Implement a Scoring Model

October 2021
Data Scientist
OpenClassroom Project 7

TABLE OF CONTENTS

Introduction

For project 7 of my data scientist degree, we were tasked with deploying a machine learning model online. The goal of the model is to predict the probability of a defaulting on their loan **with little or no credit history.** With the model, we must also provide an interactive dashboard to help the customer relationship managers interpret the model decision.

Dataset overview

The data is from this Kaggle competition. The organisation hosting this competition Home Credit Group an international consumer finance provider operating mostly in Asia. The majority of the contract types in the data was for *Cash loans*.

"Cash loans, which include multi-purpose financing, are typically not conditional on the purchase of specific goods or services from a specific retailer. They can ordinarily be used for any purpose (subject to local regulatory requirements) including, among other things, the purchase of home appliances, travel, home renovation, education, wedding expenses and medical expenses."

The dataset *instalments payments* seemed to suggest that these loans could have up to 50% interest rate.

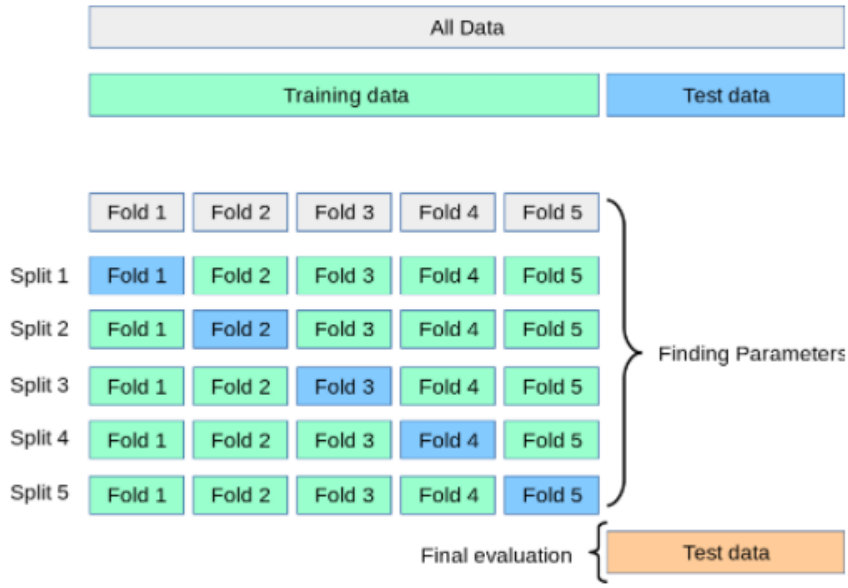
Model

We were asked to select a Kaggle kernel to focus on the model interpretability, optimization and comprehension. I selected this one and modified it. It uses LGBMClassifier "a fast, distributed, high performance gradient boosting framework based on decision tree algorithms". It had good ranking, was well written, which made the modifying process easier.

The notebook uses every dataset to do a substantial work of features engineering, creating min, max, mean columns as well as ratio when relevant. I did minimal modification to the original notebook.

Created a variable where I could select a percentage of the original dataset. It separates the customers based on the amount of credit they requested and uses one of the groups. I wanted to reduce the amount of data because of hardware constraints and this seemed preferable to choosing randomly. If performance were increased, having 4-5 models would not be operationally difficult to implement. This does assume that the currency of the Credit is uniform across all the data, which we have no guarantees.

The LGBM model then fitted with a standard StratifiedKFold



I did not find it pertinent to change the hyper parameters. There is a noticeable imbalance in our classes to predict. Only ~8 % *defaulting* customers. I did some testing with some resampling techniques, but it did not have a noticeable impact on model performance.

To increase the performance and interpretability of our model, I then reduced from the initial 592 features to 63. I used a very rudimentary process involving looking at the LGBM *feature importance* attribute. A more robust method would have been using RFE ("Recursive Feature Elimination"), but I maintained very similar scores on the evaluation metric of the model.

Optimization metrics and cost function

As stated above, we are in an imbalanced binary classification problem, so special care needs to be taken when evaluating the performance of our model. A typical accuracy test with a model predicting every instance in the majority class would have an excellent accuracy score. The classic approach and the metric used to score the Kaggle competition would be to use the "Area under the curve" between the predicted probability and the observed target." It's standard in this type of problem, every customer that defaults on their loan is much more impactful than a customer who repays their loan.

		True Labels	
		Repaid	Defaulted
Model Prediction	Will Repay	**True Repaid**	**False Defaulted**
	Will Default	**False repaid**	**True Defaulted**

True = Our model makes the **correct prediction**
False = Our model makes an **incorrect prediction**
Repaid = The outcome at the end of the loan, the customer **repaid** their loan with no issues
Defaulted = The outcome at the end of the loan, the customer **defaulted** on their loan

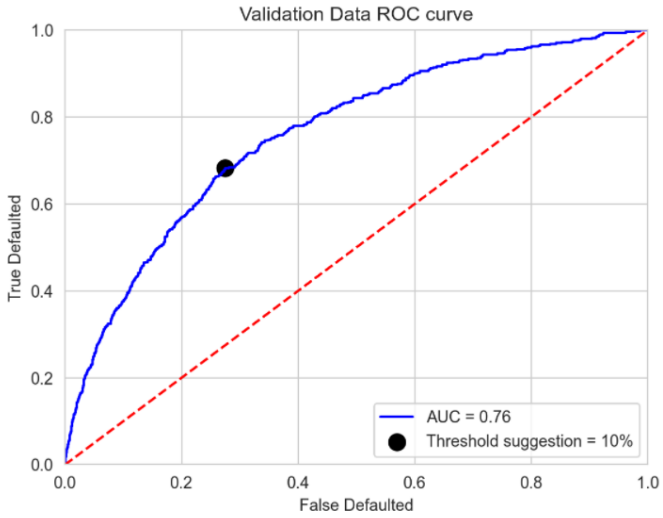
Area under the curve of our training data

Consider the table above and what each outcome would mean for our company:

- **True Repaid**: A customer who would have repaid their loan is awarded a credit; revenue received from the interest on the loan.
- **False Defaulted**: A customer who would have defaulted on their loan is awarded a credit; the balance of credit not reimbursed is lost.
- **False Repaid**: A customer who would have repaid their loan is denied a credit; lost revenue of the interest the loan would have generated.
- **True Defaulted**: A customer who would have defaulted on their loan is denied a credit; potential lost of revenue is averted.

The **False Defaulted** are much more impactful than the **False repaid**. Since the profit generated by one loan is only a percentage of the loss from an unpaid credit.

The area under the curve metric (auc for short) aims to maximize our **True Defaulted** vs our **False Repaid.**



The best score of the competition had an AUC of 0.81

It's the metric they used in the evaluation for the competition and one we used in our model during fitting. Our model output is a probability between 0 and 1, the *auc* metric aims to *"provides an aggregate measure of performance across all possible classification thresholds"*. For a more detail explanation on *auc* and *optimal threshold for imbalance data* you can visit this excellent post.

For this specific problem though, we can optimize our threshold to maximize profits instead of the *auc*.

Let's return to our 4 scenarios discussed previously. We can quantify the result of each scenario. We just need the following 2 columns in our dataset:

Potential Profit: The expected profit of the loan
Potential Loss: We take the historical data and calculate the **average loss** incurred by defaults **as a percentage of their credit.**

We then run, **for all possible** classification thresholds, the outcomes. By computing the outcome, we can find the **optimal cut off,** the threshold which generated the most profits.

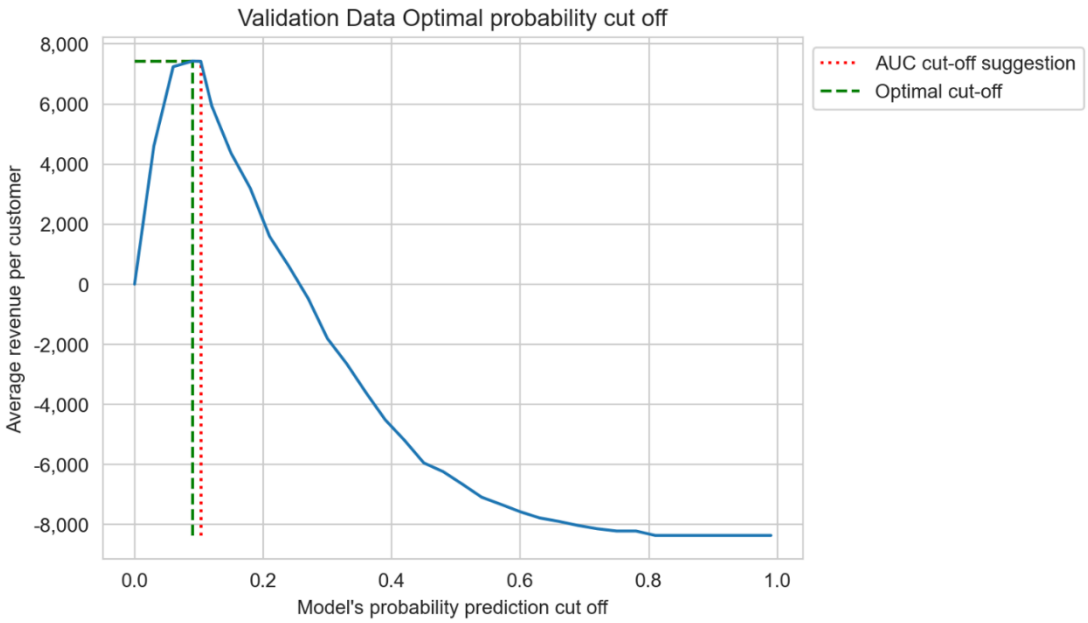
Unfortunately, we could not calculate accurately the *potential profits or potential loss* in our dataset. We did not have the number of installments to calculate potential profit, nor did we have at which installment did the repayments ended. Even if this information was not in the competition dataset, we can assume a finance company would have that information at its disposal. For the purposes of this project. I decided to create these two columns based on user inputs.

- **Potential Profit** = Expected Return, as a percentage of the loan
- **Potential Loss =** Average return before default, as a percentage of the loan

Let's look at the following graphs.

- X axis: **Each classification thresholds** (at 0 everyone is denied a loan)
- Y axis: **Average profit per customer**
- Red intersect: **Geometric Mean**
- Green intersect: **Threshold that yielded the most profits**

When the conditions are bad for the bank (40% reimbursed before default, 5% interest rate)



The optimal cut off is denying slightly more people loans then the original cut off.
Let's go back to our 4 scenarios:

- **True Repaid**: 5 % of their credit amount is added to the profit pool.
- **False Defaulted**: 60% of their credit is subtracted from the profit pool.
- **False Repaid**: The model denied the customer. No change to profit pool
- **True Defaulted**: The model denied the customer. No change to profit pool

For each threshold we sum our profit pool based on the scenario above and divide by number of loans. We see that the Geometric mean method to determine the optimal cut-off is very close to our custom threshold.
Now let's look at our confusion matrix. We will see the **SUM prediction Will Default** will be higher in our **Optimal Cut-off** then in the **Geometric mean.**

Confusion matrix at optimal cut-off (5000 customers)

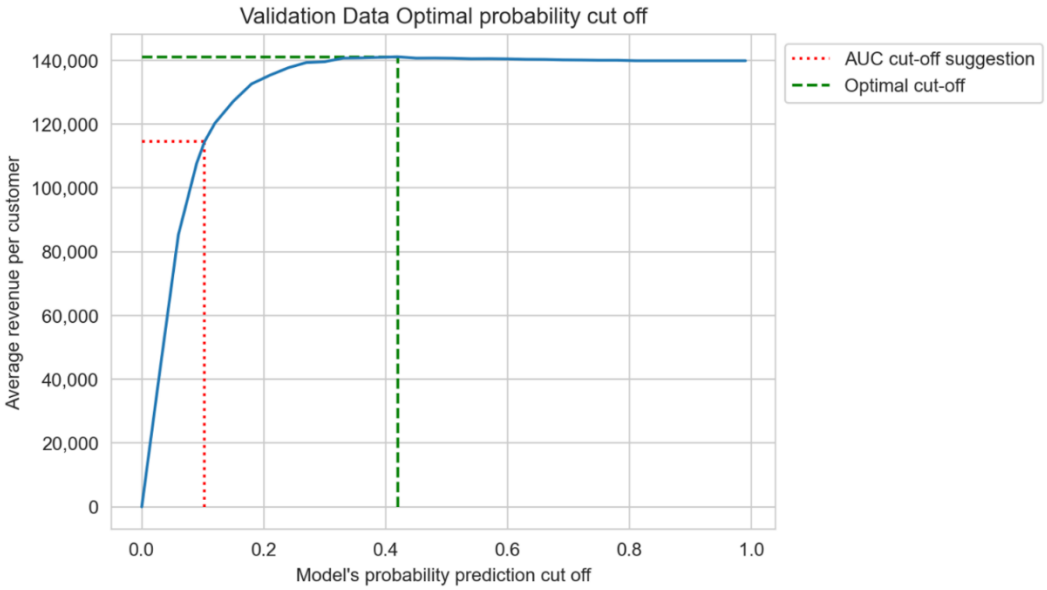
Optimal Cut-off		**True Labels**		
		Repaid	Defaulted	**SUM prediction**
Model Prediction	Will Repay	**3055**	**141**	3196
	Will Default	**1445**	**359**	1804
SUM True labels		4500	500	

Geometric mean		**True Labels**		
		Repaid	Defaulted	**SUM prediction**
Model Prediction	Will Repay	**3263**	**159**	3422
	Will Default	**1237**	**341**	1578
SUM True labels		4500	500	

Reducing by 18 the *False Defaulted* and increasing by 208 the *False repaid* yielded a bigger profit pool with *40% reimbursed before default, 5% interest rate*.

Let's have a look when the conditions are more favorable for the company.

When the conditions are good (40% reimbursed before default, 35% interest rate)



Now we are granting more loans then the initial threshold.

Optimal Cut-off		True Labels		
		Repaid	Defaulted	**SUM prediction**
Model Prediction	Will Repay	**4442**	**449**	4891
	Will Default	**58**	**51**	109
SUM True labels		4500	500	

Geometric mean		True Labels		
		Repaid	Defaulted	**SUM prediction**
Model Prediction	Will Repay	**3263**	**159**	3422
	Will Default	**1237**	**341**	1578
SUM True labels		4500	500	

With these parameters, the cut-off only denies 109 people loans vs 1804 in our first simulation.

To recap, while training the model, we input as an evaluation metric: *Area under the curve*. Once the model is trained, we use historical data on our validation data to get the optimal threshold for our model.

Model interpretability

When a customer relationship manager chooses a customer, we return a selection recap.

Selection Recap

106819

CUSTOMER ID

491823

CREDIT REQUESTED

45%

AVERAGE PERCENTAGE OF CREDIT REIMBURSED

88528

EXPECTED INTEREST ON LOAN

The customer model score and the cut-off score for approval

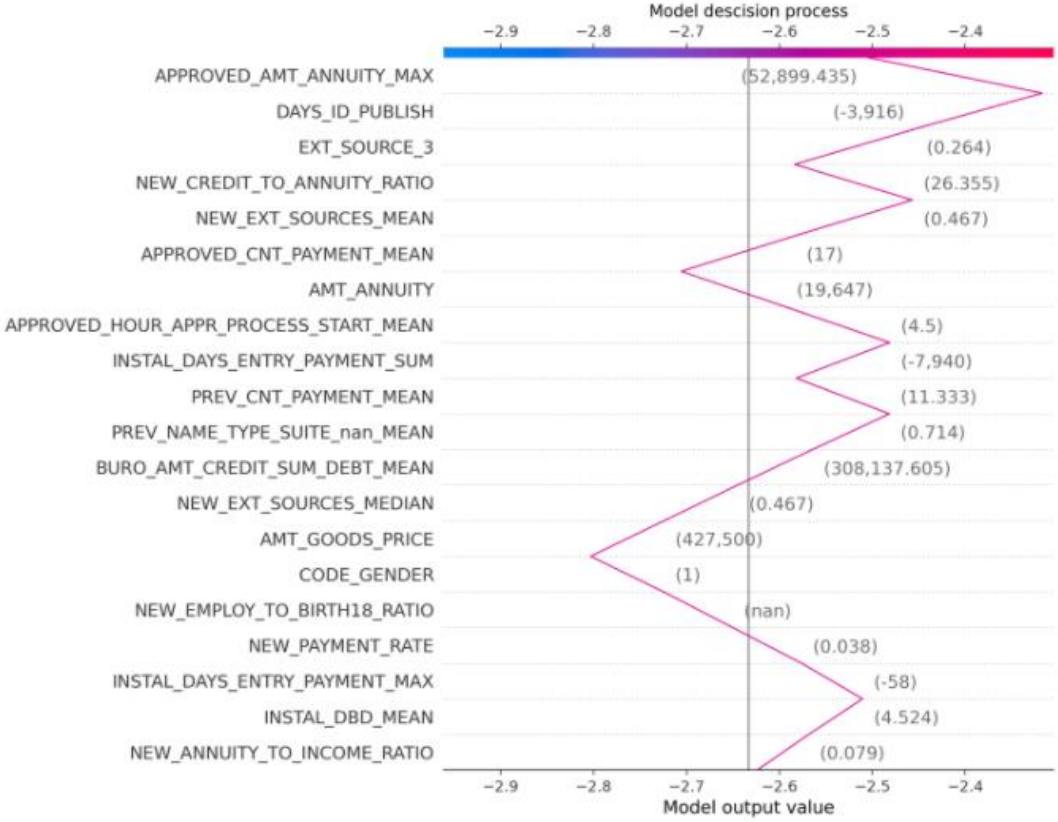
Model prediction

Customer Score: 7 %

Cut of score: 27 %

Customer Credit application is approved! 🏆 💰

Then we display how each feature affected the probability score, in order of importance



Possible optimizations and considerations

The optimization functions

Having a calculated profit generated per customer would yield much better result to calculate the threshold of the model than the technique we used in this project. Applying a percentage of the credit as expected profits is not a great solution since any changes in loan annuities would impact our variable to predict, whether they defaulted or not.

Production

This project serves as a demonstration, if we wanted to scale this approach to production, we would need to consider:

- Restructuring the data to optimise memory requirements
- Security of the customers information
- Monitor model drifting