UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Master's Degree in Artificial Intelligence and Data Engineering

Data Mining and Machine Learning
Loan-Default Classifier

Author:
Nicolò Bacherotti

Project GitHub Repository:
https://github.com/NicoBache/DMMLproject

academic year: 2024/2025

Contents

1 Introduction **3**

2 Dataset Description **4**
- 2.1 Source and Size 4
- 2.2 Features Overview 4

3 Exploratory Data Analysis **6**
- 3.1 Target Distribution 6
- 3.2 Associations Among Categorical Features 6
- 3.3 Numerical Correlations 7

4 Data Preprocessing and Feature Engineering **8**
- 4.1 Preprocessing and Encoding strategies 8
  - 4.1.1 Handling Imbalanced Data 9
- 4.2 Feature Engineering 9
- 4.3 Log-Transform of Skewed Features 10

5 Modeling and Validation Workflow **11**
- 5.1 Initial Analysis with Cross-Validation 11
- 5.2 Hyperparameter Tuning with Grid Search 12
- 5.3 Hold-Out Validation 13
- 5.4 Results and Discussion 13
- 5.5 Explainability 15
  - 5.5.1 Feature Importance 15

6 Comparison with Literature **16**
- 6.1 Preprocessor adopted 16
- 6.2 Initial cross validation 17
- 6.3 Grid Search Setup 17
- 6.4 Models Evaluation 18
  - 6.4.1 Results 18
  - 6.4.2 Final Comparison 19

7 User Interface (GUI) **21**
- 7.1 Graphical User Interface (GUI) 21

Chapter 1

Introduction

Credit risk prediction is a central problem in the financial sector. When a borrower defaults on a loan, the financial institution suffers a direct economic loss and may also incur reputational damage. Traditional credit scoring systems, often based on linear models and rigid rules, are not always sufficient to capture the complex patterns in modern financial data. Machine learning techniques, on the other hand, can learn non-linear interactions and subtle patterns, offering the potential for more accurate predictions. This project aims to leverage machine learning models to improve default prediction performance, especially in the presence of class imbalance.

Chapter 2

Dataset Description

2.1 Source and Size

The dataset was obtained from Kaggle. It contains approximately 45,000 records of loan applicants, each corresponding to a single loan request. The data combines demographic attributes, financial information, and loan-related details, and is suitable for predictive modeling in credit risk assessment.

2.2 Features Overview

The dataset includes 14 input features, which can be grouped into three categories:

Personal Information

- **person_age**: Age of the applicant.
- **person_gender**: Gender of the applicant.
- **person_education**: Educational background (e.g., High School, Bachelor, Master).
- **person_income**: Annual income of the applicant.
- **person_emp_exp**: Years of employment experience.
- **person_home_ownership**: Type of home ownership.

Loan Details

- **loan_amnt**: Loan amount requested.
- **loan_intent**: Purpose of the loan.
- **loan_int_rate**: Interest rate applied to the loan.
- **loan_percent_income**: Ratio of loan amount to annual income.

Credit and Loan History

- **cb_person_cred_hist_length**: Length of the applicant's credit history (in years).
- **credit_score**: Credit score of the applicant.

- **previous_loan_defaults_on_file**: Indicator of whether the applicant has previous loan defaults (Yes/No).
  Although highly informative, this feature was removed from the analysis because of its high risk of being too predictive. In real credit scoring practice, such a variable can act almost as a direct proxy for the target: if an applicant has already defaulted in the past, the probability of defaulting again is much higher. For this reason, the feature was excluded from training and evaluation.

Target Variable

- **loan_status**: Target variable. Encoded as 0 if the loan was repaid successfully, and 1 if the applicant defaulted, meaning that the borrower did not repay the debt as agreed.
  The dataset is highly imbalanced, with the majority of cases belonging to class 0.

Chapter 3

Exploratory Data Analysis

3.1 Target Distribution

The first step of the analysis consisted of inspecting the target variable distribution. As shown in the figure below, the dataset is strongly imbalanced: non–defaulted loans (class 0) account for the vast majority of cases, while defaulted loans (class 1) represent only a minority.
This imbalance means that standard accuracy alone would not be a reliable evaluation metric; for this reason, measures such as F1-score, ROC AUC, and PR AUC are considered throughout the evaluation, and oversampling techniques like SMOTENC are adopted in the preprocessing stage.

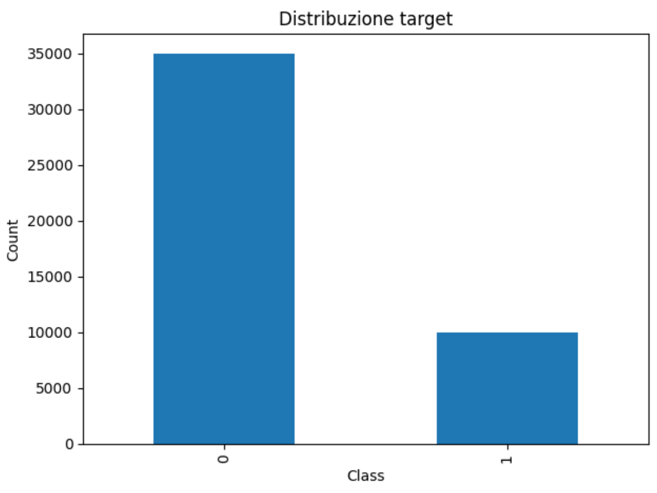


Figure 3.1: Target class distribution (non-default vs default).

3.2 Associations Among Categorical Features

To investigate dependencies among categorical variables, we computed pairwise associations using Cramér's V, a statistic that ranges from 0 (no association) to 1 (perfect association). The resulting heatmap shows generally weak associations: most variables appear independent from each other. This suggests that categorical features provide complementary information, reducing the risk of redundancy.

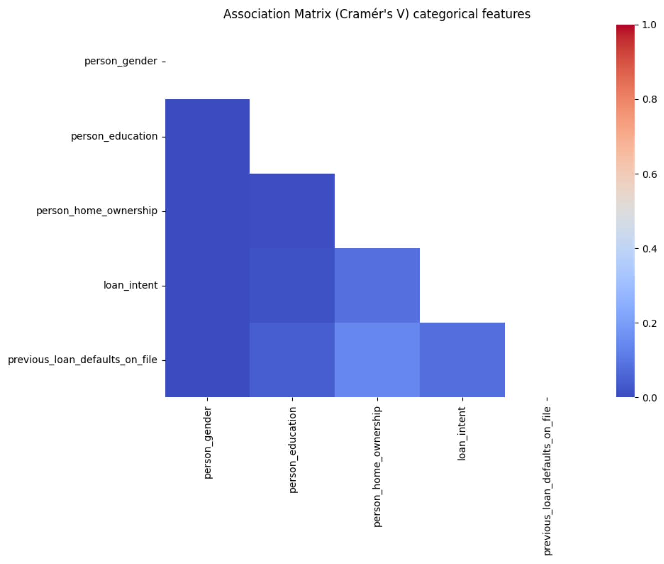


Figure 3.2: Cramer Matrix

3.3 Numerical Correlations

To conclude this first part of the analysis, the correlation matrix is presented. As expected in financial datasets, correlations with `loan_status` are modest in magnitude. Even if correlations are small, they highlight meaningful drivers of credit risk, later confirmed in the analysis.

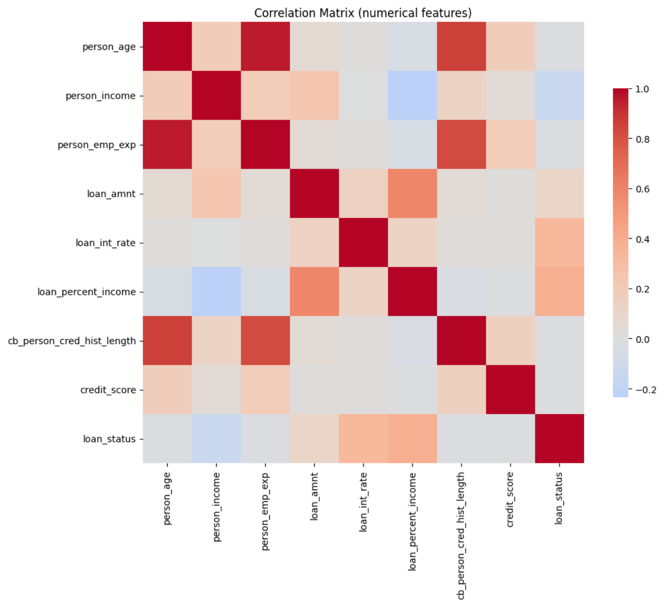


Figure 3.3: Correlation matrix

Chapter 4

Data Preprocessing and Feature Engineering

Although the dataset did not contain missing entries, null values, or NaN elements, a cleaning step was explicitly included in the preprocessing pipeline. For numerical variables, the chosen imputation strategy was the median, which is robust to skewed distributions. For categorical variables, the most frequent category was selected as the default replacement.
Even if not strictly necessary for this dataset, implementing these strategies ensures robustness and reproducibility: the pipeline can handle unexpected missing values in future data and maintains consistent preprocessing across different scenarios.

4.1 Preprocessing and Encoding strategies

Two dedicated preprocessors were designed to handle the dataset before feeding it into the learning algorithms, one for every phase of this Project. The preprocessors are implemented in two separate python files and they work inside the pipeline of the algorithms.
Both preprocessors follow a common structure: numerical features are imputed using the median to handle potential missing values and then standardized, while categorical features are imputed using the most frequent category to ensure consistency. Numerical features were standardized using `StandardScaler`. Standardization ensures that features with different scales contribute equally to the learning process and prevents models like logistic regression from being dominated by high-magnitude features. The main difference lies in the encoding strategies:

- **Preprocessor with Ordinal Encoding**: categorical variables are transformed into integer codes. This approach preserves compact representations and is fully compatible with SMOTENC, which was the oversampling method adopted in the first phase of the project.
- **Preprocessor with One-Hot Encoding**: categorical variables are expanded into binary indicator vectors using One Hot Encoding. This strategy was implemented to have another method to compare the results and, mainly, to allow a comparison with existing literature.

The first preprocessor is used in the first part of the analysis, the main workflow. The second one is created for comparison with the state of the art, described in the last chapters of the documentation.

4.1.1 Handling Imbalanced Data

To address the strong class imbalance in the dataset, a resampling strategy was required.
In this first section of the project, SMOTENC (Synthetic Minority Over-sampling Technique for Nominal and Continuous features) was adopted. Unlike SMOTE, which assumes all features are continuous, SMOTENC is explicitly designed for mixed datasets, allowing the user to specify which variables are categorical, then treat them in the appropriate way.
To make SMOTENC work properly, categorical features were encoded with Ordinal Encoder, which was introduced because it provides integer labels that SMOTENC can correctly interpret as categorical, without implying any semantic ordering among categories. To prevent data leakage, SMOTENC was embedded within the training pipeline and executed separately inside each fold of the cross-validation. This guarantees that resampling is applied only to the training partitions, leaving validation data unaffected and providing an unbiased estimate of performance.

4.2 Feature Engineering

To enrich the dataset and capture relationships not directly observable from the raw variables, five engineered features were created and included throughout all the analysis. Each of them was designed to reflect specific aspects of borrowers' risk profiles and financial capacity, providing additional predictive power to the models:

- **person_age_bin**: Applicant age grouped into intervals (<25, 25–35, 35–50, 50+). Younger applicants may have unstable incomes, while older applicants may have more stable financial profiles. Binning highlights these differences more clearly than raw age.
- **income_to_loan**: Ratio between annual income and requested loan amount. This serves as a proxy for financial sustainability, indicating how many times the borrower's income covers the debt. A higher ratio suggests that the loan is relatively affordable, while a low ratio points to higher repayment difficulty and thus higher risk of default.
- **emp_exp_x_age**: Employment experience divided by age. This ratio measures the consistency of a borrower's career relative to their age. A high value indicates early employment history, which is a sign of reliability, while a low value may reveal employment instability, potentially increasing credit risk.
- **loan_over_score**: Loan amount divided by credit score. Borrowers requesting large loans with low credit scores are flagged as higher risk, while

smaller loans relative to strong credit scores suggest a more balanced financial profile.

- **loan_int_rate_bin**: Interest rate grouped into intervals (<10%, 10–15%, 15–20%, >20%). Since higher interest rates are typically charged to riskier clients, binning allows the model to directly capture risk tiers.

4.3 Log-Transform of Skewed Features

To reduce the impact of skewed distributions, a log-transformation procedure was applied to skewed numerical variables.
The skewness of each numeric feature was first computed on the training set, and variables with an absolute skewness greater than a fixed threshold were identified as candidates for transformation.
For these features, a shift was added when necessary to ensure strictly positive values, and the transformation was applied, replacing the original columns. The same transformation and shifts were then consistently applied to the test set to avoid data leakage.

Chapter 5

Modeling and Validation Workflow

5.1 Initial Analysis with Cross-Validation

The first experimental step consisted of a systematic cross-validation benchmark to evaluate different classifiers. A 5-fold stratified cross-validation strategy was adopted to preserve the class distribution across folds and reduce the variance of the estimates.

The models tested were:

- Logistic Regression.
- Random Forest.
- XGBoost.
- LightGBM.
- CatBoost.

Two pipelines were defined: the first included only the preprocessing stage and the classifier, while the second integrated `SMOTENC` within the pipeline to address class imbalance.

Each model was trained and evaluated under both configurations; furthermore, the macro-averaged F1-score was chosen as the primary evaluation metric because it balances precision and recall across both classes. Results were averaged across folds to ensure robustness.

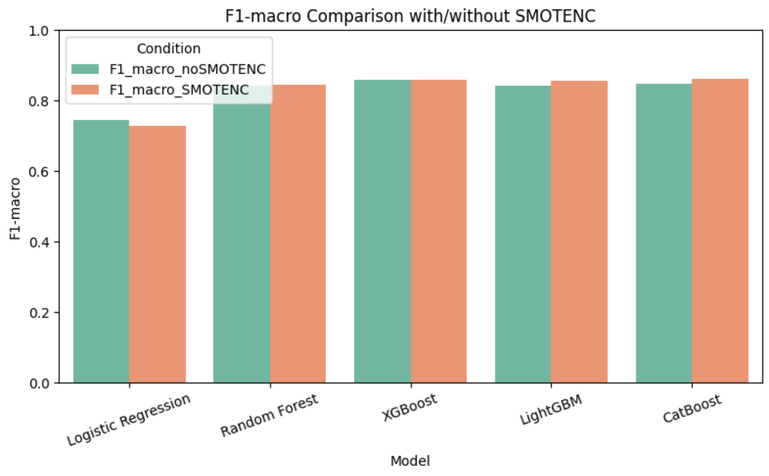


Figure 5.1: Comparison of f1_macro for classifiers.

Overall, the results show that models such as Random Forest, XGBoost, LightGBM, and CatBoost achieve high F1_macro values, above 0.80 in both configurations, even though SMOTENC leads to marginal but positive improvements. Logistic Regression, in contrast, exhibits lower performance, with F1-macro values around 0.75 in both cases.
The differences between the two conditions are small, which indicates that these algorithms are inherently robust to imbalance.
Even if SMOTENC does not lead to dramatic improvements, the results show that it contributes to slightly increasing performance. For this reason, SMOTENC was retained in the subsequent analyses as part of the main pipeline.
XGBoost and CatBoost emerged as the best-performing models (with SMOTENC), with a very small difference in performance (XGBoost=0.859, CatBoost=0.861).

5.2 Hyperparameter Tuning with Grid Search

After the initial comparison, the best-performing models were further optimized using Grid Search with 5-fold stratified cross-validation.
The search aimed to identify the optimal hyperparameters for each classifier, with the macro-averaged F1-score used as the selection criterion since it balances precision and recall across both classes in the imbalanced dataset.
Tables below report the parameters considered.

Table 5.1: Grid Search parameters for XGBoost.

Parameter	**Values tested**	$\lvert\cdot\rvert$
`n_estimators`	{200, 400, 600}	3
`max_depth`	{3, 5, 7, 10}	4
`learning_rate`	{0.01, 0.05, 0.1}	3
`subsample`	{0.6, 0.8, 1.0}	3
Total combinations		**108**

Table 5.2: Grid Search parameters for CatBoost.

Parameter	**Values tested**	$\lvert\cdot\rvert$
`iterations`	{200, 400, 600}	3
`depth`	{4, 6, 8, 10}	4
`learning_rate`	{0.01, 0.05, 0.1}	3
`l2_leaf_reg`	{1, 3, 5, 7}	4
Total combinations		**144**

The best hyperparameter configurations identified for the two boosting models were the following.
For `XGBoost`:learning rate of 0.1, a maximum depth of 5, 600 estimators, and a subsample ratio of 0.8.

For `CatBoost`, the best configuration was obtained with a depth of 6, 600 iterations, an $L2$ regularization parameter of 1, and a learning rate of 0.1.

5.3 Hold-Out Validation

After hyperparameter tuning, the best configurations of XGBoost and CatBoost were re-trained and validated on the independent test set. The evaluation pipeline, the one chosen in the cross-validation step, included three main components:

- **Preprocessor**: applied imputation strategies, scaling of numerical features, and encoding of categorical variables.
- **SMOTENC**: synthetic oversampling of the minority class. This step was crucial to mitigate imbalance directly during training, generating realistic synthetic samples that preserved categorical structure.
- **Classifier**: the final model trained with the optimized hyperparameters obtained from the Grid Search stage.

5.4 Results and Discussion

The final evaluation on the independent test set provided a comprehensive view of the generalization ability of the optimized models.
Figure 5.2 reports both ROC curves and a side-by-side comparison of evaluation metrics. Both models achieved strong performance, with ROC AUC values above 0.93.
The barplot comparison shows that XGBoost, which before was slightly less performant than CatBoost, now has F1_macro=0.869 against CatBoost with 0.863, indicating a marginal but consistent improvement after hyperparameter tuning and electing XGBoost as the best-performing model overall.
Recall values remained more modest for both models, reflecting the difficulty of identifying minority-class (default) cases.

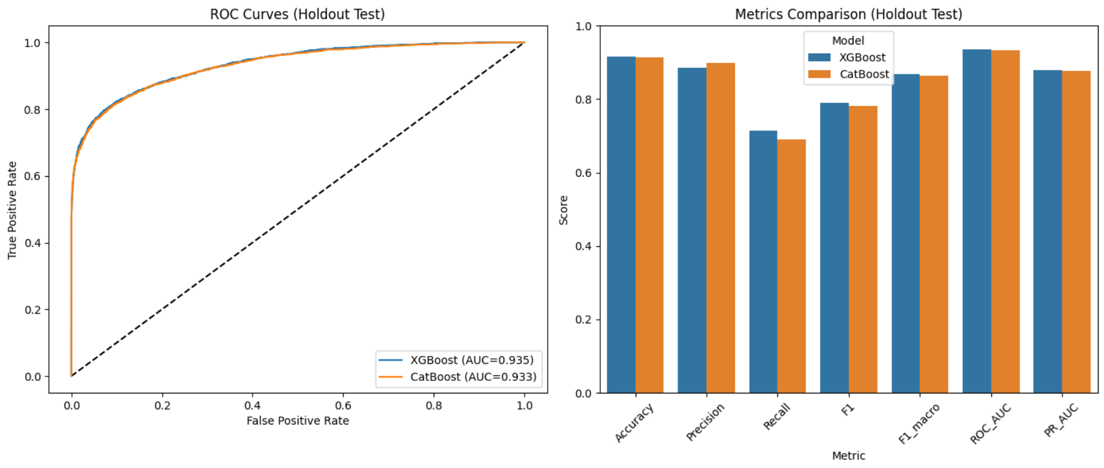
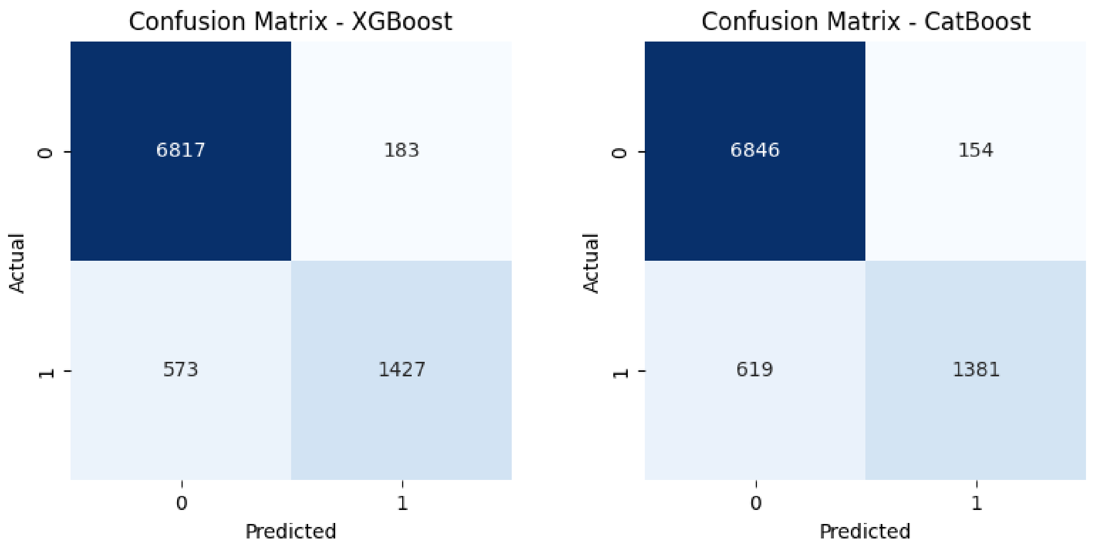


Figure 5.2: ROC curves and metric comparison on the test set.

Confusion matrices highlight that XGBoost achieved the best balance, detecting 1,427 defaults with fewer false negatives. CatBoost performed similarly, though with slightly more missed cases.



Overall, the results confirm the superiority of XGBoost, emerging as the best model in terms of balanced performance.

5.5 Explainability

5.5.1 Feature Importance

The feature importance analysis reveals a set of predictors that appear among the top-ranked features in both CatBoost and XGBoost, confirming their crucial role in default risk modeling.
In particular, `loan_percent_income`, `income_to_loan` are highly ranked by both models, underlining the importance of affordability measures and credit history in assessing repayment ability.
`Loan_int_rate` also emerges as a relevant predictor, reflecting the lender's risk-based pricing strategy.
Beyond these commonalities, each model also emphasizes different aspects of borrower information. CatBoost ranks `person_age` and `person_emp_exp` among its most important features, suggesting that demographic and career stability indicators play a stronger role in its tree splits. XGBoost, in contrast, highlights `person_home_ownership` as the single most influential feature, indicating that categorical distinctions between owning, renting, or holding a mortgage provide highly discriminative patterns.
Overall, the convergence on affordability and credit history measures validates the robustness of the findings, while the divergences show how CatBoost tends to capture demographic stability, whereas XGBoost leverages categorical splits such as housing status and loan purpose to improve discrimination between defaulting and non-defaulting borrowers.

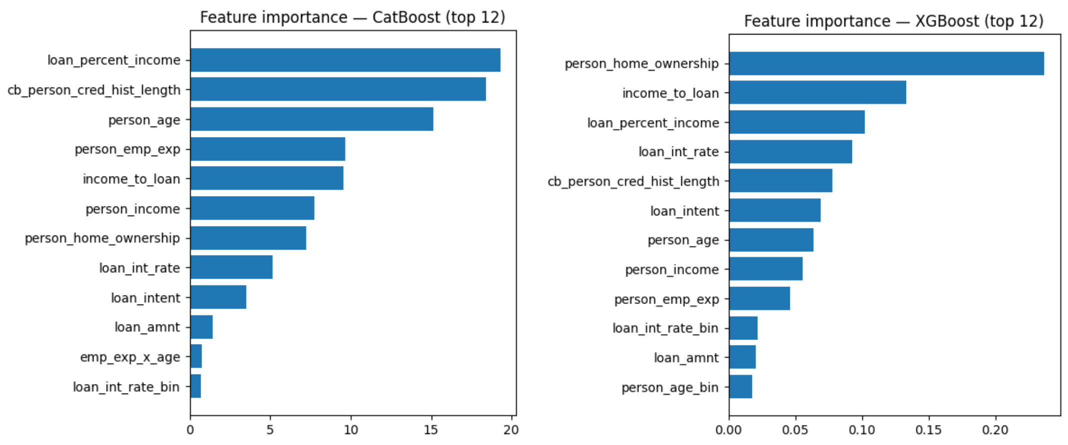


Figure 5.3: Feature importance for CatBoost and XGBoost.

Chapter 6

Comparison with Literature

Both reference studies explored machine learning models for credit risk prediction, but adopted different preprocessing and resampling strategies compared to the present work.
Fekadu et al. [1] combined integer and One-Hot Encoding for categorical variables and addressed imbalance with oversampling techniques such as KMeans-SMOTE and CTGAN. Their experiments covered several classifiers, with XGBoost achieving the best performance, with F1-scores of 0.92 and 0.91 for the majority and minority classes, respectively. Their oversampling was applied only to the training set, but the procedure seemed to not be embedded in a cross-validation pipeline, which can lead to less robust performance estimation.
Monje et al. [3] relied on One-Hot Encoding and label encoding for categorical variables and tested multiple resampling strategies including SMOTE, undersampling, and oversampling. They benchmarked various models, again highlighting XGBoost as the most effective with an AUC of 0.731 and an accuracy of 0.795. Despite these attempts, the authors reported that resampling techniques provided limited improvements; furthermore, no approach specifically designed for mixed data was considered.
In contrast, the present work used **SMOTENC**. This choice was motivated both by the intention to explore a different resampling strategy and by the natural suitability of SMOTENC for handling categorical features, an aspect not addressed in the reference studies.
By specifying categorical indices and interpolating only numerical variables, SMOTENC avoids the creation of implausible synthetic samples that may arise when applying One-Hot Encoding with standard SMOTE. In addition, oversampling was embedded within each fold of the cross-validation pipeline, preventing information leakage from the validation data and providing a more reliable estimate of model performance. The adoption of SMOTENC, therefore, allowed us also to assess whether it could deliver an advantage in the specific context of credit risk prediction.
Since the datasets considered in the reference studies differ from the one used in this work, a fully direct comparison is not possible. For this reason, a **second experimental procedure** was implemented following a setup closer to that of the first stage of the project, by combining One-Hot Encoding for categorical variables with standard SMOTE for class imbalance.

6.1 Preprocessor adopted

The preprocessor used in this second phase, based on One-Hot Encoding, handles numerical and categorical variables separately, like the one used in the previous phase. For numerical variables, the flow applies median imputation for missing values, followed by standardization through StandardScaler. For categorical variables, missing values are imputed with the most frequent category, after which the variables are converted into binary vectors using One-Hot Encoding.

6.2 Initial cross validation

An initial 5-fold cross-validation was conducted , the models tested were `XGBoost` and `CatBoost`, the 2 best of the previous phase, complemented by `Random Forest`, which was included to ensure comparability with both previous studies, since it was also used in both of them.
The results confirm that `XGBoost` (F1-macro = 0.863) and `CatBoost` (F1-macro = 0.864) achieved very close performances across all metrics, outperforming `Random Forest` (F1-macro = 0.850). For this reason, the subsequent analyses are focused on XGBoost and CatBoost, leaving out Random Forest.

6.3 Grid Search Setup

Similarly as before, hyperparameter tuning was performed using stratified 5-fold cross-validation with F1-macro as the selection criterion.
Tables below report the parameters used for both XGBoost and CatBoost in the grid search.

Table 6.1: Grid Search parameters for XGBoost (OHE + SMOTE).

Parameter	**Values tested**	$\lvert\cdot\rvert$
`n_estimators`	{300, 600}	2
`max_depth`	{4, 6}	2
`learning_rate`	{0.05, 0.1}	2
`subsample`	{0.8, 1.0}	2
Total combinations		**16**

Table 6.2: Grid Search parameters for CatBoost (OHE + SMOTE).

Parameter	**Values tested**	$\lvert\cdot\rvert$
`iterations`	{400, 600}	2
`depth`	{6, 8}	2
`learning_rate`	{0.05, 0.1}	2
`l2_leaf_reg`	{3, 5}	2
Total combinations		**16**

The best hyperparameter configurations identified for the two boosting models were the following.

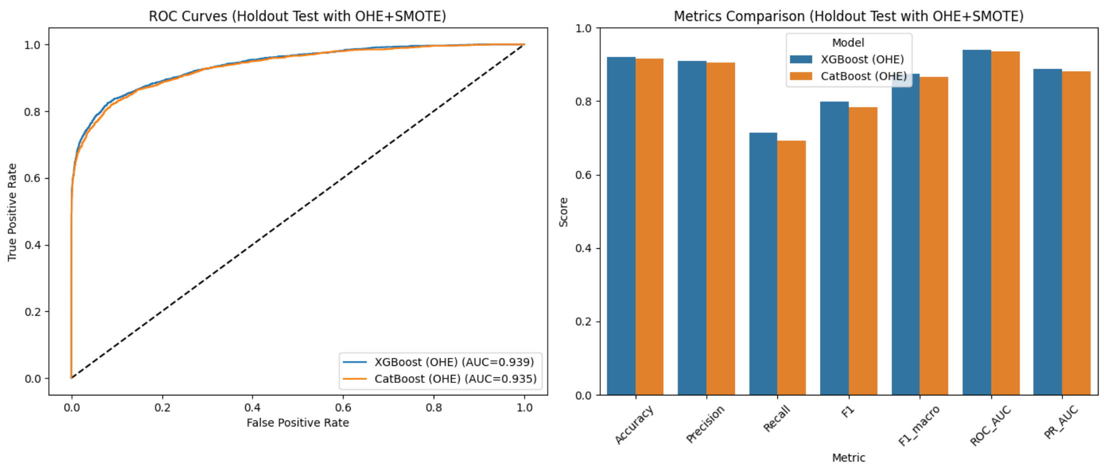
For `XGBoost`:learning rate of 0.1, a maximum depth of 6, 600 estimators, and a subsample ratio of 0.8.
For `CatBoost`, the best configuration was obtained with a depth of 6, 600 iterations, an $L2$ regularization parameter of 3, and a learning rate of 0.1.
In this comparative section, the hyperparameter grids were not exhaustive as before. The goal was to reproduce a representative and computationally feasible setup consistent with the literature, in order to enable a fair comparison between the two modeling strategies.

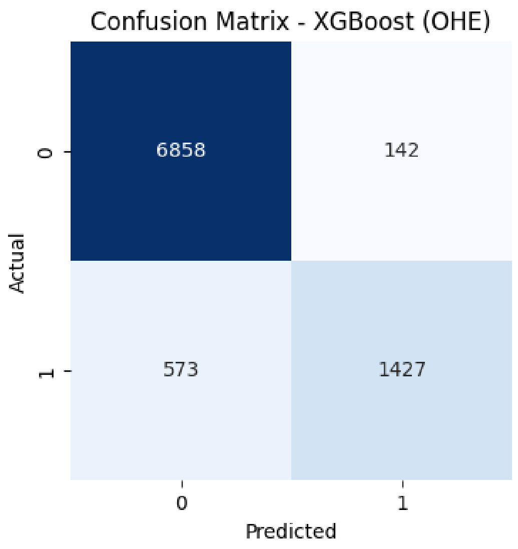
6.4 Models Evaluation

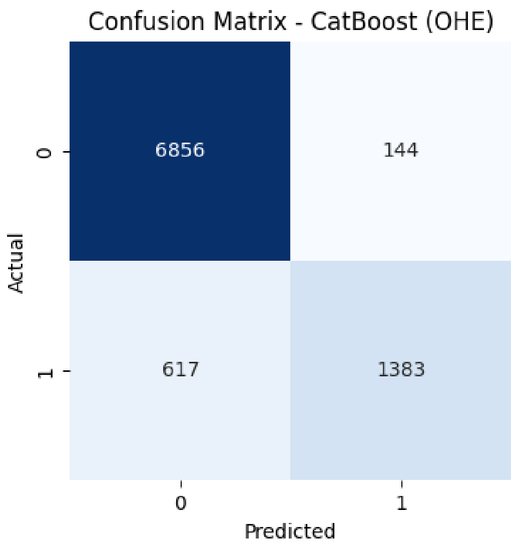
Following the same procedure adopted in the main analysis, the models optimized through Grid Search were re-trained and evaluated on the independent test set. Each pipeline included preprocessing with One-Hot Encoding, oversampling with SMOTE, and the classifier with tuned parameters.

6.4.1 Results

The final validation confirms that both the models exhibit consistent behavior also under the OHE+SMOTE configuration. As illustrated by the ROC curves in Figure 6.4.1, XGBoost achieved the best results (F1_macro=0.875 against 0.865). The barplot of metrics further supports this result: XGBoost reached the best trade-off across precision, recall, and F1_score.
The confusion matrices confirm the results just discussed, with XGBoost achieving the best balance between errors, with fewer false negatives and false positives.







6.4.2 Final Comparison

Figure below summarizes the results obtained by XGBoost under the two experimental settings.
The metrics show very close performances across the two configurations, with only marginal differences. Accuracy, ROC AUC, and PR AUC are practically identical, while small variations appear in precision and F1-macro, where the setting with OHE and SMOTE slightly outperforms SMOTENC. Overall, the results confirm that both configurations are valid and competitive, but the second one(with Smote) appears to be slightly better (F1_macroSMTNC=0.869, F1_macroSMOTE=0.875).

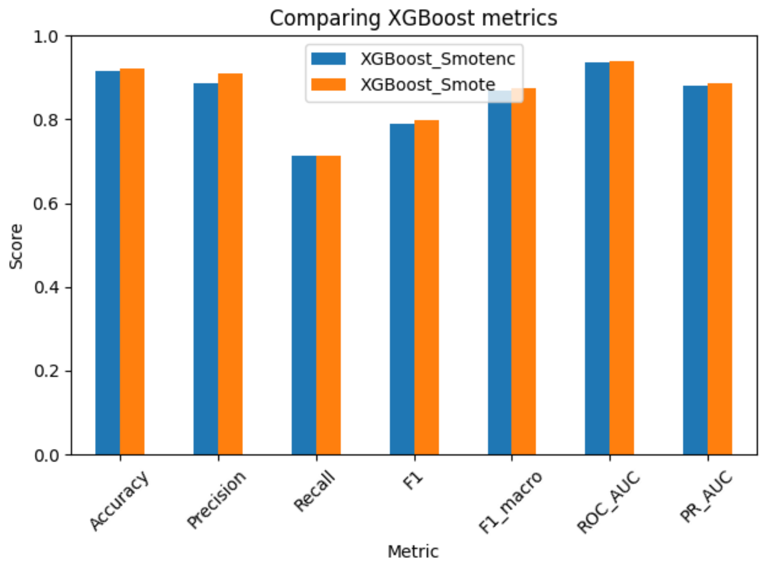


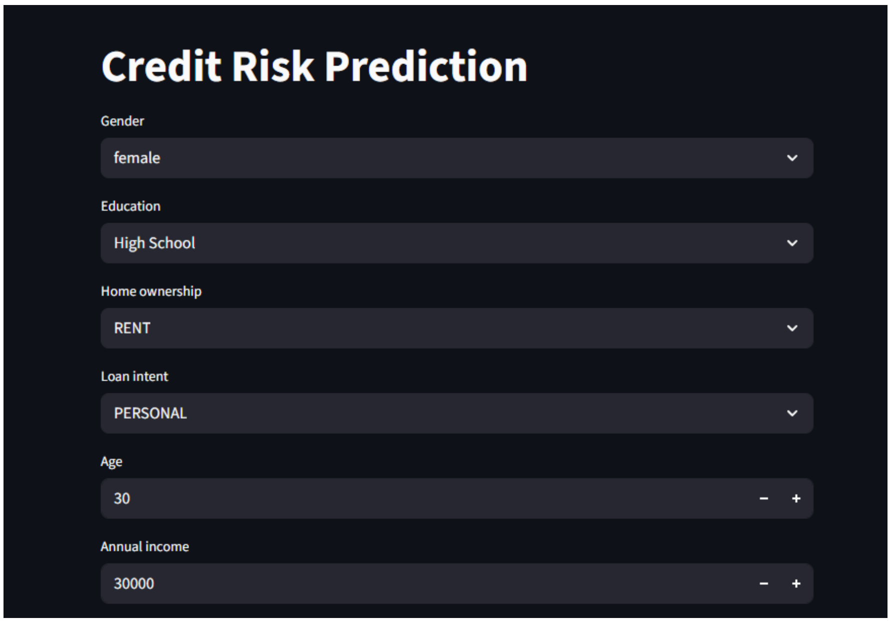
Figure 6.1: Comparison of XGBoost metrics between two configurations.

Chapter 7

User Interface (GUI)

7.1 Graphical User Interface (GUI)

To make the results of the project more accessible, a simple graphical user interface was developed using `Streamlit`.
The interface allows users to manually input borrower characteristics and automatically performs feature engineering steps consistent with the preprocessing pipeline. Once the inputs are provided, the trained pipeline is invoked to generate both a class prediction (default vs. non-default) and the probability of default, making the system usable even by non-technical users.



Credit Risk Prediction

Gender: female

Education: High School

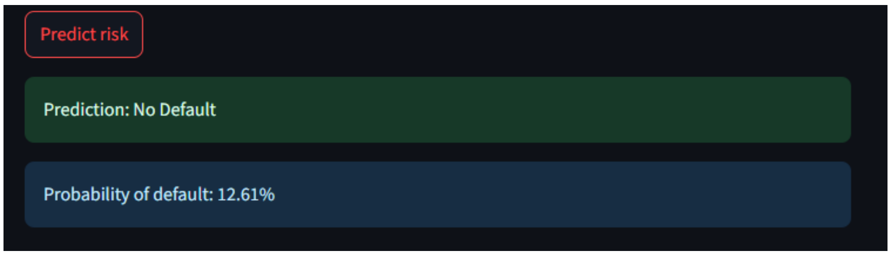
Home ownership: RENT

Loan intent: PERSONAL

Age: 30

Annual income: 30000

Figure 7.1: Partial Input form of the GUI.



Prediction: No Default

Probability of default: 12.61%

Figure 7.2: Prediction output of the GUI.

Bibliography

[1] Rufael Fekadu, Anteneh Getachew, Yishak Tadele, Nuredin Ali, and Israel Goytom. Machine learning models evaluation and feature importance analysis on npl dataset, 2022. URL: https://arxiv.org/abs/2209.09638, arXiv:2209.09638.

[2] Uday Malviya. Bank loan data, 2021. Kaggle dataset. URL: https://www.kaggle.com/datasets/udaymalviya/bank-loan-data?select=loan_data.csv.

[3] Leticia Monje, Ramón Alberto Carrasco, and Manuel Sánchez-Montañés. Machine learning xai for early loan default prediction. *Computational Economics*, June 2025. doi:10.1007/s10614-025-10962-9.