MELLERIO Antoine
MORVAN Théo
TRIPARD Arsène

Data for finance

Interest rates strategy for banking loans

October, 2022

Table of contents

A. Round 1

a. Estimate the default probability

Data Exploration

Before diving into modeling any default probability, we first took time to analyze the data to mitigate any potential issue with it. By doing so, we noticed several elements:

1) there was no missing entry in the whole dataset. This dramatically reduced our effort on this part of the project, but we know it does not necessarily reflect reality
2) the repartition between non-defaulters and defaulters was, as expected, extremely skewed : 80% of non-defaulters vs 20% of defaulters : we would need to tackle this imbalance when training our models

Once we had identified this potential pitfall, we moved on to the next step, i.e estimating the default probability.

Modeling Assumptions

First, we determined which features should be used for the predictions :to avoid any bias, we decided not to use the gender feature, as it would by definition be extremely biased. Then, we preprocessed our data (logarithmic transformation for the income, OneHotEncoding for employment status) and moved on to the classification task. Indeed, we considered the problem as a classification one (1 being default, 0 none), train ML models and then gather the predicted default probabilities for the new dataset.

We tried different ML models and different combinations of hyperparameters to our models. To reckon with the class imbalance, we first put higher weights on the default entries : the cost of not predicting a positive (i.e a default) for the model was higher than the opposite. Secondly, we optimizing our models, we relied on the recall score and the f1-score. The recall measures how well-identified the defaults are by a model, while the f1-score is the harmonic mean between the recall and the precision: a balance between properly classifying the positive results while not predicting every entry as positive. We did so to reckon with the winner's curse and our lack of comprehensive information about clients. However, the results given by a mix of XGBoost and LGBM (state-of-the-art models for classification) led to very conservative results with a default probability distribution very centered around the mean, and very high interest rates. To mitigate this, we build upon the results of these models a Logistic Regression, which would try to rebalance a bit the results. As explained in the market outcome, this choice was a mistake : we had too low interest rates.

b. Rating strategy

Once the default probabilities are determined, we need to set our rating strategy. Two different things are to be taken into account : rentability and competitiveness.

Profitability

The main objective is profitability. It is to say that our interest rates must be high enough to compensate for the loss of the clients that default. We hence define the break-even rate $\overline{r}_i$ of the client i as following, PD_i being his default probability :

$$\overline{r}_i = \frac{PD_i}{1 - PD_i}$$

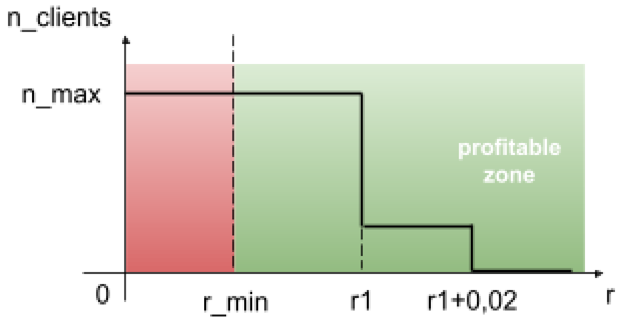
All the client's rates that we are going to offer will be greater than their break-even rate : $r_i > \overline{r}_i$.

Competitiveness

The second factor guiding our strategy is competitiveness. Indeed, clients are going to accept the offer with the lowest interest rate. However, it is important to notice that all clients already have a favorite bank, for which they are willing to have a rate 2% greater than the other banks.

The following graph represents the evolution of the number of clients we get depending on the rate we fix, in the simple case where all clients have the same profile. r_min is the break-even rate and r1 the lowest rate set by one of the competitors.

Number of clients obtained depending on the rate fixed



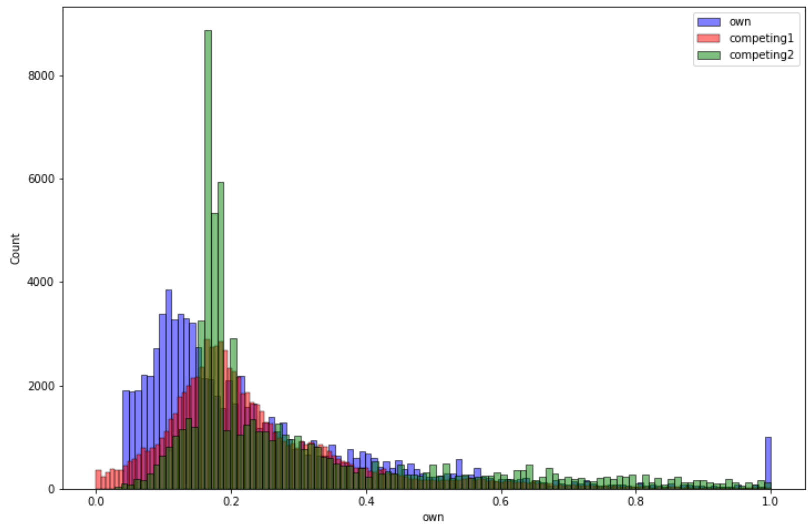
The strategies of the competitors are relatively unpredictable : they could either optimize the benefit per client rising r or the number of clients lowering r. We hence decide to set our rate to ensure that we keep the clients whom we are the priority of while maximizing their rate :

$$r_i = \overline{r}_i + 0,02$$

c. Market outcome

After Round 1, we got 15.7% of market share but landed at -6.7M profit.

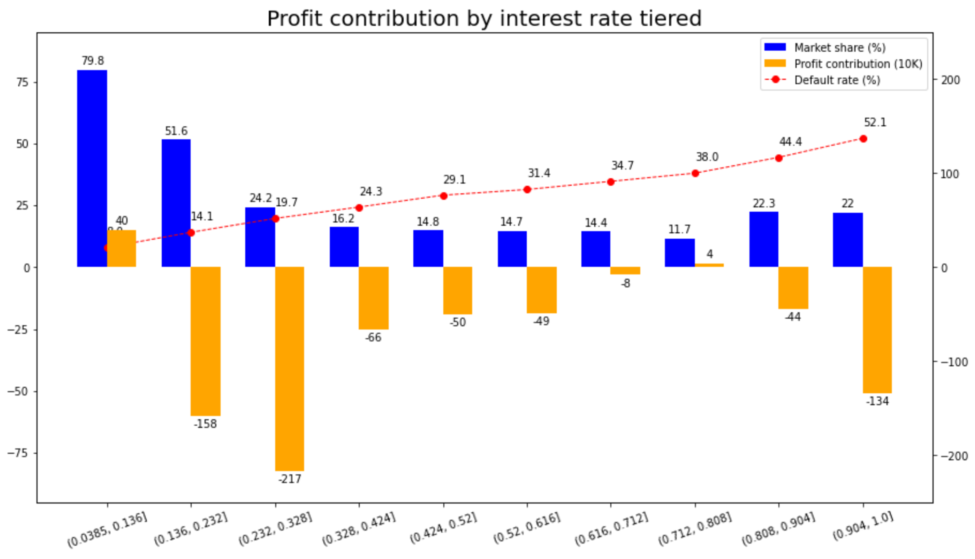
Distribution of rates in the first round



Comparing our rates to the competition, as it is done just above, we understand the following points :

- **we forgot that rates above 100% were truncated to 100%.**
  ↳ it's likely that we are losing a lot of money from junk loans at 100% rate. We should have refused them rather than compensated for their risk with >100% rate.

- **for safe borrowers, we are underpriced compared to the market.**
  ↳ either both our competitors are overpricing this segment, or we are neglecting some of the risk. We are probably the market leaders for safe borrowers, but it doesn't mean we are profitable.



The graph above groups the loans by interest rate, and plots the market share we get, and the contribution of this segment to our overall profit.

- **we are doing well on very small interest loans, in other words on super safe borrowers (less than 10% default rate)**
    - ↳ as we showed earlier, we are way cheaper than our competitors and therefore we managed to secure 80% of the "super-safe" market and make a positive profit at 400K.

- **on little risk (rate in [0.13, 0.32]) we perform very poorly: massive losses and not dominant in market shares**
    - ↳ we are underpriced, but this time we see that we don't get massive market shares like before. Two explanations are possible : either our competitors are underpriced as well, or our competitors are well priced and we are on average less than 0.2 below them, so they get their typical borrowers (bank 2 gets digital2 borrowers).

We have some information with predictor digital3 that our competitors do not have. Conversely, our competitors each have an exclusive predictor (digital1 and digital2). **Based on round 1, is it a game changer ?**

It's hard to say precisely because the final models we kept (LGBMClassifier and XGBClassifier) are black boxes. Therefore, we cannot explain a prediction based on the input predictor values. Yet, when plotting the feature importances we note that for both LGBM and XGB, the feature digital3 happens to be the 2nd most informative feature.
So, it's likely that even with the same model, the 3 competing groups would never predict the same default probability.

B. Round 2

a. Improve the default probabilities

After analyzing the results of the first round, we quickly realized that the Logistic Regression upon the predictions was counterproductive and led to too low interest rates. We thus took this step out of the process and focused on improving the f1-score. Why? We want to have the lowest interest rates for the non-defaulters and the highest for the riskiest clients to make the highest margins possible.

Unfortunately, we struggled to make significant progress from the first round (after having removed the logistic regression on top). We investigated QuadraticDiscriminant Models (which fit Gaussian mixtures) to get this "bimodal" distribution of default (low probabilities for safe borrowers and high for the others) and other similar methods but it did not turn out to be conclusive… We tried different approaches (linear mix of predictions from different models, stacking predictions and different types of algorithms), but we did succeed in dramatically improving our results. A linear combination of XGBoost , LGBM and Logistic Regression results yielded the best results.

To find the optimal combination of these different predictions, we leveraged the package Optuna, a hyperparameters optimization package relying on bayesian optimization and parallelized computations for fast and efficient optimization of black-box functions.

b. Redefine the pricing strategy

First of all, we observed a loss on almost all intervals, indicating rates globally too small. We hence decide to add a constant value b to all the rates :

$$r_i = \overline{r_i} + b$$

We then want to avoid the variance on the benefits inducted by clients having high default probabilities. We thus add a rate term proportional to the break-even rate :

$$r_i = a * \overline{r_i} + b$$

In order to find the best values for a and b, we apply a backtesting optimization technique, maximizing the benefits we would have had at the first round with different (a, b) couples.

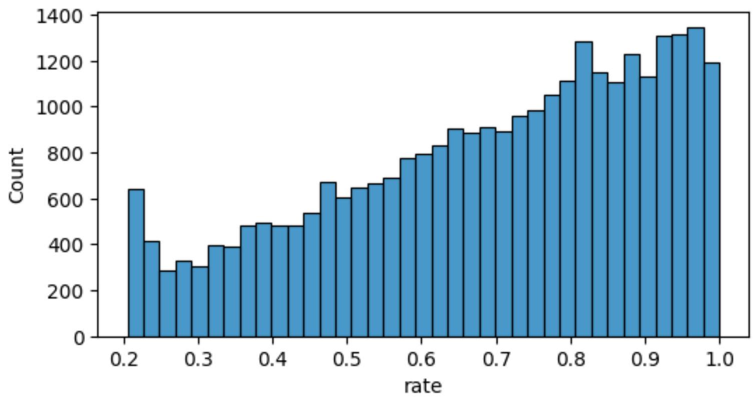
We are aware that the strategies of the competitors are meant to evolve but we assume that the winner’s strategy is not going to evolve drastically either. The code of the optimizer is to be found in the annex pages. Moreover, the best relationship could as well be of degree 2. This is a lead to explore for further improvement.

The final strategy would have allowed us to have a benefit of 4276$ with a market share of 0,006 % (6 clients). We are aware that the market share is low. However, this can be explained by the overall poor results of all the teams that often lost money in the first round. We expect the teams to higher their rates in the second round, and thus to have a greater market share than 0,006%. The final formula is the following :

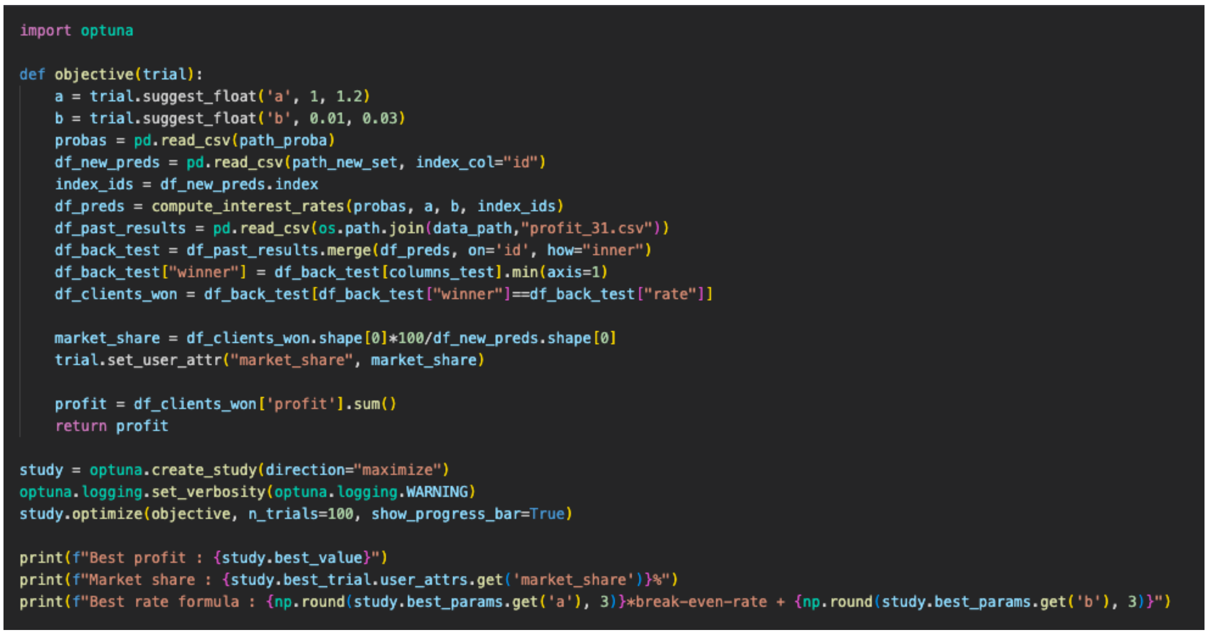
$$r_i = 1,608 * \overline{r_i} + 0,182$$

Applying this strategy to the dataset of round 2, we offer a loan to 30% of the applicant, and obtain the following rate distribution :

Rate distribution of round 2



Annexe



```python
import optuna

def objective(trial):
    a = trial.suggest_float('a', 1, 1.2)
    b = trial.suggest_float('b', 0.01, 0.03)
    probas = pd.read_csv(path_proba)
    df_new_preds = pd.read_csv(path_new_set, index_col="id")
    index_ids = df_new_preds.index
    df_preds = compute_interest_rates(probas, a, b, index_ids)
    df_past_results = pd.read_csv(os.path.join(data_path,"profit_31.csv"))
    df_back_test = df_past_results.merge(df_preds, on='id', how="inner")
    df_back_test["winner"] = df_back_test[columns_test].min(axis=1)
    df_clients_won = df_back_test[df_back_test["winner"]==df_back_test["rate"]]

    market_share = df_clients_won.shape[0]*100/df_new_preds.shape[0]
    trial.set_user_attr("market_share", market_share)

    profit = df_clients_won['profit'].sum()
    return profit

study = optuna.create_study(direction="maximize")
optuna.logging.set_verbosity(optuna.logging.WARNING)
study.optimize(objective, n_trials=100, show_progress_bar=True)

print(f"Best profit : {study.best_value}")
print(f"Market share : {study.best_trial.user_attrs.get('market_share')}%")
print(f"Best rate formula : {np.round(study.best_params.get('a'), 3)}*break-even-rate + {np.round(study.best_params.get('b'), 3)}")
```

Figure 1 : Optimization of rating strategy through backtesting