# Learning Machine - Prediction Assigment

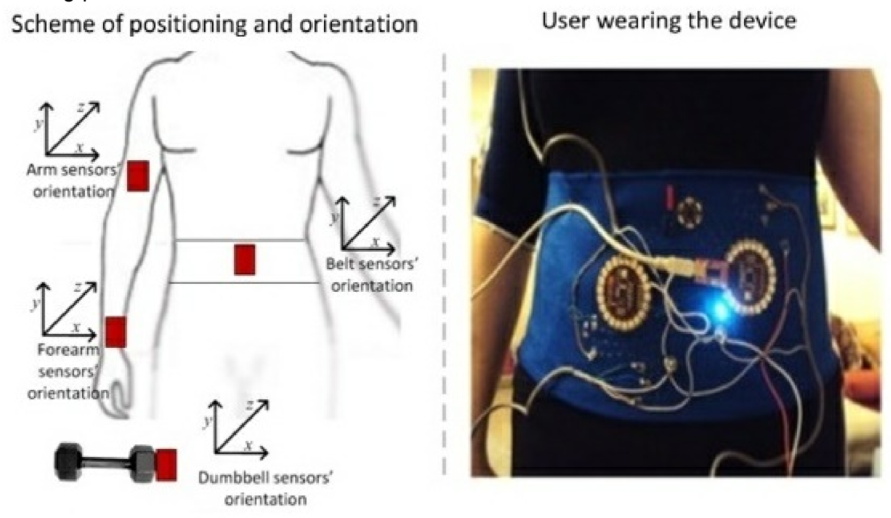
*R.E.Lobel, March 2016*

```r
setwd("K:/Pastas em Comum/Coursera/Machine Learning/assignment")
library(caret); library(rpart); library(rattle); library(randomForest); library(dplyr)
```

## Introduction

The assignment is presented with little information about the problem, in the same way problems are presented in real life. My first step was trying to better understand the problem, by researching the project “Weight Lifting Exercise”, developed at the Software Engineering Laboratory (LES) of PUC Rio de Janeiro at http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) and reading the following paper about this research: “Qualitative Activity Recognition of Weight Lifting Exercises”, which can be downloaded at “http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf (http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf)”

PUC-Rio research collected data from sensors fixed on the body of 6 participants, measuring the quality of the execution of Weight Lifting Exercises (WLE). Data was collected from sensors fixed on the belt, forearm, arm, and dumbbell, positioned on the participants as shown in the following picture.


Scheme of positioning and orientation
User wearing the device
Arm sensors' orientation
Belt sensors' orientation
Forearm sensors' orientation
Dumbbell sensors' orientation

The goal of this assignment is to use the sensors data to build a model able to predict how well the WLE was done and verify its accuracy by testing the built model against 20 provided cases. Two datasets were provided. The first one (pml-training.csv) contains the data from the sensors (predictors) and an output variable (classe), with five factors (A to E), each level defining how well the exercise was done. The second file (pml-testing.csv) has twenty cases, to be used with the model generated from the train dataset to check the accuracy of the chosen model.

## Loading and cleaning the Datasets

After loading and inspecting these datasets, I verified many non-necessary columns (with NAs and “”) which I removed from both datasets. I also removed from both datasets the initial columns (X, user_name, time_stamp and window number) because they are not predictors to the Classe outcome.

```r
training = read.csv("pml-training.csv",na.strings = c("NA",""))
training = training[,colSums(is.na(training))==0]
training = training[,-c(1:7)]
col_train = colnames(training)
dim(training)
```

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

```r
testing = read.csv("pml-testing.csv",na.strings = c("NA",""))
testing = testing[,colSums(is.na(testing))==0]
testing = testing[,-c(1:7)]
col_test = colnames(testing)
dim(testing)
```

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

```r
for (i in 1:dim(training)[2]) if (col_train[i] != col_test[i]) print(col_train[i])
```

```
## [1] "classe"
```

After this initial preparation of both datasets, remained 53 variables, including the 52 predictors and the “classe” outcome in the training dataset and a “problem_id” variable (1:20) on the testing dataset

## Choosing a method

Analyzing the training dataset, we face a problem with 52 labeled predictors and a non-numeric 5 factor outcome variable (“classe”) which can take the values “A,B,C,D and E”. My approach to tackle this kind of problem is by using a supervised classification machine learning algorithm.

I decided to start using “Decision trees” which is a popular method for various machine learning classification tasks, which is simple, fast and easy to interpret its results. If the resulting model shows not to be accurate, my second method to be tried will be the “Random Forest” method, which is slower but tends to provide more accurate results.

From Wikipedia, comparing the two methods we have that “trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, because they have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model.”

## Splitting the Data

When fitting a model, we capture the signal and noise. The objective is to capture as much as possible the signal and avoid the noise. It is always possible to design a perfect in-sample predictor, but this approach will include both signal and noise, overfitting the model. Overfitted models won´t perform as well on new samples.

This why it is so important to do a cross validation of the model using out of the sample data. Only predicting the outcome with this out of sample data will make possible to check the expected out of sample error and have a better visibility of the model accuracy.

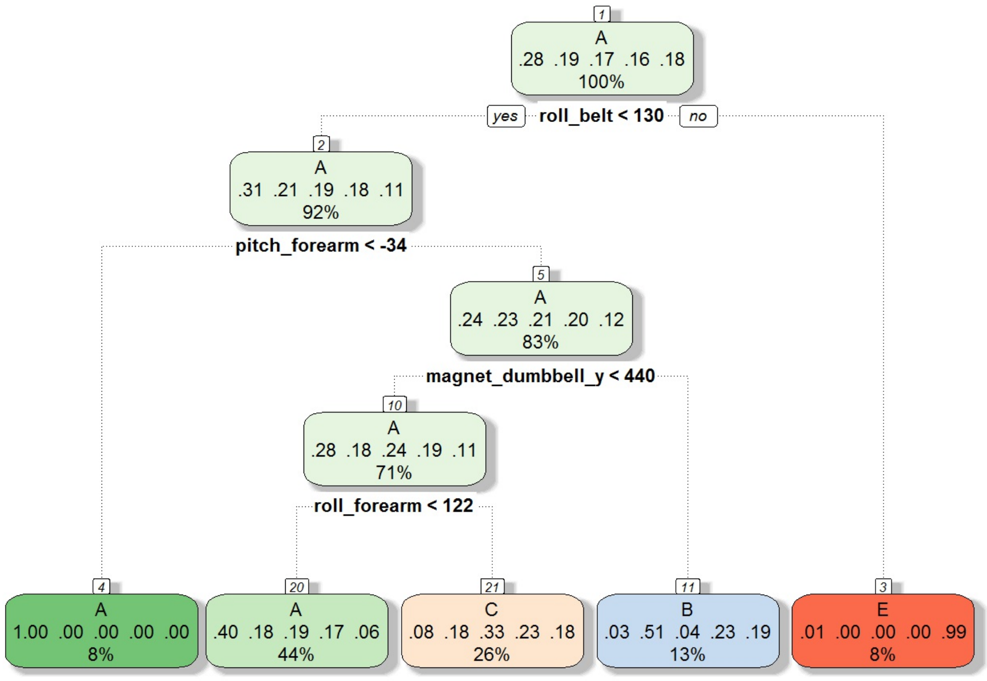
To be able to do the cross validation of the resulting models, get the expected out of sample error and an estimation of the model accuracy, I decided to split the train dataset in two data-sets. 70% of the data allocated for model building and 30% for the cross validation.

```
subtrain = createDataPartition(training$classe, p=0.7, list=F)
train_main = training[subtrain,]
train_chk = training[-subtrain,]
```

# Building the Models

## Building a Decision Tree model with RPART method

```
# Build a RPART model, by training the predictors in train_main
set.seed(1103)
model_rpart = train(classe~.,data=train_main, method="rpart")
fancyRpartPlot(model_rpart$finalModel)
```



Rattle 2016-Mar-01 19:21:15 gene

```
# The count of Classe == D Outcome factor in the training dataset
classes = train_main %>% group_by(classe) %>% summarise(count=n())
classes[4,2]/ nrow(train_main)
```

```
##       count
## 1 0.1639368
```

```
# As the generated model has no branch resulting in "classe=D",
# while 16% of the data is indeed of classe D, we can infer this model has very low accuracy.
```

## Cross validation & out of sample error for the RPART method

```
# Perform cross validation by predicting "classe" with out of sample data
pred_rpart = predict(model_rpart,train_chk)
# Measuring the out of sample error
confusionMatrix(pred_rpart,train_chk$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.4915888
```

```
# 50% accuracy shows that the RPART does not give an accurate model for this problem
```

## Building a Random Forest model

```
set.seed(1103) #seed used for reproducibility
model_rf = randomForest(classe~.,data=train_main,ntree=100, importance=TRUE)
```

## Cross validation & out of sample error for the Random Forest method

```
# Perform cross validation by predicting "classe" with out of sample data
pred_rf =predict(model_rf,train_chk)
# Measuring the out of sample error
confusionMatrix(pred_rf,train_chk$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.9943925
```

```
# Above 99% accuracy shows that Random Forest gives an accurate model for this problem
```

## Predicting the “classe” outcome of the Testing Dataset

```
# With an accuracy over 99% on the prediction over the out of sample dataset ("train_chk")
# I am confident to apply the "model_rf" Random Forest generated model on the testing data set
pred_test = predict(model_rf,testing)
# The following results were obtained for the 20 test cases of the testing data set
pred_test
```

```
##  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
```

## Conclusion

The problem was configured as a supervised classification machine learning problem. Two classification methods: “RPART” and “Random Forest” were used to generate the predicting models.

RPART generated a poor model, with an accuracy of 50% on the out of sample data. Random Forest generated a very good predictive model, with an accuracy over 99% on the out of sample data.

Therefore we used, with success, the model generated from the Random Forest method to predict the 20 test cases of the testing data set.