Predictive Modeling for Credit Default: An In-Depth Analysis

HARSHIT KHANDELWAL (3210664), NICOLÒ CHEN (3206752), JORDI MURADYAN (3204735), DAVIDE CHENGRUI XIE (3196932)

Banks use a credit score algorithm to estimate the probability of default. This project tries to construct once such comprehensive model given the data. We start with the explorative data analysis (EDA), followed by cleaning the data and trying different techniques to get the best learner/classifier.

1. INSPECTION

The train dataset presents itself with a total of 112,500 entries and 12 features. The column *'unnamed: 0'* could be cleaned up for consistency and readability.
Our target variable *'SeriousDlqin2yrs'* has two classes: 0 and 1, representing non-default and default, respectively. Approximately 93.3% of the samples belong to the non-default class (0), while only around 6.7% belong to the default class (1). This suggests that the dataset is highly unbalanced, with a significant majority of samples being non-default cases.

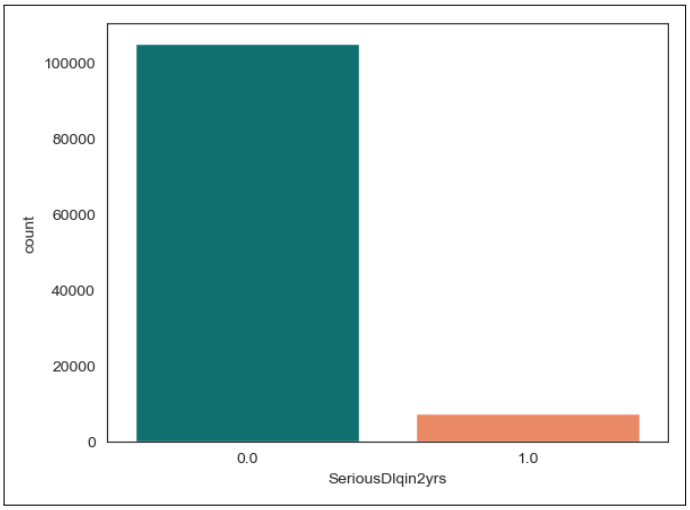

Fig. 1. Distribution of the dataset which results unbalanced.

A. Correlation Matrix

While exploring the correlation matrix, it's important to note that due to '*SeriousDlqin2yrs*' being a categorical binary variable, the correlation coefficients may not fully elucidate the relationship between variables and the target. However, notable correlations include a strong association among '*NumberOfTime30-59DaysPastDueNotWorse*', '*NumberOfTime60-89DaysPastDueNotWorse*', and'*NumberOfTimes90DaysLate*', with correlation coefficients hovering around 99%. Their similar distributions, as evident from the boxplots (Fig. 3.), underscore this relationship. Additionally, the correlation of 0.43 between '*NumberOfOpenCreditLinesAndLoans*' and '*NumberRealEstateLoansOrLines*' suggests a tendency for individuals with more real estate loans or lines of credit to possess a higher number of open credit lines and loans. Moreover, there is a negative correlation coefficient of -0.21 between 'age' and '*NumberOfDependents*', indicating a slight inverse relationship between age and the number of dependents.

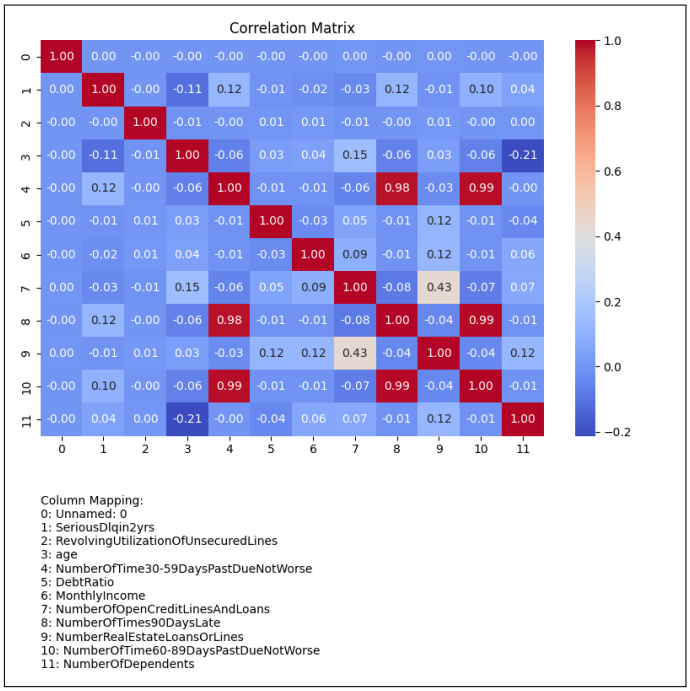

Fig. 2. Correlation Matrix

B. Examination of anomalous values

RevolvingUtilizationOfUnsecuredLines: 75% of the data falls below 0.56. Despite this, the mean results to be 6.26 due to considerable outliers with value way beyond the majority part of the data (maximum value is 50,708). However, we choose to retain these outliers as they provide valuable insights for our model, contributing to accurate predictions.

Age: it ranges from -1 to 103, with a mean of approximately 52.47 years. The majority of individuals fall within the age range of 41 to 63, as indicated by the interquartile range. The anomalous value that we have to deal with is the individual who has -1 as the age.

30-59, 60-89, 90 Late: we are referring to '*NumberOfTime30-59DaysPastDueNotWorse*', '*NumberOfTime60-89DaysPastDueNotWorse*', and '*NumberOfTimes90DaysLate*'. They demonstrate similar distributions, typically ranging between 0 and 17. However, the presence of outliers such as the values 96 and 98 in all three variables appears anomalous and requires further investigation, as they fall outside the expected range. These anomalies comprise a total of 202 observations, with 53.5% of them classified as defaults and 46.5% classified as non-defaults.

DebtRatio: Notably, the majority of the dataset, approximately 75%, exhibits a Debt Ratio below 0.86, indicating a relatively low debt burden for most individuals. However, anomalies are observed, with the maximum Debt Ratio reaching a staggering 329,664, significantly skewing the mean Debt Ratio to 353.83. Furthermore, a substantial proportion of individuals with missing values in the Monthly Income column, totaling 20,788 out of 26,218 individuals with a Debt Ratio exceeding 1, suggest a potential correlation between missing income data and elevated Debt Ratios, warranting further investigation.

MonthlyIncome: its mean is around $6,328 and it has a significant standard deviation of $14,134, indicating wide income variability. Approximately 20% of entries in this column, totaling 22,187 values, are missing. Individuals with missing monthly income values exhibit a high mean debt ratio of 1689.51 and a median of 1159. Furthermore, one individual in this group has the maximum debt ratio value of 329,664. Among those with zero income (1,227 individuals), the median debt ratio is 924 and the mean is 1,547.8, suggesting financial strain. Interestingly, about two-thirds of those with zero monthly income also report zero dependents, indicating a potential correlation.

NumberOfDependents: it predominantly consists of entries reporting zero (65107 observations) and one (19817 observations) dependent. However, there exists an outlier with 20 dependents. Additionally, there are 2,945 missing values in this column, accounting for approximately 3% of the dataset entries. Notably, all IDs with missing values in the '*NumberOfDependents*' column also have NaN values in the '*MonthlyIncome*' column.

After conducting extensive analyses and model evaluations, we observed that the removal of outliers did not yield significant improvements in the predictive performance of our model. Surprisingly, excluding outliers actually led to a deterioration in the model's predictive capability. Therefore, we made the decision to retain outliers within the dataset. These outliers contain valuable insights and patterns that contribute to the model's ability to learn and generate accurate predictions. This approach ensures that our model captures the full spectrum of data variability, resulting in more robust and reliable predictions.

2. DATA PREPROCESSING

Data preprocessing is a crucial step in any machine learning project. It involves cleaning and transforming raw data into a format that is suitable for modeling. In this project, we performed several preprocessing steps to ensure the quality and integrity of our data. To make preprocessing more compact and more professional, we fit all the modifications into a single pipeline making it easier to recreate on other datasets.

A. Removal of Observation IDs

The dataset contained a column labeled "Unnamed: 0", which held observation IDs. Since these IDs do not contribute to the predictive power of our models, we removed this column from the dataset.

B. Age Data Correction

We noticed an anomaly in the age data: one observation had an age of -1. Given that the minimum age for loan eligibility is 19, we considered this entry to be a data error. To correct this, we replaced the anomalous age with the minimum age of 19. Given the size of our dataset (approximately 120,000 observations), this single adjustment is unlikely to impact our analysis.

C. Negative Value Correction

In our dataset, the only feature that contained negative values was the age feature. However, to ensure the integrity of our data and to mitigate the risk of potential data entry errors, we implemented a correction step for negative values across all features. In this step, any negative values that might appear in the dataset would be converted to their absolute values. This ensures that all values in our dataset are non-negative, which is consistent with the context and nature of our analysis.

D. Payment Delay Rectification

The payment delay data presented some interesting insights. A small subset of 202 observations had values exceeding 95, while the majority were below 20 (Figure 3). These outliers all shared the same ID, and the same value was recorded across all three columns for these outliers. Given these observations, we concluded that these were likely data entry errors. To rectify this, we assigned a random number to these observations using the following steps:

1. Extract the maximum value from the rest of the observations in the respective column. 2. Assume that these values were intended to be high, but with some variation. 3. Assign the new value as the maximum value minus a rounded random exponential variable, ensuring that most assignments were near the maximum value.

E. Imputation of Monthly Income and Number of Dependents

We faced a significant challenge with the '*MonthlyIncome*' column, which had a substantial amount of missing data, reducing our usable dataset by approximately 20%. The '*NumberOfDependents*' column also had missing data, but to a lesser extent, around 2%. To tackle this, we utilized the IterativeImputer method from the sci-kit learn library in Python. The method functions as follows:

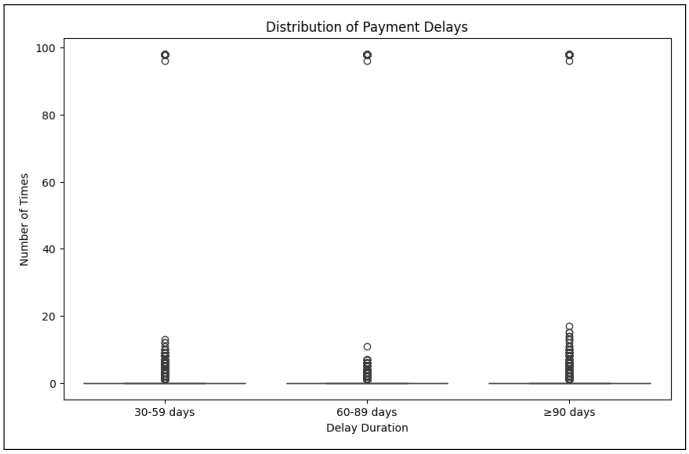

Fig. 3. Distributions of Payment Delays.

- **Model Training:** For each column with missing data, it trains a regression model using the remaining columns as features. In our case, '*MonthlyIncome*' and '*NumberOfDependents*' were the target variables, and the other columns served as features.

- **Missing Value Estimation:** The trained model is then used to predict the missing values in the column. This is done iteratively, meaning the model is retrained and the predictions are updated until they stabilize or after a predefined number of iterations.

- **Post-Processing:** After the imputation, some post-processing was necessary. Since '*MonthlyIncome*' cannot be negative, any negative predictions were set to zero. Similarly, '*NumberOfDependents*' should be an integer, so we rounded off any non-integer predictions.

- **Result:** This results in a dataset with no missing values, ready for further analysis or modeling. The change in '*MonthlyIncome*' observations after imputation is visualized in Figure 4.

This process not only filled in the missing values but also preserved the relationships among variables, making our dataset more reliable for subsequent machine learning tasks. The pipeline we built ensures that this complex process can be easily and consistently applied to any new dataset we encounter, enhancing the efficiency and reproducibility of our work.

It's worth noting that applying a linear regression on binary data like the number of dependents is not very efficient. However, this is not a significant issue in our case because for all the entries of NaNs in the number of dependents, the monthly income was also blank. It makes sense to assume that if one does not have a monthly income (bemacs students (us)), they also have no dependents. But the most important factor in the estimation of the number of dependents is that it will be removed from the estimation model, and it is only used to improve the predictions of monthly income, as those two are correlated in a sense (0 monthly income $\implies$ 0 dependents).

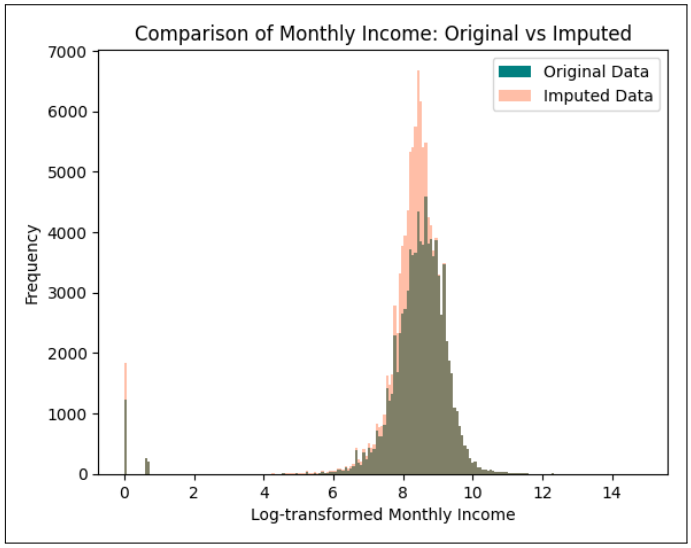

Fig. 4. Imputed Monthly Income.

3. COMMENTS ON THE CLEAN DATASET

A. Making sense out of the data

The main task at hand is to predict the chances of someone having serious delinquency in the coming 2 years. Now the data we have is highly unbalanced, that is a huge proportion of data belongs to one class and the other class has the remaining minority of data. This also means that absolute accuracy is not something we should be running behind. To elaborate, we could just predict 0 for every input and still get 90% + of the labels correct! But we will naturally miss out on the all people who will face serious delinquency and thus we need a smarter approach. We need to look at the false positives and false negatives. For our purpose, it is much more important to minimize false negatives rather than the false positives, as it is much better for a bank to not give loans to more people, some of whom would default, rather than give loans to a lot of people and many end up defaulting. Therefore we need to deal with this fact carefully and devise a systematic way to incorporate this fact.

B. Imbalance Correction? (Why not oversample and undersample)

As previously mentioned, we can notice that our dataset is highly imbalanced. We have way more negative entries (0) than positive ones (1) which is obvious because we do not expect many people to default (short for serious delinquency). Now it is believed that one should balance the dataset as it prevents bias and makes our models easier to train. The most common approach to this problem is to oversample or undersample so as to obtain a balanced dataset. Undersampling causes a huge leakage in information, so we will not touch upon them. For Oversampling, for example one of the most commonly used oversampling is called SMOTE, which creates synthetic samples along the lines joining the nearest neighbors in the feature space. We believe such approaches can be very problematic and thus we avoid them due to the following reasons:

1. Our purpose is not to just classify the people into Yes and No, but also to get a better understanding of the inherent probabilites. Sampling methods decalibrate the underlying probabilities which are very useful for us. One could try

to recalibrate them, but it is a very difficult process. Calibration means that e.g. a probability of 30% really means 30%. That is, if you bundle all 30% classifications of a well-calibrated classifiers together, you will find that 30% of them, in expectation, matched the ground truth. Not that classifiers are necessarily calibrated by default. However, by sampling to change the distribution of the target classes, you ensure that your model is not calibrated. Therefore methods like SMOTE leave probability interpretations disturbed and the usage of probability methods of Sklearn useless (Elor & Elor, 2022).

2. There is empirical evidence to suggest that methods to generate new data points for the minority class might actually generate points which do not belong to the minority class at all, leading to poor generalisation (Hassanat et al., 2022). As a result, training a classifier on these samples while pretending they represent minority may result in incorrect predictions when the model is used in the real world. Balancing seemed to improve the pure classification only in the case of weak learners. On the other hand, the strong learners trained on an imbalanced data dominated the former.

3. There is also empirical evidence to suggest that class rebalancing leads to overestimation of risk (Carriero et al., 2024). Infact, Tripod+AI, which is a 27-item checklist which aims to harmonise the landscape of prediction model studies has introduced the requirement of stating as to why the imbalance methods were used, and describe deeply how recalibration was done.

4. As a final argument, we challenge anyone to find a model which won any competition on platforms like kaggle using SMOTE or any other similar techniques :)

C. Outliers

We have tried to clean off the most common outlier patterns as described above. Still, we have a very few outliers in each category. We decided to continue with them and rather focus on finding robust models so that our model can tolerate uneven data better in the real world scenario.

4. MODELING

A. Model Shortlisting

After analyzing the nature of the problem, we realized that we should look at models concerning classification. Thus we started by shortlisting a few feasible models which are as follows: SVM, Logistic Regression, Random Forest, XGBoost, LightGBM Classifier, CatBoost Classifier.

We covered most of them in the class and therefore they don't require an introduction. However we can across two new types of boosting decision tree algorithms, CatBoost and LightGBM, which have the following features:

A.1. CatBoost

- **Ordered Boosting**: Unlike XGBoost and LightGBM that use traditional gradient boosting, CatBoost uses a variant called "Ordered Boosting". This reduces overfitting and makes the model more robust.

- **Oblivious Trees**: While XGBoost and LightGBM use regular decision trees, CatBoost uses a special type of decision tree called "oblivious tree", which makes the model more efficient and capable of handling different types of data.

- **Symmetry in structure**: The boosting algorithm utilizes a symmetric tree structure and optimized data layout for improved performance.

A.2. LightGBM

- **Leaf wise Growth**: This boosting algorithm focuses on leaf-wise tree growth rather than level-wise, which can lead to faster training times and reduced memory usage.

- **Use of GOSS**s: LightGBM supports categorical features using a technique called 'Gradient-based One-Side Sampling (GOSS). GOSS is a sampling method that accelerates the training process by focusing on instances with large gradients during the training of decision trees. It keeps the instances with small gradients unchanged while aggressively sampling instances with large gradients, thereby reducing the number of instances used for building trees without sacrificing accuracy.

B. Performance comparison

The table below reports the various average ROC_AUC scores obtained from 10-fold cross validation.

Model	ROC_AUC Score
SVM	0.680
Logistic Regression	0.761
Random Forest	0.835
XGBoost	0.850
LightGBM	0.857
CatBoost	0.859

With raw parameter setting, it was seen that SVM performed the worst as it was not able to process efficiently the whole dataset if not in batches. Logistic Regression performed relatively bad as well due to the unbalanced nature of data. It can be observed that Gradient Boosting frameworks performed better than others, making them more suitable for this specific problem; and within this family of models, XGBoost is dominated by the other two. So finally, we narrowed our choice on CatBoost Classifier and LightGBM, which are able to perform better on this dataset also due to their robustness against overfitting. (CatBoost seemed to handle false negatives better, so we mainly focused on that, but we tuned LightGBM as well)
We based our evaluation also on the confusion matrix and precision, with the objective of minimizing false negatives.

C. Optimizing the Best Performing Algorithm

Now that we have the algorithm that is performing the best in raw setting, we can go ahead and try to boost the performance further. The column '*NumberOfDependents*' was dropped because it wasn't affecting the performance of the model and we used feature importance methods to check this, which confirmed our hypothesis.
As mentioned before, instead of balancing data to find the best algorithm, we would rather fine tune the particular algorithm

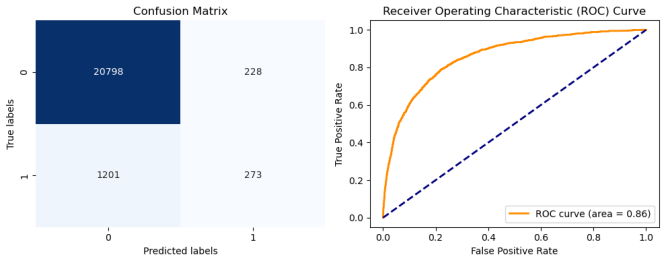

Fig. 5. CatBoostClassifier() with raw settings

on the imbalanced dataset. The two techniques we are going to talk about include hyperparameter tuning and cost sensitive learning.

Disclaimer: This is not to say that the models we did not shortlist will never outperform the shortlisted model, but due to computational constraints, we can not hyper tune every model to find the optimal. Therefore we continue with the best one.

C.1. Hyperparameter Tuning

All of the algorithms have some parameters which we can adjust so as to get the optimal solution. For our algorithm, we implemented a gridsearch, that is various combinations of different parameters and choosing the one which perform the best.

C.2. Cost sensitive learning

From the domain knowledge, we know that false negatives (predicting no default) are way more harmful for us than false positives (predicting default). Therefore it follows that it would make sense to penalise the model for classifying false negatives. There we introduce an additional cost (1:4) for every false negative classification done.

D. Final Comments regarding the Model

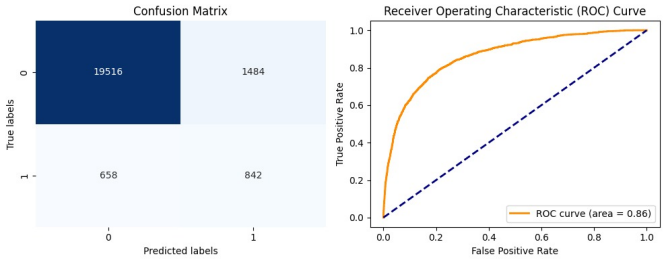

Fig. 6. CatBoostClassifier() after tuning

Tuning (exact parameters present in the code file) improved the performance of our algorithm slightly. The average 10 fold cross validation AUC score for this tuned model is 0.865. Moreover from the confusion matrix, it is clear that the number of false negatives reduced drastically. Moreover since we did not perform any balancing method, the probabilities this model generates is going to be calibrated and consistent.

One last thing we tried was to ensemble two strong learners, that is see how the average of catboost and lightgbm would perform. So we trained both the models separately and used the average of their individual probabilities to classify the data. This showed another small improvement where our AUC rose to around 0.86.8 on average, but it also had higher variance when we tried cross validating (by running test train split again). We suspect this might be due to the fact that both of these boosting methods have a different structure, so individually they might be a bit prone to some kind of mistakes, but averaging it all out cancels out any individual intrinsic issue. A more deeper look is required to validate this hypothesis, but it falls beyond the scope of the project.

5. CONCLUSION

In conclusion, our project aimed to construct a comprehensive machine learning model to predict the probability of serious delinquency in the coming two years using banking data. We began with exploratory data analysis (EDA), followed by data preprocessing and modeling.

The dataset presented several challenges, including imbalance, outliers, missing values, and anomalous entries. We addressed these issues through careful analysis, data cleaning, and preprocessing techniques such as imputation, outlier correction, and feature engineering.

For modeling, we shortlisted several classification algorithms and evaluated their performance using cross-validation. Based on the results, we selected CatBoost as the best performing algorithm due to its robustness against overfitting and ability to handle imbalanced data.

We further optimized the CatBoost model through hyperparameter tuning and cost-sensitive learning, resulting in improved performance and reduced false negatives. Additionally, we explored ensemble techniques by combining CatBoost and LightGBM models, which yielded a slight improvement in performance.

Overall, our approach demonstrates the importance of thorough data analysis, preprocessing, and model selection in building effective predictive models for credit risk assessment. While we achieved promising results, there remains potential for further exploration and refinement, including the incorporation of more advanced techniques and ensemble methods to enhance predictive performance.

REFERENCES

1. A. Carriero, K. Luijken, A. de Hond, *et al.*, (2024). ArXiv:2404.19494v1 [stat.ME].
2. A. B. Hassanat, A. S. Tarawneh, G. A. Altarawneh, and A. Almuhaimeed, "Stop oversampling for class imbalance learning: A critical review," (2022).
3. Y. Elor and H. Averbuch-Elor, "To smote, or not to smote?" (2022).
4. G. Lemaitre, "Euroscipy 2023 - get the best from your scikit-learn classifier," https://www.youtube.com/watch?v=6YnhoCfArQo&list=LL&index=10&t=1399s (2023).
5. J. E. de la Calle, "I declare myself the 1 enemy of over/undersampling, smote and adasyn, here's why how i circumvent it," https://juandelacalle.medium.com/i-declare-myself-the-1-enemy-of-over-undersampling-smote-and-adasyn-heres-wh (2023).

[1] [2] [3] [4] [5]