A customized credit granting model to determine which clients to finance : validated with AUC.

May 2021

Marie-Lou Baudrin,	*Directed by*:
Cécile Brissard,	Paul Thavenot,
Serena Gruarin,	Guillaume Clément
Théo Lorthios	*Programming in*: Python

Sorbonne School of Economics, University, Paris, France

Note : With this article, are also available in the joined folder :

- ⩾ the scripts for managing the data shared in four sub-datasets,
- ⩾ the data-frame resulting from the sorting of the data and on which the modeling is carried out,
- ⩾ and the modeling script.

Contents

1 Introduction

The *Lending Club* [1] is America's largest peer-to-peer lending company, connecting borrowers with investors since 2007. Lending Club enables borrowers to create unsecured personal loans between $1,000 and $40,000. The standard loan periods are 36 months or 60 months. Investors are able to search and browse the loan listings on the website and select loans that they want to invest in. Lenders can thus base their reflexion on the information supplied about the borrower, amount of loan, loan grade, and loan purpose and can decide how much to fund each borrower. Investors make money from the interest on these loans. *Lending Club* makes money by charging borrowers an origination fee and investors a service fee.

One of the dangers of lending is the risk of default. Thus, it is essential to ensure the creditworthiness of clients upstream, in order to limit explicit (due to non-repayment) and implicit losses (opportunity costs of having lent to a bad payer). The objective of our project is therefore to design a model for pre-screening borrowers. To do so, we have access to a raw database, which we will clean and harmonize. From this, we will model the profile of "bad payers". Finally, we will select the model that allows us to limit the investors' losses as much as possible.

Our problem is therefore to build a turnkey tool for the investor aiming at optimizing the credit grant and consequently the profitability of the lending activity.

Thus, the selection of variables will be explained in Section 2. Section 3 is dedicated to the development of the modeling methods used. Sections 4 and 5 will give, respectively, the results implying the choice of the model and a way of maximizing profits. Finally, we will conclude in Section 6.

2 Sample : variables' selection

The database as it was had 2 260 701 observations and 151 variables, and 108 486 249 missing variables in total, as well as 38 object variables.
We performed this part using a dictionary found online [2]. This database had to be cleaned in two steps:

– At first, in a purely technical way, by making the database usable, i.e. by taking care of the missing variables, the variables of object type as well as other problems (for instance outliers, colinearity, heterogeneity...).

– Secondly, by using a business intuition to make the database interpretable to explain our target, i.e. by identifying the important variables for the modeling.

Note that the target we want to explain is the variable *loan_status* which had the following modalities: "$default$", "$In\ grace\ period$", "$charged\ off$", "$fully\ paid$", "$late\ (16-30\ days)$", "$late\ (31-120\ days)$", "$current$", "$does\ not\ meet\ the\ credit\ policy\ (fully\ paid\ and\ charged\ off)$". The terms we are interested in are only charged off, fully paid, default and late (31-120 days). We have therefore deleted the observations that had the other terms, grouped the terms charged off, default and late (31-120 days). Plus, our *target* is binary and is worth 1 if the individual is in a situation that is not profitable for granting credit, 0 otherwise. This is what we want to predict before granting a credit.

For the first part of this section, we divided the 151 variables among ourselves and proceeded as follows: - Deletion of the variables with more than 70% of missing values - Replacement of the missing values in an intelligent way: by micro model or with the help of statistics (median). - One hot encoding of polytomics variables - Dichotomics variables in binaries.

We then merged the four data frames into a single one that contained 2 258 141 observations (i.e. 2 560 less) and 194 variables (because of the one hot encoding).

We then moved on to the business intuition part. We deleted 29 variables that corresponded to variables that could not be known at the time of the credit granting. Indeed, the goal of our modeling is to know if we accept or not a credit request on a data set that includes people who have already paid back their credit or are in the process of paying it back. Then we removed 10 variables that do not make sense to explain our *target*.

Finally, we checked for multicollinearity by building a heat map using the correlation matrix. Each time we removed one of the two variables correlated at more than 80%, which removed a total of 12 variables.

We still had too many variables (80), so we decided to recover the most correlated with the *target*. In the end, we recovered a data frame with 1 372 770 observations and 29 explanatory variables. We can now proceed to modelisation.

3 Modelisation method

As requested in the topic, we split the final data frame in two, according to the date of the credit application. Thus, our data frame for training includes applications made before 01/01/2017, and has 1 127 248 observations, whereas the data frame for validation includes those made after this previous date and has only 245 522 observations. This division will allow us to post validate our model in 3 steps.

First, we will train our different models on the first data frame and then check that they are correctly working by comparing the predicted values of these different models to the values actually observed.

Secondly, we will go into the optimization of the internal parameters of the models (hyper-parameters). This step is essential to avoid having a model that is too precise and therefore not general enough, i.e. with a phenomenon of overfitting, or conversly, not precise enough and therefore pretty much useless. In order to do this, we use cross-validation selection algorithms. We divide the database in 5 sub-bases in which we will test the models. The idea is to see if the training sample is not specific to the base and that on each sub-base, the models work in the same way: we want a model built on homogeneous data.

Following the optimization of our models, we will finally be able to compare them and choose the best model. In order to do this, we must choose the selection metric that best fits the problematic. In our framework, we will use three metrics: AUC, Recall and the F1 Beta Score. Indeed, the recall and the F1 score will be preferred to the precision because we want to reduce the number of false negatives. In other words, we will prefer to refuse a credit to a person who can pay it back, than to grant on to someone who will not pay it back.

To briefly summarize the method in distinct steps :

1. We split our database into 3 parts: a training part for the model, a second validation part to optimize the hyperparameters of these same models and a last test part to calculate our prediction metrics.
2. We make a first test of our models with a defaults' parameterization and we compare their results.
3. We will optimize our models by varying their hyperparameters and we will cross-validate our sample in 5 K-folds.
4. We will compare all of our models using the AUC, F1 and Recall metrics, and confusion matrix ; we will take the most accurate model [3] [4].

4 Results

In order to model the risk of being a bad payer, we chose to test 4 different algorithms :

- The Logistic regression
- The KNN (K-nearest neighbors)
- The Random Forest
- The xgBoost

These four algorithms are the most widely used for supervised default risk prediction models.
Due to the first optimization of the 4 different models, we find the following metrics:

Metrics compareasons to define the ideal model

Model	Accurancy	Recall	AUC	Precision	F1 Score	F_β with $\beta = 2$	Confusion matrix
Logistic	**0.912***	**0.516***	0.829	0.943	0.667	1.0123	$\begin{pmatrix} 197181 & 1453 \\ 22711 & 24177 \end{pmatrix}$
KNN	0.859	0.469	**0.809***	0.715	0.566	0.8826	$\begin{pmatrix} 189856 & 8778 \\ 24903 & 21985 \end{pmatrix}$
xgBoost	0.904	0.511	0.840	0.976	0.670	1.011	$\begin{pmatrix} 198039 & 595 \\ 22945 & 23943 \end{pmatrix}$
Random Forest	0.903	0.510	0.824	**0.979***	**0.671***	**1.0124***	$\begin{pmatrix} 198037 & 497 \\ 22952 & 23936 \end{pmatrix}$

We notice in this table that the Random Forest outperforms the other models. Indeed, its accuracy and recall metrics are maximized but the AUC - the air under the ROC curve - is minimized by the KNN model. In terms of confusion matrices, the 125-branch Random Forest model appears to be the most efficient for detecting bad payers. In addition, this algorithm admits very few assumptions, unlike the logistic one.

Our final model therefore allows us to predict with a 2.1% $(1 - 0.979)$ error risk that our client will be profitable at the bank.
It is possible to improve the model with the optimized xgBoost version, but the number of data and the number of variables require us to enter into big data algorithms to which we do not have access.

5 Maximizing Profit

As our purpose is to automatically lend money to borrowers, our goal is also to maximize profit. We saw how to prevent to lend to someone who will probably defaults. With our model, our investment will return a profit and will be less risky but how to maximize this profit?

First of all, we enter the characteristics of our potential borrower and only if he pass with flying colours, we can go further. But the riskiest the borrower is, the highest the interest rate will be, so if we want to make an important profit we have to target high enough interest rate without jeopardizing our investment. We have to find a balance between risky to get a nice return on our investment and someone who will not default. With the help of *groupby* we can have the mean of the interest for the people who paid in full and those who did not, and also for each of those categories if it is for 36 months or 60 months.

It is in our interest to lend to high interest-rate-borrower so having the mean interest of people who paid in full for a 36-month loan gives us a limit. We should lend to people above this average but under the one for borrowers that go into default for 36-month loans. The length of the term has a big impact on the interest:

it is for this reason we decided to take it as a "decisional" parameter. It is not a perfect decision rule but it gives a good indication.

In this case, the average interest for people who paid in full for a 36-month loan is 11.78% whereas for the ones who will go into default it is of 14.14%, so we could say that in order to maximize profit when lending for 36 months we should choose loan with an interest between 11.78% and 14.14%.
In the same way, for 60-month loans, we should pick loans with an interest between 16.2% and 18.1%. Lending to borrowers that have a higher than average interest rate, we maximize our return on investment, but by lending to people under the average interest rate of defaulting borrowers, it limits the risk of not getting our money back.

Altogether, if we want to maximize profits we should only lend to long term borrowers as their interest rates are higher and it is less risky with our model which can predict if the client will defaults or not.

Moreover, we can estimate the profit of this model would bring to investors if it had been applied to the test part of the model. For this, we will consider a double hypothesis :

1. The investor's gain for a client in good standing is equal to the negotiated interest amount.
2. The loss associated with a bad payer is equal to the interest negotiated from the customer himself plus the implicit interest lost if the credit had been paid.

Thus, we can calculate the profit before the application of the model which would be \$ 86 244 089[1] according to our assumptions. Following the logic of our model, which selects the profitable clients in the amount of the loan from the others, we could therefore remove the losses due to the bad payers predicted by the model. Taking into account the errors of the model, we obtain a profit of \$ 315 145 228.
We can therefore calculate the contribution of the model to the platform's investors on this sample which would be \$ 228 901 138. Our model allows us to earn the equivalent of \$ 932 on average per client.

6 Conclusion

The final forecasting model seems to be a viable solution for the loan company. Indeed, by entering the values of the new customers - such as the credit object, its amount, the number of active accounts - we can know if it will be profitable or not for the company. Given the amount of data we had and the low error rate of our model, it is relevant to use this giant decision tree to pre-select future loan buyers. It will allow us to lend money automatically to borrowers without too many risks and thus to maximize our investment.

[1]You can find the details of the calculations in the script: Modelisation3 of python

References

[1] https://www.lendingclub.com.

[2] Jonathan CHAN. 2003.
https://www.kaggle.com/jonchan2003/lending-club-data-dictionary.

[3] Sarang NARKHEDE. “Understanding AUC - ROC curve”. *Towards Data Science*, 2018.
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[4] Ofir SHALEV (@ofirdi). “Recall, precision, F1, ROC, AUC, and everything”. *The Startup, Medium*, 2019.
https://medium.com/swlh/recall-precision-f1-roc-auc-and-everything-542aedf322b9.