0    0    2    5 2879 0.002425502
```

```
varImpPlot(modfit_rforest, cex = 0.8 )
```




```
confusionMatrix(predict_rforest, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    2    0    0    0
##          B    0  757    2    0    0
##          C    0    0  681    7    0
##          D    0    0    1  635    2
##          E    0    0    0    1  719
##
## Overall Statistics
##
##                Accuracy : 0.9962
##                  95% CI : (0.9937, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.9952
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9956   0.9876   0.9972
## Specificity            0.9993   0.9994   0.9978   0.9991   0.9997
## Pos Pred Value         0.9982   0.9974   0.9898   0.9953   0.9986
## Neg Pred Value         1.0000   0.9994   0.9991   0.9976   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1930   0.1736   0.1619   0.1833
## Detection Prevalence   0.2850   0.1935   0.1754   0.1626   0.1835
## Balanced Accuracy      0.9996   0.9984   0.9967   0.9933   0.9985
```

5 - Conclusion

Model choice

The Random forest gets an accuracy of 0.9949 versus a 0.7353. Without any doubt the Random forest is seriously the best for our case.

Quizz prediction with the test set

```
predict_quizz <- predict(modfit_rforest, test_quizz, type = "class")
predict_quizz
```

```
##  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
```