ENS data challenge - Report
MAP569 - Machine Learning II

Nissim Maruani
Jean Baptiste Soubaras

March 2021

Contents

1 Introduction

For this paper, we chose to tackle the data challenge given by the ENS, which aims at classifying different traders according to different features into three categories : HFT, NON HFT and MIX.

The file *x_train.csv* contains 35 features taken each on a different dates for each trader. The file *y_train.csv* only contains the classification of the traders, but not that of the rows. In order to classify the traders, the first step is thus to classify the different rows of the train file.

2 Classifying the rows of the train file

We first filled the missing values of the file with an average of their respective column. Then we tried different method to classify the rows.

The first is the naive solution, which consists in considering that all the rows of a HFT trader are HFT, all that of a NON HFT are NON HFT and those of a MIX trader are MIX. It is probably quite close to the real for the HFT, because a HFT trader has more than 85% of their rows as HFT, but clearly less for the MIX traders who have barely more than 50% of their rows as MIX, and the NON HFT traders can even theoretically not have a single NON HFT row at all.

Another solution is the use of clusters : using k-means clustering which is the best adapted for large data sets with few clusters and taking into parameter the number of clusters, we simply clusterize all the rows of the train file into 3 clusters. We then count the representation of each clusters among the cumulated rows of each type of trader, and like that we know that the cluster which is the most represented among HFT traders is that of HFT rows, the cluster the most represented among MIX trader is MIX, and the last one is NON HFT. However, when implementing this, we got a problem : the cluster the most represented among both HFT traders and MIX traders was the same. So we decided that our clustering should be more precise.

In order to do so, here is the idea : first we cluster the rows into more than 3 clusters. Then we attribute each clusters to a category the following way : starting with HFT, we gradually attribute the HFT label to the most represented unlabeled cluster among HFT traders, until we reach at least 85% of representation of the selected clusters. Then we do the same thing for MIX traders until the MIX clusters contains at least 50% of the rows of MIX traders. The clusters that left unlabeled are then labeled NON HFT. We implemented this solution with 6, then 9 clusters, and each time the problem was the same : once we labeled the HFT clusters, we couldn't reach 50% of representation of the MIX clusters among MIX traders, even when labeling all of the clusters left as MIX. So we tried to make compromises, considering that 75% of HFT rows was enough for HFT traders and 40% was enough for MIX traders.

We runned the algorithm with each of these implementations, and in the end, to our

surprise, that which gave the best results was the naive solution.

3 Classifying the traders of the test file

Once we had classified this way all the rows, we just needed to find the classification algorithm we could use. Since we had a large data set (more than 100.000 rows in the training file), the most efficient to us was the Stochastic Gradient Descent classifier.

We trained the algorithm on the training file, then we fitted the model on the testing file. As a result, we obtained a classification of all the rows of the testing file. So we attributed each trader a status HFT, MIX or NON HFT, according to the proportion of each type among their rows.

We had also tried other classifiers, the Support Vector Machine (that couldn't compute on so much data) and the K-Nearest Neighbors that gave less satisfying results.

4 Conclusion

To put in the nut shell, we first filled the missing data using the average value of each column, then we attributed to each row of the training file a status HFT, MIX or NON HFT (we tried to clusterize using K-means clustering but the naive solution was more efficient in the end). Then we trained the SGD classifier on the training set to apply it on the testing set. Eventually, we created the file $y_test using the results of the algorithm$.