1st Kaggle : Safe Driver Prediction

As soon as we got ourselves inside the subject and its various components, we quickly identified two main problems:

- With columns missing between 20% and 70% of the data, how to fill in the **missing data**?
- In a second time, which **prediction algorithm** to use to solve our problem?

Problem of missing data

- First idea:

In order to solve the problem of missing data, we had two distinct ideas. First, we thought of replacing the missing data - the "-1" - by the average of the neighbouring data, then by their median or the most present value for the categorical data. We thought that would add consistency to the data set. It turns out that the score we obtained with this pre-processing was lower than without - in parallel we implemented an algorithm XGBoost on which we will return later.
How to explain it? We can suppose that the -1 add information to the dataset: "these people did not fill in all the information". This can impact the likelihood that customers will make a claim.

- Second idea:

We have refined our model to complete the dataset. For a user X for whom the data is missing, we try to calculate it on the users who have the same profile rather than calculating the average over the whole dataset. For this, we implement a clustering algorithm. With a **k-means algorithm**, we create 15 client clusters. Our individual X belongs to a cluster C(X) and we calculate the median on this set. For categorical data, we use the category most present in the cluster. Thanks to this data processing, we obtain a slightly higher score than the initial score.

Pre-processing of the dataset

Data not being named, it was difficult to go further in preprocessing by trying to link several categories as we did in kaggle 2.

We kept on looking for correlations between the features. We have done this for a HeatMap of the correlation matrix. We see that there are few correlations between the data, some columns are even completely decorrelated from all the others.

We also tried to reduce the dimension through two algorithms:

- **Principal Component Analysis**: We seek to highlight different "populations" in order to treat them separately. However, the results were not relevant since one of the eigenvalues largely dominated the others that seemed to indicate that the intrinsic dimension of the space was 1. As it seemed unlikely, we assumed that the PCA was not adapted to our dataset, probably because the correlation between the data was too weak.

- **Linear Discriminant Analysis**: Here, we sought to highlight a projection axis differentiating drivers according to the output label. We obtain a very slight separation according to the label; a tiny part of risk drivers is highlighted but this is difficult to exploit.

We therefore decide to use the provided datasets, completing only the missing data, and to look more at the prediction algorithms.

Prediction algorithm

Which algorithm could we use? We explored three paths:

- Neural network
- LGBoost
- XGBoost

We quickly realized that the **neural network** was not adapted to the data we had - at least the neural network we coded for the first assignment. Indeed, we had only 3% of positive data, so minimizing the cost function was not relevant; Especially since the goal was to maximize the gini index (score 0,258). Rather than optimizing the neural network, we have instead turned to Boost algorithms.

With a simple **XGBoost** algorithm and default settings, we immediately got a score of 0.276. In order to optimize the parameters and given the limited number of submissions allowed by the site, we have written a cross-validation algorithm to implement a search for optimal parameters resembling a **grid-search** (we define the possible values for each parameter, we test the possible combinations and we keep the one with the best score). We finally get a score of 0.282.

Then we implemented a LGBoost algorithm, following the same process with a similar score evolution.

Finally, by averaging the predictions of the two algorithms, we obtain a final score of 0.284.

Potential improvement:

- test several Boost models with different parameters and average results
- optimize the Neural Network then average with the Boost algorithms.