RDS Project Draft Report

Tom Yang yy2949, Lyris Wei yw4182

Background

The ADS we propose to analyze in the project is Nicholas's solution for Give Me Some Credit. We found this project in Kaggle competitions. The host has published the training set, test set, sample entry and submission files. Nicholas's solution is the second solution under the Most Comments filter. All his code can be found on GitHub and notebook with explicit explanation and running results. Our works can be found on GitHub too.

The goal for this ADS is to help borrowers make better financial decisions in loan granting by improving a credit scoring system that makes a guess at the chance of default (delinquency). Participants need to build a model to predict the probability that individuals may have financial problems in the next two years. From Nocholas's github, we could know that he trained a XGBoost model on the dataset, which is able to attain private and public AUC scores of 0.86756 and 0.86104 respectively. One thing we noticed in Nocholas's ADS is that he mentioned younger people are more likely to default. Therefore we want to figure out if his model determines that age is a significant feature in this credit scoring system, and if there are more fairness issues in his model.

Input and Output

a. About the source of the data, it is provided by sponsors who held this Kaggle competition. In the data description, they didn't explicitly mention who collected the data and how, but vaguely said the data are from 250,000 borrowers. It is likely that the data comes from the bank system.

b. Next we look through the features one by one. The table has 11 columns in total, including 10 features and 1 predictor (y variable).

A comparison of all the data is shown in the table below. We draw the distribution of each feature in the notebook. But to make the report look simpler, we collect the data from `df.describe()`, and summarize them in the table below.

Feature Name	Description	Feature Type	Distribution	Notes
SeriousDlqin2yrs	Person experienced 90 days past due delinquency or worse	Boolean (1 or 0)	0 (No delinquency):1(experienced delinquency) = 94:6	Predicting Variable
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt	float	min: 0.0 50%: 0.15 max: 50708 mean: 6.05 std: 249.75	No missing value
Age	Age of borrower in years	integer	An approximately normal distribution for all and for those creditable people; multimodal for those that are not creditable	Can find the graph of each distribution in the code; No missing value
NumberOfTime30-59DaysPastDueNotWorse	Number of times the borrower has been 30-59 days past due but no worse in the last 2 years.	integer	120k are 0 among the 150k lines of data min: 0 max: 98	No missing value
NumberOfTime60-89DaysPastDueNotWorse	Number of times the borrower has been 60-89 days past due but no worse in the last 2 years.	integer	140k are 0 among the 150k lines of data min: 0 max: 98	No missing value
NumberOfTimes90DaysLate	Number of times the borrower has been 90 days or more past due.	integer	140k are 0 among the 150k lines of data min: 0 max: 98	No missing value
DebtRatio	Monthly debt payments, alimony,living costs divided by monthly gross income	float	mean:353.01 std: 2037.82 min: 0 50%: 0.37 max: 329664	No missing value
MonthlyIncome	Monthly Income	float	60% lower than mean. Also very imbalanced.	19.8% percent missing values.
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards)	integer	mean: 8.452760 std: 5.145951 min: 0 median: 8 max: 58	No missing value

NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit	integer	mean: 1.01 std: 1.54 min: 0 median: 1 max: 54	No missing value
NumberOfDependents	Number of dependents in a family excluding themselves (spouse, children etc.)	integer	mean: 0.76 std: 1.12 min:0 median: 0 max: 20	2% missing value

Table 1. Data Description

For correlation between each feature, with the predictor at the first row, we draw a correlation matrix to check their mutual relation. The graph is shown below.

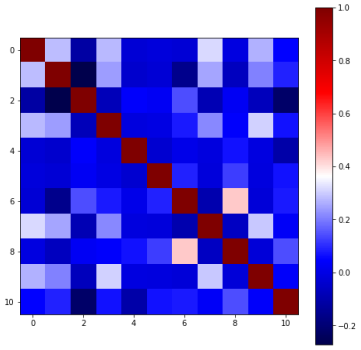


Figure 1. Correlation Matrix

The sixth row, representing `NumberOfOpenCreditLinesAndLoans`, and the eighth row, representing `NumberRealEstateLoansOrLines`, has a rather high correlation (0.43). All other features' correlation are lower than 0.25. So we might need a model more complex than linear regression to make a good prediction.

c. The predictor is the first column (first row of the table above), it represents whether a borrower has enough credits (to be trusted). The model predicts this value as a probability first, and then assigns label 1 if probability is greater or equal to 0.5, and label 0 if smaller than 0.5.

Implementation and Validation

a. As shown in the exploratory data analysis part, the author finds that most input features other than age are highly unbalanced. Fortunately, the model still provides a satisfactory performance on such data.

Another important step is empty value imputation. There are two major groups of missing values in the training data: 20% of `MonthlyIncome` is missing, and 2.6% of `NumberOfDependents` is NaN. Intuitively we believe these values can be important factors to decide whether a borrower is creditable or not. Therefore, we should not just throw these two columns away, but shall try to repair the data with some imputation strategies.

The author uses two different methods here. For `MonthlyIncome`, we can use either median or random normal imputation, with default method median (since this feature is highly imbalanced, mean may not be a good choice). For `NumberOfDependents`, since the missing value only takes a small portion and the values vary a lot, we use mode (0 here) for imputation.

There are some little revisions and removal of unreasonable or unnecessary data, in order to make the training data cleaner and the training process more effective. For example, the id column is dropped to avoid repetition with the default pandas index. Id column, with 1-150000 through the rows, will not be a helpful feature in training at all. Also, there are some rows whose `NumberOfTimes90DaysLate` is very large removed. By calculation the author finds that the total days late will be greater than 2 years: the overall time the data is collected and calculated. This data, without the preprocessing part, is very likely to become outliers and impact the model's learning. Thus, data cleaning, though being a nasty and unpopular job, still plays a critical role in the whole model lifecycle.

b. The machine learning model system is rather straightforward to understand. The author uses XGBoost, one of the most popular and successful ensemble learning packages in Kaggle competition that does not require a GPU (actually it’s because XGBoost cannot be

accelerated by GPU) but can still train very fast. It is based on gradient boosting, a method that ensembles a group of weak classifiers to become a strong classifier.

The author uses it to predict the y variable: `SeriousDlqin2years` with all other preprocessed columns as input features. Then, to find the best model, he uses random search cross validation to fine-tune a large group of hyperparameters. This process may take hours, and one hyperparameter, n_estimators, can significantly affect the training time for each k-fold. In fact, if we tune n_estimators as the original code, doing random search in [50, 100, 150, …, 750], then the cross validation will take more than a half day on one CPU, and stopping the process to make a checkpoint is not applicable in the given code. Using the best hypermarater set the author gave, we found that the change of n_estimator does not significantly affect the performance of our model. Thus we use the given lowest n_estimator = 50 to rerun our task. For other hyperparameters, we remove the 'gblinear' booster to avoid warning of its incompatibility with other hyperparameter values. The total training time now reduces to 4-6 hours on a Google Colab CPU, and we find that the same group of best hyperparameters in the original training process appeared the most here too.

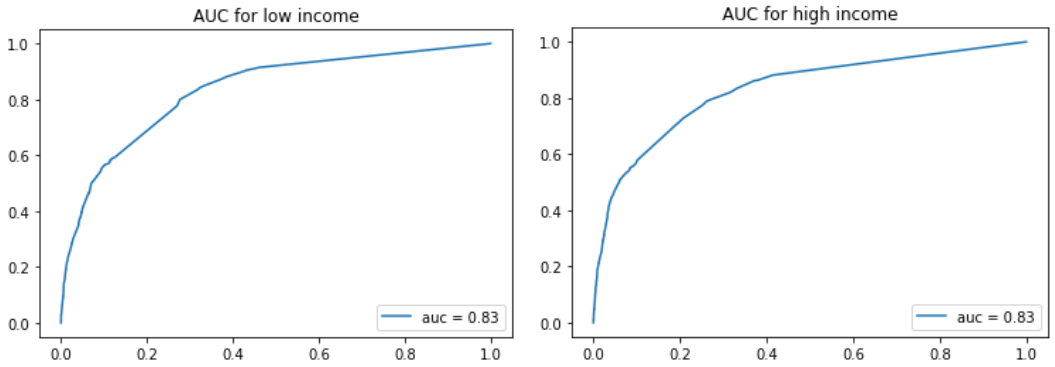
c. In the given code, the author uses all 150k training data for training, and predicts the 102k testing data with no true labels. The predicted result will be submitted to Kaggle for scoring, and only a score but not a true answer(label) will be given back. Validation is done automatically in the random search cross validation.

To ensure our later exploration of the model and dataset can work fine, we must not only have validation in random search CV. In fact, we need to make some changes at the beginning, by splitting the training data (all have y labels) into a training set and a validation (testing) set. We use train_test_split from sci-kit learn, with a ratio of 0.8:0.2, and random_state = {2019, 2020, …, 2023}. We try the training and testing with different splits and take the average, using the best model we found with cross validation all the time. The

AUC score of the author's model is 0.86 (according to the Kaggle scoreboard). Our model has an average AUC of 0.855 on the validation dataset. The best AUC result on the leaderboard is 0.869. Our result is close to both.

Outcomes

a. For this part, we will compare the model's performance across low-income and high-income groups. $4300 is the low-income cut-off. The model accuracy score for the low-income group is 0.918, while for the high-income group is 0.942. The AUC for both groups are similar, which is around 0.83.

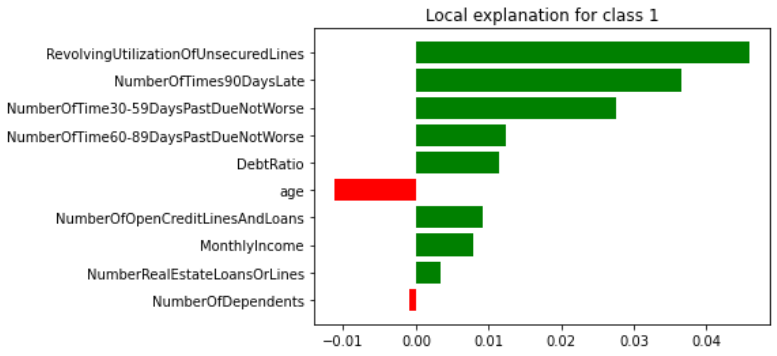


Therefore from the accuracy scores and AUC, we could see that the model for high-income groups has higher accuracy than the model for low-income groups. The difference between them is not significant, but there is still a possibility that this model has fairness issues. We will discover further in the next part.

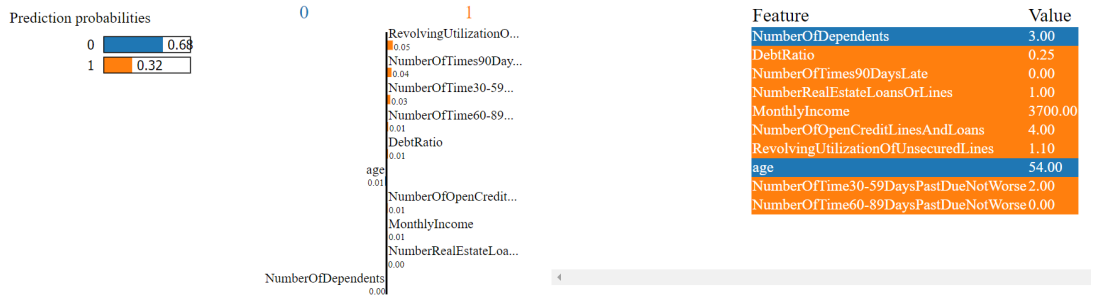
b. We will look for and quantify bias in this ADS to see if there are fairness issues by using a metric known as the "Disparate Impact Ratio". The disparate impact ratio is the ratio of positive outcomes (no credit issue) in the unprivileged group (in our case, young people or low-income group) divided by the ratio of positive outcomes in the privileged group (old people or high-income group). I selected these unprivileged/privileged groups of age and income since those are features that may cause bias to enter the machine learning process. The outcomes could be overrepresented or underrepresented. An acceptable lower bound of

disparate impact ratio is 0.8, which means there will be a disparate impact violation if the unprivileged group receives a positive outcome less than 80% of the time that the privileged group does. First, I check for existing bias in the actual data that our model is tested against with the age measure. We got a disparate impact ratio of 0.945. This indicates that the actual test split slightly favors the privileged group (older people). Then I calculated the disparate impact ratio of the predictions produced by the model, which is 1.0. Disparate impact ratio of 1 indicates complete equality, so there is no fairness issue with age feature. Then I check for existing bias in the actual data that our model is tested against with the income measure. We got a disparate impact ratio of 0.969. This indicates that the actual test split slightly favors the privileged group (high-income people). Then I calculated the disparate impact ratio of the predictions produced by the model, which is 0.9918. Such disparate impact ratio indicates nearly complete equality, so there is no fairness issue with income feature.

c. To find a clear intuition of how a given feature affects a prediction in our model and how important that feature is, I use LIME for further analyzing this ADS performance. A bar chart of feature contribution for this sample is shown below.



Green features contribute more towards getting financial issues within 2 years, red features contribute more towards the other class that not getting financial issues within 2 years. Next, we explain how we come to a particular prediction based on feature contribution.



The left table is the prediction probability of the two classes “0” class (not getting financial issues within 2 years) and “1” class (getting financial issues within 2 years). The middle chart shows the important features with their bounding values. And the right table is the actual corresponding feature value in the observation row passed, all these features contributed to the predicted probability for getting financial issues within 2 years. If a person’s age is less than 54, it may contribute to not getting financial issues within 2 years; if a person’s number of open credit lines and loans is greater than 4, it may contribute to getting financial issues within 2 years, etc.

Summary

a. This dataset is in general appropriate, but not the most qualified choice for this ADS. Comparing the training data with the testing data, we will find that the input features have very similar distribution, and even the amount of missing values in the columns are similar. Therefore a good performance on the training data can easily transfer to the testing data. As long as the real data distribution the ADS is facing has a similar distribution with the training data here, we can expect a quite good performance.

With that said, however, it is still far from perfect. As we can see from the leaderboard, even the best model cannot have an AUC higher 0.90. This ADS is used to decide whether someone can get a loan, and a chance larger than 10% of being misclassified is both unacceptable for the bank and the borrower, since that will mean the loss of a huge amount of money. If we want to say the ADS is good enough, perhaps a larger, more

comprehensive dataset that can help the model to learn the real world conditions better shall be required. As the competition participants have almost drained all the possible model architecture choices, a better direction for breakthrough is to improve the quality of this dataset.

b. We did not test its robustness. Considering that there are only about 10 input features and each of them has a stable distribution, it is unlikely that a small perturbation of the input will strongly influence the performance of the model.

The model is fair. The disparate impact ratio of 0.99 and 1.0 shows nearly complete equality in assigned privileged and unprivileged groups. No features are overrepresented or underrepresented.

The model is accurate too. If we use accuracy instead of AUC, we would get 93-94 with our model, 94-95 with the original model (larger n_estimators), and an even higher performance with the best model on the leaderboard. An accuracy of 95 shall be good enough to compare with human intelligence. Thus we conclude our model is accurate.

Nevertheless, it can still be further improved. With a desirable accuracy of classification, the bank can save a lot of time and money hiring people to go through the detailed data and loan history of every customer, but can leave the job for the machine. Nevertheless, as long as there are still some misclassified cases, some unqualified people will still be able to get the loan without being penalized. Since those who are actually qualified will certainly go to the bank frontdesk to argue when they receive a refusion, while those who are not qualified will always stay in silence after they receive the money they do not deserve, in general the customers will find it easier to get loans with an ADS than with a human intelligence service desk client.

c. This ADS can be a good example for the industry, but it is certainly not ready to be applied to the public sector. The industry can learn the strategies of data cleaning and

hyperparameter tuning from this ADS, as these methods are very universal and are proven practical. However, this ADS has poorly protected inputs. These feature values are very sensitive and special data of the customers, and are unlikely to have a lot of repetitions. Without additional processing, it is very probable to locate a specific customer by filtering with his private data here and some other information. In that case, the private personal loan situation may leak.

Meanwhile, the Kaggle competition never mentions where and how they collect the data. After this competition is closed for more than 10 years, we should attach greater importance to whether the customers know their data is collected for academic or commercial usage, and to what extent is their information being protected. If we further clean the data a lot, then a new model might be needed to repeat the whole training procedure.

d. We don’t know from the dataset description how representative the dataset is to the whole market, so perhaps we need a more universal dataset. Data shall be collected more carefully to avoid too many empty or wrong values as we have now. More careful privacy protection processing is needed to ensure the private information about the customers will not be exposed. Hyperparameter tuning has been proven useful in this task, and the hypothesis space here can be broadened to further explore better hyperparameter settings. Finally, and most importantly, there is one thing that every responsible data scientist should not forget in such tasks: while seeking a higher performance, pay attention to its robustness, fairness, privacy protection, etc.