Challenge B

Amund Hanson Kord / Mauricio Hitschfeld Arriagada

Link to the Github repo: https://github.com/mhitschfeld/Challenge_B_Final.git

Task 1B - Predicting house prices in Ames, Iowa (continued)

First, we load the dataset and call it train and test.

We analyse the data and get that there are some missing variables.

We remove the columns where there are a lot of missing variables, and we remove the rows where there are fewer misiing variables.

Step 1: We choose to use the random forest technique. Random forest technique are an ensemble machine learning method. As opposed to single learning algorythms, ensamble methods obtain multiple learning algorithms, or in this case, "decision trees"" wich in many cases gives the model better predictive capabilities. Decision tree models uses a a sequence of algotythms to go from the observations to the given predictions. However, "deep" trees (trees with a lot of decision points) have a tendency to overfit their training sets. This results in a high variance even though the bias of the model is low. Random forest operate by constructing multiple deep decision trees from differen parts of the training set and then taking the average of these decision trees. Thus reducing the variance of the model, but at the expence of some bias. At each algorythm one can also adjust the number of variables randomly sampled as candidates at each split, or that the model asseses at each split what is the best variable to use. In smaller trees one could experience that the model does not use all the variables in the data.

Step 2: We start off by setting using our model from Challenge A, with 10 trees and 9 random variables that is sampled at each split.

We can see that around 74% of the variance in housing prices is explained. We then do the regression again, but this time we set the number of trees to 100, this time we do not specify the number of random variables assesed at each split. This leads the model to choose randomly among all the variables in the sample at each split.

Now we see that the variance jumped up to around 84%. We can also plot the residuals of the model at diffrent number of trees.

We see that roughly, the more decision trees we use in the model, the smaller the residuals become. That being said, even though the random forest model takes the average of several decision trees.

Step 3: We have chosen to use the OLS estimator to compare with the random forest estimator. We therefore first regress the OLS model, to make it more easily comparable we use tha same variables as we used in the random forest regression.

We then compare the of the two models by finding their respective predictive values on the data set test.

We then calculate the difference between the predictions at each point and show the results in a summary.

The variance between the two models are considerable at their max and min level. Even though we see that the first and third quantile only has an approximate error of 10.000 in respectively positive and negative direction. The mean deviation is also considerably low given the facet that we are estimating values that vary a lot. We then plot the diffrence between the two estimators on a table.

Task 2B - Overfitting in Machine Learning (continued)

For this Challenge, we use the same variables and datasets created in Challenge A (see R Script).

Step1 -Step 2: We estimate a low-flexibility and a high-flexibility local linear model on the training data using the function *npreg*.

Step 3: We compute the predictions for $ll.fit.highflex$ and $ll.fit.lowflex$ ($\hat{y}^T$ was computed in Challenge A). Then, using *ggplot* we plot the scatterplot, along with the predictions.

Step 4: Looking at the graph (R script) we check that the model $ll.fit.highflex$ is the prediction most variable, which has the lowest bandwidth. Also, we can compute the variance of each prediction and check this statement.

In addition, we can check which prediction has the least bias computing the difference between y and the $fitted\ value$ of each model and computing the absolute value of the mean.

Then, the model $ll.fit.highflex$ has the least bias.

Step 5: We reply the Steps 1, 2, 3 and 4 using the test data.

The prediction most variable and with the least bias still is the model $ll.fit.highflex$, but now its bias increased with respect to the bias using the $training$ data.

Step 6 - Step 7 - Step 8 We create a vector using the function *seq*.

We estimate a local linear model using the functions *lapply* and *npreg*.

We create the function *train.mse* which contains the MSE. The MSE was computed using the estimation/prediction of our local linear model and y. At the end we only organize our result in a list, using *unlist* and *lapply* (see R Script).

Step 9: We reply the same done in the Step 8, but using the test data.

Step 10: We arrange in a table the *bandwidth* and both MSE, and finally we plot.

Task 3B - Privacy regulation compliance in France

Step 1: We import the dataset using the url from the CNIL website.

Step 2: We have chosen the approach of splitting up the postal code so that we get a column in the data set with only the two first digists of the code.

We then first identify the number of unique combinations of companies and first two digits of the postal code. This excludes the cases where the company has two representatives in the same department. Then in the list of unique CNIL per department list we identify the number of duplicates in the list of company names. This gives us the number of companies that has a representative in at least one department. We then use the function table() to show how many cases we have where the same company has nominated a CNIL for several departments.

The first six companies that have one representative per department is shown here, the full list is in the data frame uniq.cil. We then need to identify how many of these companies that have designated an CIL-responsible for each department. We do this by using the unique function.

This means that there are 17667 companies with a unique CIL-responsible per department, and 238 companies that have designated a CIL representative for two or more departments.

Step 3: First we import the data set using the function $fread()$. The import time is reduced by adding several arguments inside the $fread()$ command, we therefore set the following arguments inside the table when importing it **(about 10 minutes, in R script is the code that details the time needed)**.

Then we transform CIL to a data table so that the format of the data is similiar. We then merge the list of CIL representatives and the SIREN data set by the variable “SIREN” since this variable is the same in the two data sets.

The new data table “total” now only contains the companies that have a CIL representative in the data set SIREN. If one wishes to include the companies that do not have a CIL representative this is easy to do, by only changing the argument all=TRUE.

Step 4: Since we now have the data set total that contains all the companies that have a CIL representative we use the data.table total to plot the size of the company by the size.