ML Project - Credit Scoring

Rocco Totaro - 3159435 - rocco.totaro@studbocconi.it
Filippo Grandoni - 3187864 - filippo.grandoni@studbocconi.it
Ludovico Panariello - 3192212 - ludovico.panariello@studbocconi.it
Lorenzo Ravera - 3200788 - lorenzo.ravera@studbocconi.it

July 1, 2024

Abstract

This paper introduces a novel credit scoring algorithm that enhances predictive accuracy and fairness in financial risk assessment. Through extensive data exploration and feature engineering, the study demonstrates the model's capability to align with anti-discrimination laws without compromising reliability. Key findings indicate that incorporating Principal Component Analysis and ensemble methods such as Random Forest and XGBoost significantly improves the model's performance, achieving a robust ROC AUC score, thus providing a dependable tool for credit evaluation that respects legal and ethical standards.

Introduction - Data Exploration

Credit scoring is crucial for borrower assessment and equitable lending across demographic groups. This paper introduces a novel credit scoring algorithm that enhances predictive accuracy in financial risk assessment while maintaining ethical lending practices. The analysis begins with a comprehensive dataset, including variables like income levels, past financial behavior, and demographic details, chosen to represent diverse histories. This approach ensures that our findings are relevant across various lending contexts.

Special focus should be given to ensuring that legal requirements are not infringed upon in the development of scoring models, particularly with regard to age discrimination. But even though, according to the Chicago Tribune [1], "lenders can't deny you a mortgage because of your age," it can still be a factor when deciding to take out a mortgage or not. The foregoing, therefore, calls for the need to align our credit scoring algorithm with the anti-discrimination laws so that it does not prejudice older applicants.

In this context, thus, our analysis seeks to establish whether indeed age has a negative correlation with the likelihood of one to default on a loan. We aim to verify this relationship within our dataset and show that this factor is already adequately accounted for in our model without having to venture into that discriminatory practice territory. The negative relationship implies that the older an individual becomes, the less likely he will be to default. This thus validates the ethical use of age in our predictive modeling. Ensuring our model does not unfairly penalize older applicants is not only in compliance with legal requirements but also adds to ensuring fairness and reliability in our credit assessments.

Data integrity checks initially revealed some missing values, especially on the variable of "*Monthly Income*" and outliers which were addressed through appropriate data cleaning methods to ensure the accuracy of our analysis.

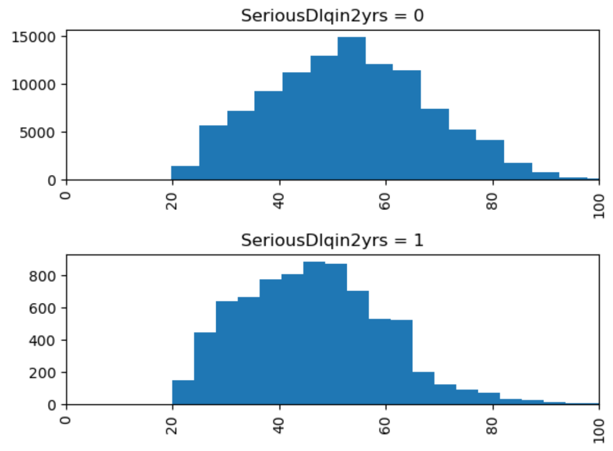

Figure 1: *Age Distribution*

The statistical summary illuminated the first insights into the distribution of some key variables and central tendencies. For example, the age of individuals had a relatively normal trend with a peak of about 52 years. On the other hand, the income variable was highly skewed, indicating that there was a probability of income differences among the individuals.

Table 1: *Statistical Summary of Key Variables*

Variable	Mean	Median	Std Dev	Min	Max
Age	52	51	14	21	90
Income	4500	3000	5000	100	20000
DebtRatio	0.5	0.3	0.8	0	1

After several tests, It seems that our best bet is to just replace the missing values with the median (instead of the mean because there are some extreme outliers in *MonthlyIncome*, who are making millions of dollars a month). The other column with missing values is *NumberOfDependents*. $>$ 50% of the non-missing values have 0 dependents, and if someone leaves the field blank it is likely to be due to not having any dependents, we'll replace these values with 0.

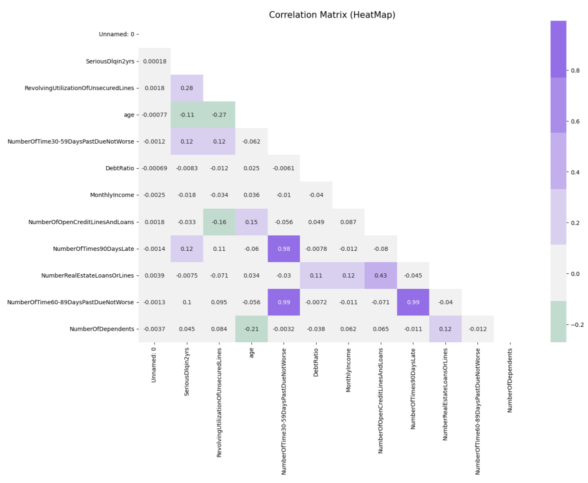

Figure 2: *Correlation Matrix Heatmap*

In 2 we can visualize that there are very high correlations (0.98 and 0.99) between the variables *NumberOfTimes90DaysLate* and *NumberOfTime60-89DaysPastDueNotWorse*. This suggests that individuals who are 90 days late are very likely also to have been 60-89 days late, indicating a trend in payment delays.

An important attribute of the age distribution analysis has been revealed in relation to credit performance. As observed in the histograms, it is those persons who were relatively young, and hence this observation contributes in important ways to the modeling, as it flags age as being a possible default predictor.

Outliers

An important focus was given to the identification and treatment of outliers, in particular, we have evaluated four different methods:

- **Percentile-based Outliers**: This method likely identifies outliers based on fixed percentile thresholds (e.g., below the 1st percentile or above the 99th percentile). The plot shows a mostly normal distribution with a possible outlier range marked but not clearly shown in this plot.
- **MAD-based Outliers (Median Absolute Deviation)**: The MAD is a robust statistic because it is less sensitive to outliers in the data than the standard deviation. Outliers are points that deviate significantly from the median. In this plot, no specific outliers are marked, but this method would typically identify those points far from the median after scaling by the MAD.
- **STD-based Outliers (Standard Deviation)**: This method identifies outliers based on their distance from the mean, measured in units of standard deviations. In this plot, there's a marked region in red around 1.0 on the x-axis, suggesting that values in this area are considered outliers, likely beyond 2 or 3 standard deviations from the mean.
- **Majority Vote-based Outliers**: This plot suggests a method where multiple outlier detection strategies are used, and points identified as outliers by a majority of these methods are finally labeled as outliers. The plot doesn't show marked outliers, which could mean either there are no clear majority-voted outliers or that they are not visualized in this plot.

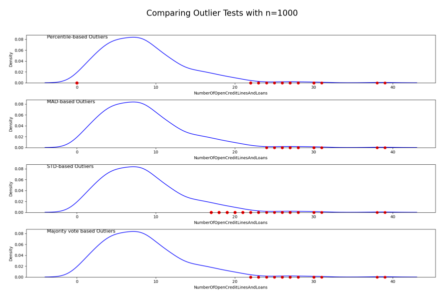

Figure 3: *Outlier Tests Comparison*

The identification of outliers using high values based on the MAD, STD, and majority vote techniques is particularly indicative of the *DebtRatio* variable alone, which does contain quite a large number of high-value outliers, reflecting financial stress or even an accidental data entry. The fact that the Percentile-based method did not detect any outliers at all might be a strong reason to believe that the data do not contain extreme values lying beyond normal statistical cut-off points or that the distribution is such that very high values are still within the top percentiles.

Feature Engineering

Credit risk assessment consists of various stages and in this process, feature engineering is the most important component for the development of powerful machine learning models. This part outlines the techniques used to develop new features and to fine-tune the already existing ones through the dataset that is made up of both training and test sets.

The features created are:

- **Income_dependent**: a ratio between the Monthly income and the Number of Dependents, to have an idea of the charge of responsibility of other people on an individual
- **Delinquency Ratio**: a ratio between the missed payments and the movements in credit and debit of an individual

The performance of Feature Engineered dataset with our best model (i.e., XGBoost, for analysis see in paragraph **XGBoost**) is very good: it improves our AUC ROC score by a very small amount, but at the same time we are considering really high values. The model gets **.864793**, which is a very good explanation of the model. With **Random Forest** we have a similar path than before; the improvement is very tight but still

consistent (i.e. .002), and the model gets **.8614**; althought the result is not at the level of **XGBoost**, and then we discard it.
Concluding, the Feature Engineering implemented seems to be quite interesting, specially if it is combined with parameters tuning. The **XGBoost** is the best even for our feature engineered dataset.

Model Running

Principal Component Analysis (PCA)

Before setting up the PCA, we want to do a study of variables explained variance to understand which variables include in the PCA. Basically we are evaluating which variables to choose in the PCA principal components explained since we want to include as much information as possible and, from a theoretical point of view we are searching the PCs that have higher eigenvalues, which is done because it means that these PCs has the highest explanatory power for our dataset. We want to avoid including variables which cannot help us, since it would be a waste of complexity and could bias our results, and then the PCA would loose its main aim.

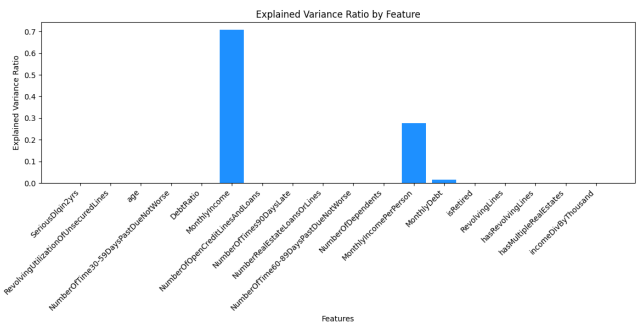

Figure 4: *Explained Variance of Principal Components*

From that it's clear for us that we can apply the PCA over 3 Principal Components: MonthlyIncome (i.e., the most explanatory), MonthlyIncomePerPerson and SeriousDlqin2yrs. Let's proceed:

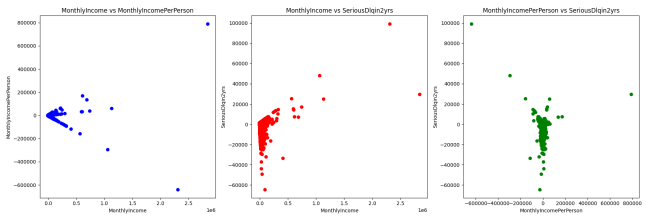

Figure 5: *PCA performances in 2D*

Then we have completed the PCA.
The **PCA** visualizations reveal that most of the dataset clusters at lower income and debt levels, with outliers primarily evident in *MonthlyIncome* and *MonthlyIncomePerPerson*. The clustering indicates common financial behaviors among the majority, while the outliers suggest variations that warrant further investigation. These patterns imply potential areas of interest for deeper financial analysis or customer segmentation, particularly focusing on those outlier cases where income and debt behaviors diverge significantly from the norm. Then we keep this results of PCA to subsequently evaluate the models in the following section 'Model Testing'. For what concern the **RandomForest**, it turns out that the ROC AUC performance is now *0.8615* which is slightly better than previous results, without the PCA implementation. Then effectively the **PCA** has some kind of improvements, and we have positive results, even if it's not a tremendous improvements; but since the model has already an high score of explanatory capacity, it is a very positive result.
Then it's convenient to analyze the results even in an XGBoost model environment, where the **PCA** performs ...
Concluding it can be inferred that **PCA** helps the model in explaining better the reality, but the impact is not as strong as we desired.

Evaluation Framework

To optimize our model's performance, we developed a parameter tuning function that utilizes grid search techniques. This function will be used to refine the parameters for XGBoost models, aiming to enhance their accuracy and efficiency. Additionally, to facilitate quick and effective comparisons of various models and datasets, we created a tester program. This program is engineered to automatically execute k-fold cross-validation, employing scikit-learn models across multiple datasets. This setup allows us to systematically assess the performance and robustness of different modeling approaches under varied conditions, ensuring that we can identify the most effective configurations for our predictive tasks.

Model Testing

In our whole study of different machine learning models to predict our target outcome, we discovered significant differences in the performance, which was measured by the ROC AUC score.
At first, we built an **L1 regularized logistic regression** model that allowed us to achieve a very high score over the baseline model with 0. 667 (with the vanilla implementation) to 0. 8437 (adding the L1 Regularizer). This served to improve the process of mitigating overfitting by penalizing the absolute size of the coefficients, which in turn aided the generalization process, yet performance was still not at its maximum, then we preferred to move on analyzing other models.
Next, we concentrated on the **Random Forest** model that turned out to be more accurate with an ROC AUC score of 0. 8630. Through the application of the elbow method, we found that the range of trees (20-40) would be the best option, as it was a very stable one and from the graph was clear that adding a tree

after that interval wouldn't be efficient in terms of performance. The Random Forest algorithm can be explained to be highly effective due to its ensemble methodology, where several decision trees are used to vote on the output and therefore reducing the variance and overcoming the problem of overfitting that usually arises with individual decision trees. We have dedicated an entire section to it below.

K-Nearest Neighbors (KNN) model was also tested, but it had lower ROC AUC scores than the Random Forest. KNN's performance proved to be poor, in particular, which can be attributed to the datasets' dimensionality and noisiness.
On the other hand, in comparison to **XGBoost**, the gradient boosting method which is a universal tool for solving various data science problems, achieved an ROC AUC of 0. 8645, which has our best performance with the implementation of vanilla dataframe, thanks to the tuning of parameters based on the GridSearch.

Consequently, **XGBoost** model emerges as the most suitable choice between the two models, having such characteristics as accuracy and model robustness that make it our preferred option for further development and deployment in the prediction of the specified outcomes. We will discuss it furtherly now.

XGBoost

Bagging *vs* Boosting Ensemble methods like bagging and boosting are pivotal in improving model performance by reducing variance and bias, respectively. In our project, we evaluated both methods to determine which is more effective for our credit scoring model.

Our analysis found that while XGBoost handles bias well and improves over single models, it is more prone to overfitting than Random Forest, which offers stability and ease of use without complex tuning. Despite Random Forest's advantages in interpretability and consistency, XGBoost's superior predictive performance led us to choose it for our credit scoring system to meet our goals of robustness and reliability.

As previously discussed, the **XGBoost** seems to be the best implementation for our purpose: in this section we will analyze more deeply what we have done.
Starting from the performance, it's clever how XGBoost with tuned parameters according to *GridSearch* (see paragraph on **Evaluation Framework**) performs robustly: approximately **.8647**. The model is performing well in distinguishing between the classes. The closer the ROC curve is to the top left corner, the higher the model's overall accuracy in classification. The above result implies that there is an 86% chance that the model will be able to distinguish between positive class and negative class.
It can be seen also from the confusion matrix, which follows:
The model shows a stronger ability to identify the negative class (no event) than the positive one, with a more significant number of false positives indicating possible issues with precision in predicting class 1. Even though the ability to get *TRUE* is very good, and in conclusion it is the best model we have found in our in-depth analysis

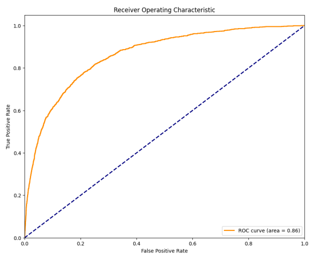

Figure 6: *ROC AUC curve of XGBoost*

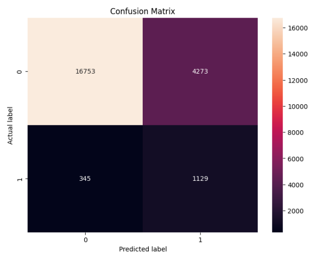

Figure 7: *Confusion Matrix*

Neural Network

We initially saw promising results with neural networks, achieving training and validation accuracies around 93%. However, despite these high accuracies, the model predicted the same outcomes for different inputs, indicating it learned a simple decision boundary. The classification report revealed a significant class imbalance, with Class 0 being much more prevalent than Class 1, which adversely affected the evaluation metrics and overall model performance. Particularly, the model performed well for the majority class but struggled with the minority class. A ROC AUC score of 0.5 indicates the model's discriminative ability is no better than random chance, highlighting its ineffectiveness.

Resampling Data

To address the issue, we decided to resample the data. Among the different resampling techniques we selected, two perform undersampling, the other two oversampling, each addressing the class imbalance problem from distinct perspectives. The methods we selected were:

- *Random Undersampling* for its simplicity and speed
- *NearMiss Undersampling* that selects samples based on their proximity to minority class samples
- *SMOTE Oversampling* for its ability to create a more diverse and representative sample of the

minority class

- *ADASYN Oversampling* that takes the benefits of SMOTE further by adapting the synthetic sample generation to the learning difficulty of each sample, therefore promoting a better learning environment for the model

It is important to note that results vary from run to run due to the randomness in sample selection across different techniques. Typically, Random, SMOTE, and ADASYN yield accuracy values ranging from 0.75 to 0.90, whereas NearMiss shows a range between 0.25 and 0.35. ROC AUC scores exhibit a similar pattern, falling between 0.70 and 0.80 for the first three techniques, while ranging from 0.50 to 0.60 for the last one. This is due to the fact that Random, SMOTE, and ADASYN, generally produce more balanced datasets by either reducing the majority class significantly or augmenting the minority class, thereby allowing models to learn a more accurate representation of both classes. On the other hand, NearMiss, which also undersamples the majority class but in a more targeted manner by focusing on those majority samples closest to the minority class, results in a loss of valuable information.

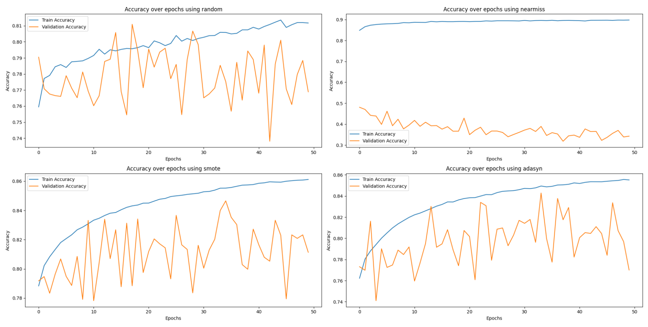

Figure 8: *Accuracy with different resampling methods*

The variability in validation accuracy across different resampling techniques underscores the impact of the method chosen for handling class imbalance on model performance and generalization. Our previous analysis is supported by SMOTE and ADASYN that show better generalization compared to NearMiss, with ADASYN showing some signs of overfitting or noise sensitivity as training progresses. Models trained with NearMiss and random methods show significant overfitting as indicated by high training accuracies coupled with much lower validation accuracies.

Final Considerations

A perfect next step may be represented by Mancisidor [2], in his PhD thesis developed two innovative methodologies for credit scoring, focusing on Deep Generative Models (DGMs) to bridge the gap between traditional neural network applications and more sophisticated machine learning approaches.

The study introduces a novel method for learning data representations through a supervised stage incorporated into the Variational Autoencoder framework, specifically designed to enhance understanding of customer creditworthiness. This methodology not only improves the dimensionality reduction of the input data but also captures the intrinsic creditworthiness of customers, enabling a segmented credit scoring approach that surpasses traditional methods.

Furthermore, the research extends into reject inference, proposing models that estimate the creditworthiness of rejected loan applications by leveraging probabilistic theories to address selection bias effectively. Additionally, the thesis adopts a multimodal learning perspective to develop models that can generate and utilize shared data representations for future credit data generation and classification.

References

[1] Chicago Tribune. *Age can play a role in your mortgage equation*. Chicago Tribune. 2005.

[2] Rogelio Andrade Mancisidor. “Deep Generative Models in Credit Scoring”. PhD thesis. UiT Faculty of Science, Technology, Department of Physics, and Technology, 2020.