# MachineLearningProject

*Gabi Mingorance*

*25/1/2020*

## Summary

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).

1. Data is loaded and cleaned (removing NA cases)
2. Three different models are then explored and compared.
3. Finally, the best of them is used to predict.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.2
```

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

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

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

```
## Warning: package 'rpart.plot' was built under R version 3.6.2
```

```
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.2
```

```
## Rattle: A free graphical interface for data science with R.
## Versión 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.2
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

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

```
library(knitr)
```

### Download data training and testing data and tidy up.

Only if the files do not yet exist, they are downloaded.

```
set.seed(12345)


if(!file.exists("pml-training.csv")){
        fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        download.file(fileUrl,destfile="./pml-training.csv")
}
if(!file.exists("pml-testing.csv")){
        fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
        download.file(fileUrl,destfile="./pml-testing.csv")
}

training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
```

Clean up data

```
columnsToBeRemoved <- which(colSums(is.na(training) |training=="")>0.9*dim(training)[1])
training <- training[,-columnsToBeRemoved]
training <- training[,-c(1:7)]

# We do the same for the test set
columnsToBeRemoved <- which(colSums(is.na(testing) |testing=="")>0.9*dim(testing)[1])
testing <- testing[,-columnsToBeRemoved]
testing <- testing[,-1]
```

Split Training data in training and testing to test our models before consuming the actual testing data.

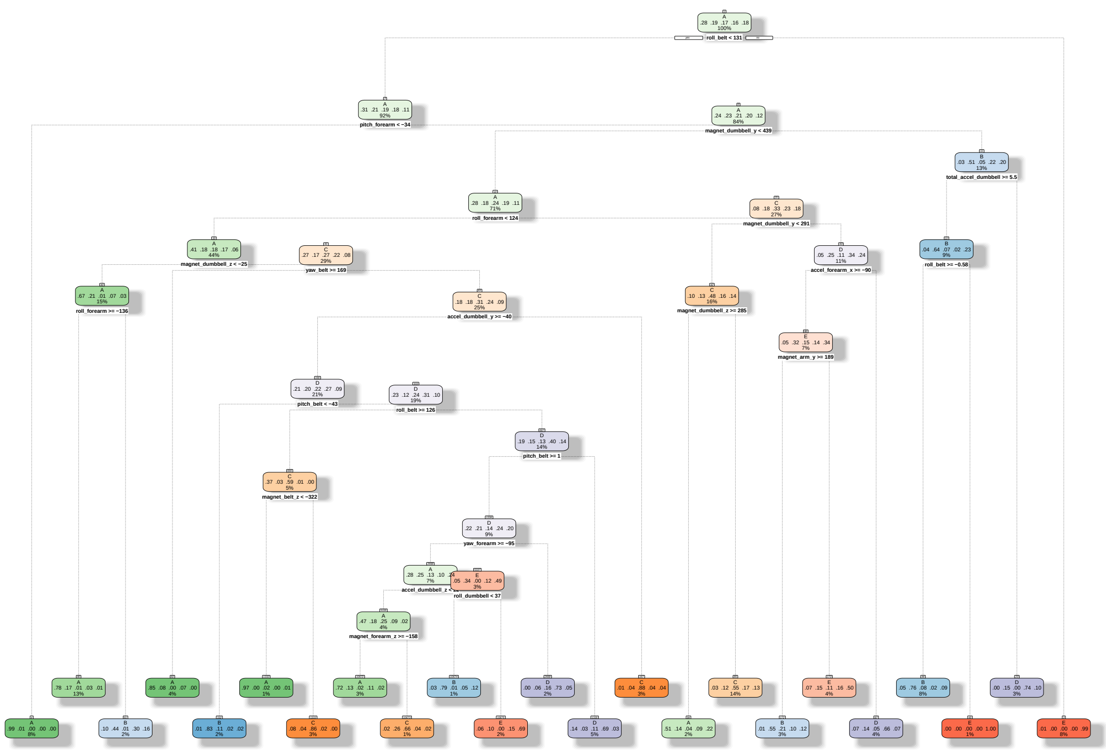
```r
partition <- createDataPartition(training$classe, p=0.6, list=FALSE)
subTraining <- training[partition, ]
subTesting <- training[-partition, ]
```

## Model Comparison

Clasification Tree

```r
model_classificationtree <- rpart(classe ~ ., data=subTraining, method="class")
fancyRpartPlot(model_classificationtree)
```

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



Rattle 2020−ene.−26 15:07:05 Mercedes

```r
subprediction <- predict(model_classificationtree,newdata=subTesting, type = "class")

confusion <- confusionMatrix(subTesting$classe,subprediction)

confusion$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1995   75   44   74   44
##          B  246  890  198  112   72
```

```
##           C   49  111 1094   79   35
##           D   83  119  153  840   91
##           E   51  115  143   94 1039
```

```
confusion$overall[1]
```

```
##  Accuracy
## 0.7466225
```

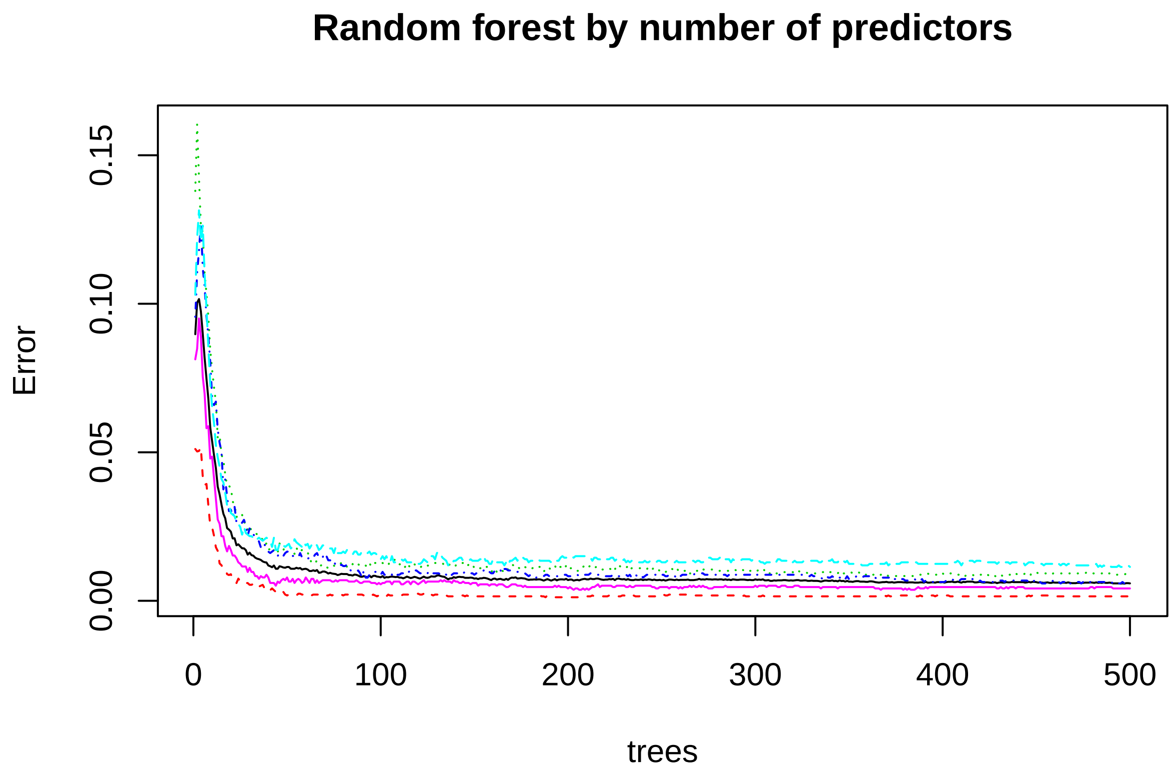
Random Forest

```
model_RandomForest <- randomForest(classe ~ ., data=subTraining)

print(model_RandomForest)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = subTraining)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.59%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3343    4    0    0    1 0.001493429
## B   15 2259    5    0    0 0.008775779
## C    0   11 2041    2    0 0.006329114
## D    0    0   20 1908    2 0.011398964
## E    0    0    1    8 2156 0.004157044
```

```
plot(model_RandomForest,main="Random forest by number of predictors")
```




```
subprediction <- predict(model_RandomForest,newdata=subTesting)

confusion <- confusionMatrix(subTesting$classe,subprediction)

confusion$table
```

```
##             Reference
## Prediction    A    B    C    D    E
##           A 2229    3    0    0    0
##           B    3 1515    0    0    0
##           C    0    6 1357    5    0
##           D    0    0    6 1280    0
##           E    0    0    2    5 1435
```

```
confusion$overall[1]
```

```
##  Accuracy
## 0.9961764
```

Gradient Boost

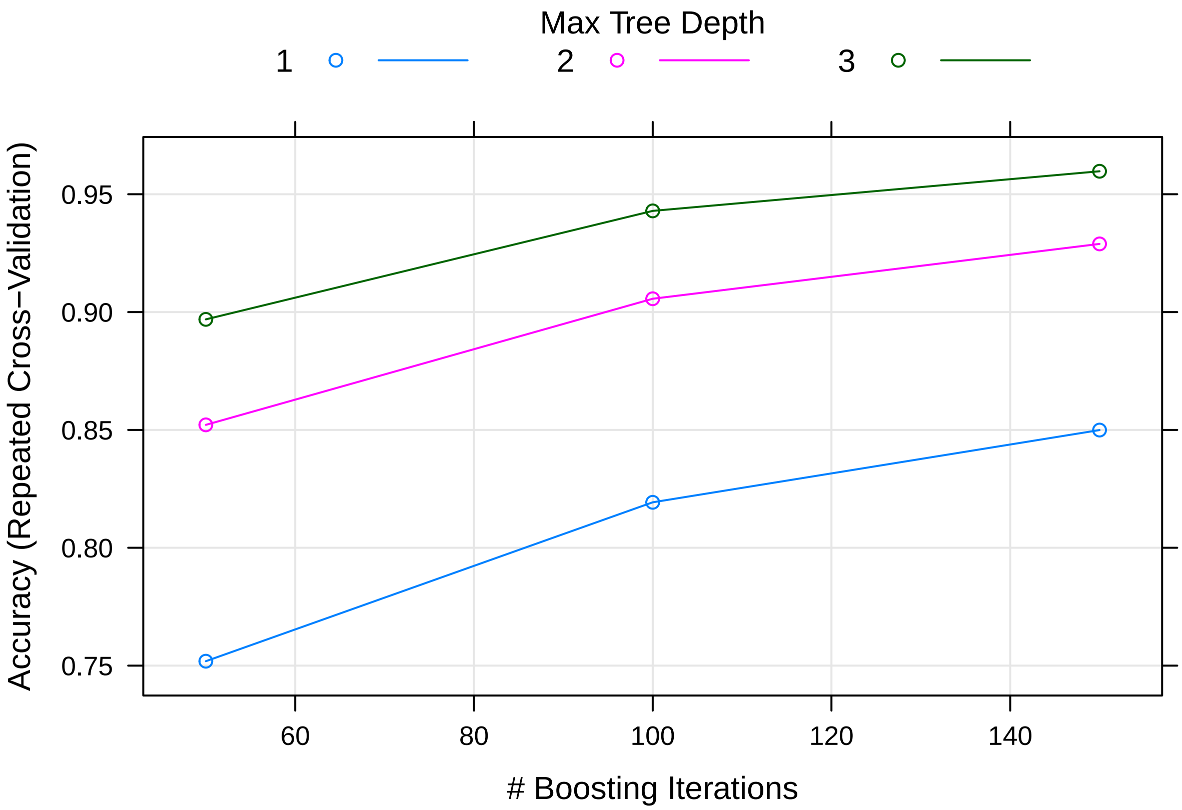
```
control <- trainControl(method = "repeatedcv",
                        number = 5,
                        repeats = 1)
```

```r
model_GradientBoost <- train(classe ~ ., data=subTraining, method = "gbm",trControl = control,verbose =

print(model_GradientBoost)
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 9421, 9422, 9421, 9420, 9420
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7518698  0.6853763
##   1                  100      0.8192942  0.7712639
##   1                  150      0.8499500  0.8101527
##   2                   50      0.8521564  0.8126400
##   2                  100      0.9056557  0.8806321
##   2                  150      0.9289244  0.9100684
##   3                   50      0.8969097  0.8694893
##   3                  100      0.9429358  0.9277971
##   3                  150      0.9597497  0.9490865
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
plot(model_GradientBoost)
```




```r
subprediction <- predict(model_GradientBoost,newdata=subTesting)

confusion <- confusionMatrix(subTesting$classe,subprediction)
confusion$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2187   33    6    5    1
##          B   49 1426   34    3    6
##          C    0   46 1304   15    3
##          D    0    7   32 1243    4
##          E    2   17   13   28 1382
```

```r
confusion$overall[1]
```

```
##  Accuracy
## 0.9612541
```

## Prediction

It seems that *Random Forest* performs better so we finally apply this model to the testing set to predict the 20 test cases.

```r
prediction <- predict(model_RandomForest,newdata=testing)
prediction
```

```
##  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
```