Challenge B - Answer sheet

Julia Guth, Thi Diem my

8 décembre 2017

## TASK 1B

# Step 1

We decide to choose the ML technique : random forest.

It is a machine learning algorithm, for classification and regression. It creates a forest with number of trees and a high number of trees gives us a high accuracy in results. So the purpose of this technique is to indentify links between a variable to be explained and a explanatory variable.

We can extract trees from the forest to regress our model (here we want to predict house prices in Ames, so we want to regress the price according to many different variables).

In this exercise, we will do a regression and adding, substract or combine trees, to see if a class is significant or not and to predict house prices in Ames.

# Step 2

We are going to train the technique random forest on the training data.We import dataset training to use it.

Then, we will transform all the characters variables into factors and remove missing values from the data.

# Step 3

We import dataset test and select only important parameters and subset them. Then we will substract all the missing values.

We are going to make predictions on the test data with our function “fit”, and compare them to the predictions of a linear regression with our function“model”.

With the ML technique of random forests we have categorical values, for example 4 houses have a price of 165 000 dollars, whereas with the linear regression we have continuous values with a price mean of houses in Ames, Iowa of 184 100

## Task 2B

# Step 1

We are estimating a low-flexibility local linear model on the training data using the npreg function and a bandwidth of 0.5. We call this model “ll.fit.lowflex”. We found a R-squared of 0.854 and a Residual standard error of 1.0854.

# Step 2

We are estimating a high-flexibility local linear model on the training data using the npreg function and a bandwidth of 0.01. We call this model “ll.fit.highflex”. We found a R-squared of 0.968 and a residual standard error of 0.507. We can see that with a lower bandwidth the R-squared is higher and the residual standard error is lower. This new model is better than the low-flexibility local linear model, more precise.

# Step 3

We plot the scatterplot of x-y, along the predictions of the two local linear models we estimated in the previous questions. The blue line corresponds to the predictions of ll.fit.highflex and the red one to the predictions of ll.fit.lowflex. The black line and points corresponds to the values of the training data.

# Step 4

We can compare the blue and the red lines to see which predictions are more variable and have the least bias. The predictions of the high flexibility local linear model (blue line) are the more variable ones and have the least bias. The blue line joins all the black points, so that the predictions are corresponding the the values of the training data, whereas, the red line doesn’t match all the black point.

# Step 5

We are estimating the y.ll.highflex model and the y.ll.lowflex model on the previous models ll.fit.highflex and ll.fit.lowflex on the test data instead of the training data using the function mutate.

We plot then the scatterplot of x-y, along the predictions of the two new local linear models we estimated. The blue line corresponds to the predictions of y.ll.highflex and the red one to the predictions of y.ll.lowflex. The black line and points corresponds to the values of the test data.

# Step 6

We create a vector of bandwidth going from 0.01 to 0.5 with a step of 0.001 using the function seq. We call it “bdw”.

# Step 7

We are estimating a local linear model y ~ x on the training data with each bandwidth using the function lapply and our new function vector “bdw”.

# Step 8

To compute for each bandwidth the MSE on the training data we create a new function “mse.training” which calculate the MSE of the model fit.model. For this we use the equation of the MSE : mean((y - predictions)^2). Then we list all the MSE of the ll.fit.flex model in a table.

# Step 9

We do the same as in step 8 on the test data instead of the training data.

# Step 10

We draw on the same plot how the MSE on training data, and test data, change, when the bandwidth increases using the function tbl\_df. The blue line corresponds to the MSE on training data and the orange one corresponds to the MSE on test data.

With a bandwith of near 0, the MSE on training data is equal to 0 whereas the MSE on test data is equal to 2.5. Then, the blue line sharply increases and the orange one sharply decreases. The intersection of both lines happens with a bandwidth of 1.4. After that, they are increasing until reaching a MSE of 2.2 for the blue line and 1.4 for the orange line with a bandwidth of 0.5.

As a conclusion, small bandwidth implies low means squared with the data set training because it has many observations. The predictions are then more realistic and close to the real model. Whereas small bandwidth implies high means squared with the data set test because it has less observations. The predictions are then less realistic and far from the real model.

## Task 3B

# Step 1

We import the CNIL and SIREN dataset. But when we imported the CNIL dataset from csv, we could see that there is only one column for all the informations. So that, it is better if we open it as an excel file and let’s rename the data “CNIL1”.

# Step 2

Before creating a nice table we have to create a first subset containing only the numbers with more than 4 figures and a second subset ontaining only the numbers with less than 6 figures from CNIl1\_SUB1 so that the Code Postal of CNIL1\_SUB2 contains the right number of figures. We saw in the table that sometimes the Code postal numbers don’t have 5 figures, so we need to take only the “Code Postal” that contains 5 figures.

Now we can create a nice table using the function colnames with the number of organizations that has nominated a CNIL per department.

# Step 3

To merge the CNIL dataframe and the SIREN data into the CNIL data we could have used the command merge but it would have taken us too much time