Credit Risk Prediction Reassessed: A Critical Evaluation and Enhanced Machine Learning Framework

University College London

COMP0050 – Machine Learning in Finance
Authors: Group 3

Tomás Barbosa Romeiro (tomas.romeiro.24@ucl.ac.uk)
Elliot Ward (elliot.ward.24@ucl.ac.uk)
Suyu He (suyu.he.24@ucl.ac.uk)
Yuxing Tao (yuxing.tao.24@ucl.ac.uk)
Yixin Jiang (yixin.jiang.20@ucl.ac.uk)
Daisy Yang (daisy.yang.21@ucl.ac.uk)

April 2025

1 Introduction and Methodology

Credit risk prediction is essential in modern finance, enabling institutions to manage potential loan defaults and maintain overall portfolio health. Many pioneering studies in this area have leveraged advanced machine learning techniques to achieve remarkable predictive accuracy. However, these studies often exhibit methodological ambiguities, such as issues with normalisation, sampling, and feature selection, that can compromise the reliability and generalisability of their findings.

Our project initially aimed to replicate and extend the methodology from the recent article "Credit Risk Prediction Using Machine Learning and Deep Learning: A Study on Credit Card Customers" [1] on credit risk prediction. Early in our work, we encountered significant challenges due to unclear data handling practices in the original study, particularly concerning normalisation and sampling strategies.

In addition to refining normalisation and sampling procedures, we incorporated categorical variables in one-hot encoded form, a detail overlooked in the original study. After evaluating several models, including Logistic Regression, Multi-Layer Perceptrons (MLPs), Long-Short Term Memory (LSTM), Support Vector Machine (SVM), Random Forests (RF) and XGBoost (the best model in the original article). XGBoost, RF and SVM were the top performers in our analysis. XGBoost, in particular, performed remarkably well by comparison, which led us to focus the analysis in the later part of the report on this model's output.

This report documents our journey from replicating an existing, albeit imperfect, methodology to developing an enhanced framework that leverages the original articles and improves and extends the methodology. The publicly available datasets allow for our results to be benchmarked against future studies and our critical enhancements underscore the importance of methodological rigour for replication. Enhanced credit risk prediction models offer practical benefits for financial institutions, particularly banks and credit card companies through superior loan default risk assessment, reduced financial risk, and more efficient resource allocation.

2 Data Processing

2.1 Data and Initial Exploration

Our study began with two primary datasets from Kaggle (Appendix B): the *Application Record* containing the raw application data for credit customers, including personal details and application-specific information, and the *Credit Record* tracking the credit performance of these customers over time, including monthly balances and status codes indicating payment delays.

The datasets comprise 438,557 and 1,048,575 records, respectively, and are linked via an "ID" field. An initial check was to compute the frequency of each ID and flagged those that appeared more than once to identify

duplicate IDs. In addition, we filtered the credit record to extract entries corresponding to these duplicate IDs for further inspection. Although these duplicate IDs weren't present in the Credit Record dataset, we decided to exclude them for precaution.

Next, we investigated the credit record data further by determining the "opening month" for each customer. We grouped the data by customer ID and extracted the minimum value of the "MONTHS_BALANCE" variable as the account's opening month. Using this, we computed a new variable, "MONTHS_SINCE_OPENED", which indicates how long each customer has been active. This variable became key in our later analysis, serving as the basis for time-dependent visualisations. Finally, we merged the two dataset on their common key ("ID") using an inner join, which keeps only records where a match is found between the two datasets. Our joined data set totals 36,457 rows.

2.2 Exploratory Visualisation and Comparison

Our EDA mimicked the approach described in the original article.

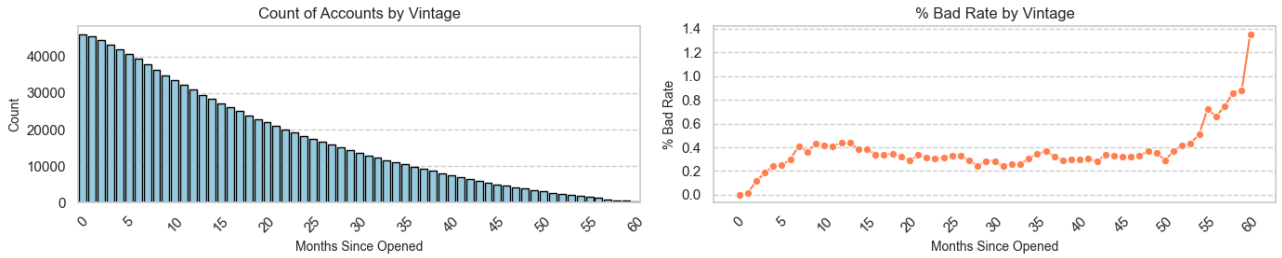


Figure 1: Number of accounts and bad rate % by Vintage

We start by generating the bar plot in figure 1, which displays the frequency of customer records across different values of "MONTHS_SINCE_OPENED" (typically known in lending as "Vintage"). We plot the percentage of customers with delinquent loans by "Vintage" by calculating a new binary variable ("bad_flag"), based on whether the customer's status exceeded a threshold. These plots closely resemble those in the article, providing a clear picture of how the credit portfolio evolves over time.

Figure 2: Good / Bad Customer Status %

Using our "bad_flag" variable we restrict the analysis period to the first 12 months of data, where bad rates are more stable, and treat this flag as out target prediction variable. Our target distribution plot (figure 2), displaying the count and percentage of good versus bad customers, revealed proportions that were in line with the overall figures reported in the article.

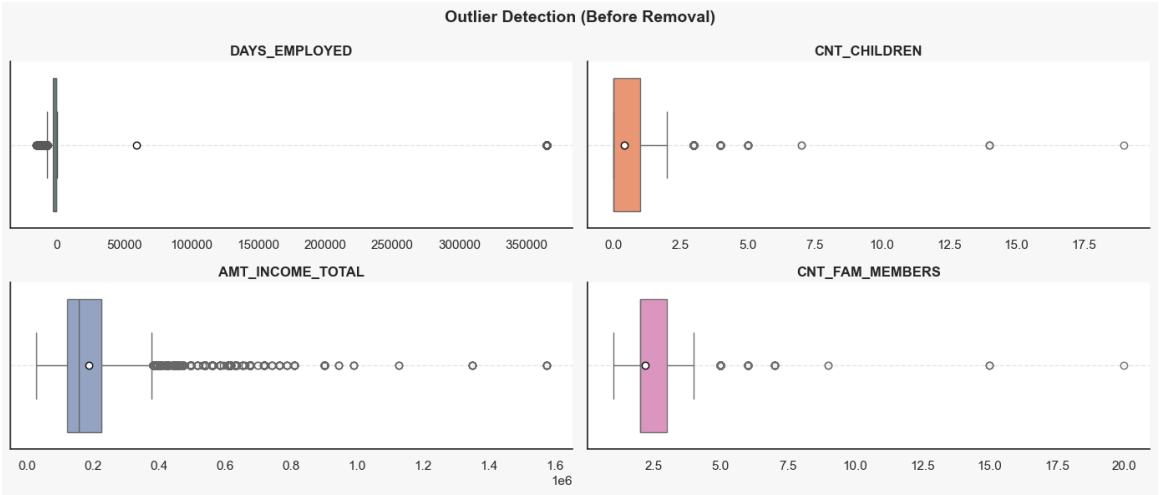


Figure 3: Outlier detection: before removal

In our next step, we focused on identifying and removing outliers. The box plots are used to visualise the distribution of key non-binary variables ("DAYS_EMPLOYED", "CNT_CHILDREN", "AMT_INCOME_TOTAL", and "CNT_FAM_MEMBERS"), showing a number of extreme values (figure 3). The application of an Interquartile Range (IQR) based removal (with a multiplier of 1.5) produced distributions (Figure 4) that were largely comparable to those reported by the authors, though we noted that slight differences started emerging. For example, we completely lose outliers above 2 in count of children and 4 in count of family members, whereas these stretch out to 4 and 6 respectively in the authors' outputs. It is worth noting that changing the factor applied to the IQR from 1.5 to 3.0 aligned our results more closely to the original article, but we decided to keep the factor at 1.5 for consistency. Our dataset's size was reduced from 36,457 rows to 26,663 rows after outlier removal, a change that appears reasonable in the context of the article's description.

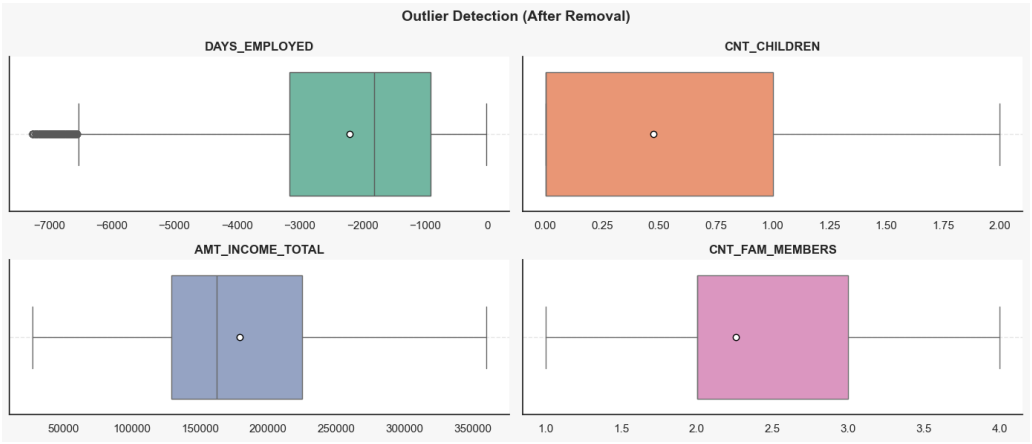


Figure 4: Outlier detection: after removal

To enable further analysis and comparison between our treatment of the dataset and the original article, we transformed "DAYS_BIRTH" into age (in months) and "DAYS_EMPLOYED" into employment duration (in months), using a conversion factor of 30.4 days per month (which might be contributing in part to the discrepancies between our results and the article's as this was not specified by the authors). Figure 5 shows that, on the surface, there are no major differences between the two types of customers when broken down by age, work experience or income. It is important to point out that the work experience plot is one of the results where we observe a larger gap when compared to the reference article, in terms of the range of the data. The article's output ranges from 0 to approximately 500 months for good customers, whereas our results only stretch to near 250 months. This likely stems from the treatment of outliers mentioned earlier in this report.

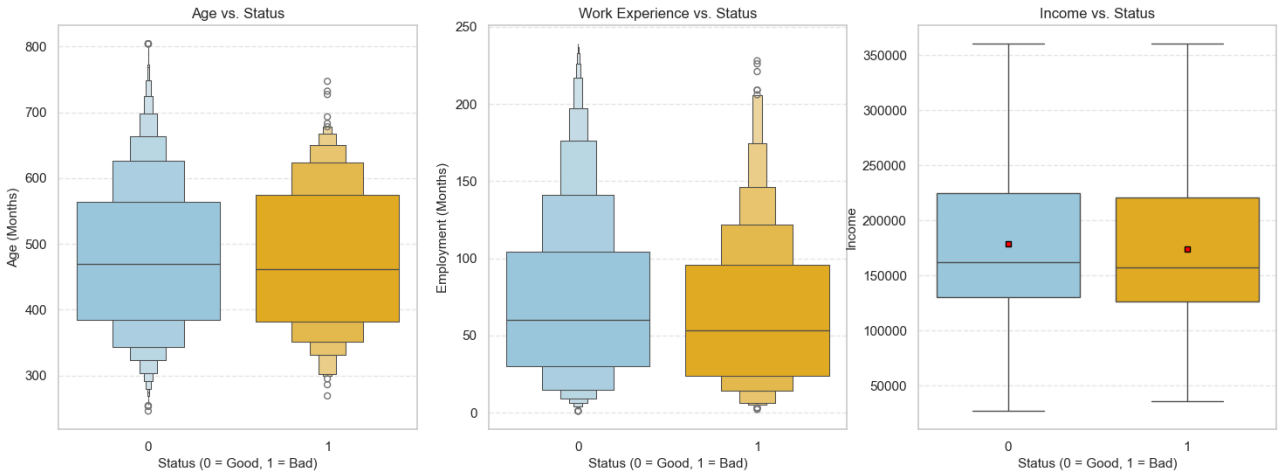


Figure 5: Age, Work Experience and Income vs Customer Status

In figure 6 below, which describes whether each type of customer owns property or not, we again arrive at outputs that are just slightly off.

In regards to what we can conclude by this initial look at some of the variables in the dataset, we see that a broad pattern of similarity between good and bad customers, the large differences being observed in age (where bad customers tend to be not as old or young in the extremes) and in property ownership (where good customers tend to own more property, around an extra 6.4 percentage points).

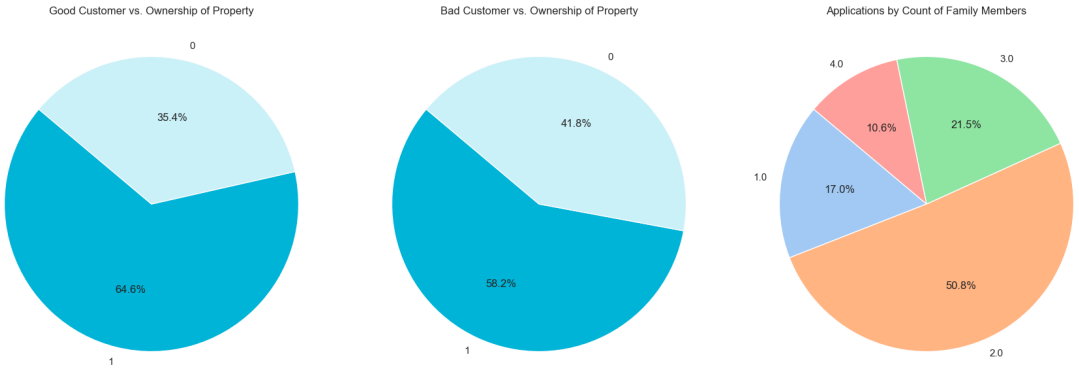


Figure 6: Ownership of Property by Customer Status and % of Applications by Count of Family Members

Given the broad similarity between the two types of customers in these variables we will, later in the report, look at the categorical variables provided with the dataset. It is not clear if they were used in the original article as they are not mentioned in their analysis.

Finally, we complete the analysis and comparisons with the original article by looking at the correlation matrix of features in figure 8 (see Appendix C), where we can see additional confirmation that we have closely followed the same process so far, given that correlations are close if not the same across most variables.

2.3 Data Pre-processing

In this section, we address several key issues: we applied proper variable normalisation, ensured SMOTE was only applied to the training set, and, importantly, we incorporated categorical variables in one-hot encoded form, a detail not covered in the original study. Following this, our model implementation section will detail the mathematical foundations underlying each model. In the results section, we then present our findings and discuss how they compare with those reported in the article. Details regarding where to access the dataset and software used in this report can be found in Appendix A.

Our pre-processing approach began by isolating the target variable from the dataset and eliminating redundant features (related with contact details) that did not contribute meaningful information. We retained the key variables present in our correlation matrix. Additionally, we enriched the dataset by incorporating

several categorical variables, such as income type, education type, family status, housing status and occupation type, which were transformed into one-hot encoded representations. In doing so, we deliberately excluded one category from each set to avoid redundancy and prevent multicollinearity.

Next, we split the dataset into training and test sets in a 70:30 ratio using stratified sampling to preserve the original class distribution. This step was critical to ensuring that subsequent processing and model evaluation would not suffer from data leakage.

To prepare the numerical features for modelling, we normalised each feature using Min-Max scaling. Importantly, the scalers were fitted solely on the training data, and the same scaling parameters were then applied to transform the test set. This guarantees that the test set is scaled consistently without exposing it during the fitting process, which would lead to data leakage issues.

Given the extreme class imbalance (98.8% good vs. 1.2% bad customers), our main analysis utilises SMOTE [5] to synthetically balance the training set, resulting in 18,438 instances per class, while leaving the test set untouched for realistic evaluation. As mentioned earlier this is the method employed in the original article, but unlike the authors we restrict it to the training set. All our models were applied on the oversampled dataset.

In addition to purely using SMOTE we further explored two hybrid approaches combining it with undersampling methods: Tomek Links (T-link) and Edited Nearest Neighbors (ENN)[6], again applied only to the training data. These methods help refine class boundaries by both oversampling the minority class and removing overlapping majority class examples. Due to space constraints we restricted this alternative sampling approach to the model with best performance (XGBoost) and performed hyper-parameter tuning for each sampling method as well.

3 Model Descriptions

In the following section we will provide the mathematical foundations of the models used in this report and describe our choice of parameters. Although we attempted and will display results for several models in later sections, we will restrict this section to the 3 most successful models in our experiment for brevity.

3.1 XGBoost

The XGBoost model [2] is designed by constructing an ensemble of regression trees. The model aggregates the outputs of K regression trees as follows:

$$\hat{y}(x) = \sum_{k=1}^{K} f_k(x), \quad f_k \in \mathcal{F}, \tag{1}$$

where $\mathcal{F}$ is the space of regression trees and $f_k(x)$ represents the prediction from the kth tree.

For binary classification, the aggregated output is transformed into a probability via the sigmoid function:

$$P(y = 1 \mid x) = \sigma\big(\hat{y}(x)\big) = \frac{1}{1 + e^{-\hat{y}(x)}}. \tag{2}$$

XGBoost is trained by minimising the following regularised objective function:

$$\text{Obj} = \sum_{i=1}^{N} l\big(y_i, \hat{y}(x_i)\big) + \sum_{k=1}^{K} \Omega\big(f_k\big), \tag{3}$$

where the logistic loss is defined as:

$$l\big(y_i, \hat{y}(x_i)\big) = -\left[y_i \log\big(\sigma(\hat{y}(x_i))\big) + (1 - y_i) \log\big(1 - \sigma(\hat{y}(x_i))\big)\right], \tag{4}$$

and the regularisation term for each tree is given by:

$$\Omega\big(f_k\big) = \gamma T_k + \frac{1}{2}\lambda \|w_k\|^2. \tag{5}$$

Here, T_k is the number of leaves in the kth tree, w_k is the vector of leaf weights, λ is the L2 regularisation parameter, and γ controls the minimum loss reduction required to make a further split.

Our implementation employs the following hyper-parameters (obtained after running GridSearchCV with 5-fold cross-validation for hyper-parameter tuning):

- $max\_depth = 14$; $n\_estimators = 400$;
- $min\_child\_weight = 6$; $subsample = 0.8$;
- $learning\_rate = 0.1$; $colsample\_bytree = 0.6$;
- $\gamma = 0.2$; $objective =$ "binary:logistic".

Thus, our XGBoost model solves the following optimisation problem:

$$\min_{f_1,\ldots,f_K} \left\{ \sum_{i=1}^{N} - \left[y_i \log\Big(\sigma\Big(\sum_{k=1}^{K} f_k(x_i)\Big)\Big) + (1-y_i)\log\Big(1-\sigma\Big(\sum_{k=1}^{K} f_k(x_i)\Big)\Big)\right] + \sum_{k=1}^{K}\left(\gamma T_k + \frac{1}{2}\lambda\|w_k\|^2\right)\right\}. \quad (6)$$

This formulation, together with our selected hyper-parameters, forms the foundation for our XGBoost implementation in credit risk prediction.

3.2 Support Vector Machine (SVM)

We employed a soft-margin SVM [3] to allow for some misclassification. Unlike a hard-margin SVM, which assumes perfect linear separability, a soft-margin SVM introduces slack variables ξ_i to permit violations of the margin. This flexibility is crucial in real-world scenarios where the data are not perfectly separable, as it provides a balanced trade-off between maximising the margin and minimising classification errors.

The decision function is defined as:

$$f(x) = \text{sign}\Big(w^T\phi(x) + b\Big), \quad (7)$$

where $\phi(x)$ represents an implicit non-linear transformation of the input space via a kernel function, $w \in \mathbb{R}^p$ is the weight vector, and $b \in \mathbb{R}$ is the bias term.

The optimisation problem for a soft-margin SVM is formulated as:

$$\min_{w,b,\xi} \quad \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \right\}, \quad (8)$$

subject to

$$y_i\Big(w^T\phi(x_i) + b\Big) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1,\ldots,N. \quad (9)$$

Here, the slack variables ξ_i allow for margin violations, and the regularisation parameter $C > 0$ controls the trade-off between margin maximisation and the penalty for misclassification. A high C forces the model to fit the training data closely (with a risk of over-fitting), while a lower C allows more misclassification but may yield a larger margin and better generalisation. In our implementation, we use the Radial Basis Function (RBF) kernel, which is defined as:

$$K(x_i, x_j) = \exp\Big(-\gamma\|x_i - x_j\|^2\Big), \quad (10)$$

where γ is the kernel coefficient that controls the influence of individual training samples. Our specific SVM model uses the following hyper-parameters (also obtained after using GridSearchCV with 5-fold cross-validation):

- $kernel =$ 'rbf' to capture non-linear relationships.
- $C = 10\,000$ to strongly penalise misclassification.
- $gamma =$ 'scale' to automatically determine the kernel coefficient based on feature variance.

3.3 Random Forest

Random Forest [4] is an ensemble learning technique that builds multiple decision trees on bootstrapped samples and aggregates their outputs via majority voting to enhance predictive accuracy and reduce over-fitting.

Suppose that we have a training set:

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}, \quad \mathbf{x}_i \in \mathbb{R}^d \quad (11)$$

Additionally, take the random forest that consists of B decision trees, and the prediction for $\mathbf{x}_i$ is the result of voting:

$$f(\mathbf{x}_i) = \arg\max_k \sum_{b=1}^{B} \mathbb{I}\left(T_b(\mathbf{x}_i) = k\right), \quad (12)$$

Note that $T_b(\mathbf{x})$ is the prediction from the decision tree b, $\mathbb{I}(\cdot)$ is the indicator function. Suppose we have K classes, k $\in \{1, 2, \ldots, K\}$

For each decision tree T_b, we do sampling with replacement from the original dataset $\mathcal{D}$ to obtain a boostrap subset $\mathcal{D}^{(b)}$, which is then used to train the corresponding decision tree.

For each node splitting in a single decision tree, we randomly select $m \ll d$ features, calculate all Gini impurities. The best splitting node should be the one that minimizes the Gini index.

Note that p_k denotes the proportion of samples that belong to class $k \in \{1, 2, \ldots, K\}$. The Gini impurity is defined as:

$$G = 1 - \sum_{k=1}^{K} p_k^2 \tag{13}$$

For a given observation, each decision tree provides a classification prediction. The final output is the class that receives the highest number of votes.

Our implementation employs the following hyper-parameters (obtained after running RandomizedSearchCV with 5-fold cross-validation for hyper-parameter tuning):

- $class\_weight = balanced$; $max\_depth = 24$;
- $max\_features = 0.5$; $min\_samples\_leaf = 4$;
- $min\_samples\_split = 4$; $n\_estimators = 246$.

4 Results and Discussion

We first examine the classification quality of each model, standard metrics such as precision, recall, and specificity, in order to understand how well each customer class is predicted. Then we move into a more detailed analysis of the best performing model, XGBoost, and comprehensively assess its performance using a suite of tests. In addition to standard metrics such as accuracy and the classification report, we compute the Receiver Operating Characteristic (ROC) curve with its Area Under the Curve (AUC) and the Kolmogorov–Smirnov (KS) statistic to evaluate the model's ability to discriminate between high-risk and low-risk customers. We also review a breakdown of the most important features contributing to the model's output. Finally, we conclude by comparing our outputs with the original article's outputs for XGBoost, which was also their best performing model.

4.1 Classification metrics

We start by taking the confusion matrix output, which converts our binary classification into counts of correct versus incorrect predictions, split by true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). We then compute the following results:

Model	Class	Precision	Recall	F1 Score	Support
XGBoost	0	0.99	0.99	0.99	7902
	1	0.38	0.36	0.37	97
RF	0	0.99	0.99	0.99	7902
	1	0.26	0.33	0.29	97
SVM	0	0.99	0.96	0.98	7902
	1	0.13	0.44	0.20	97
LSTM	0	0.99	0.98	0.98	7902
	1	0.17	0.38	0.23	97
MLP	0	0.99	0.94	0.97	7902
	1	0.09	0.44	0.15	97
XGBoost - SMOTE/ENN	0	0.99	0.97	0.98	7902
	1	0.16	0.41	0.23	97
XGBoost - SMOTE/Tomek	0	0.99	0.99	0.99	7902
	1	0.23	0.37	0.28	97

Table 1: Model evaluation metrics. Formulas for each metric are provided in Appendix D.

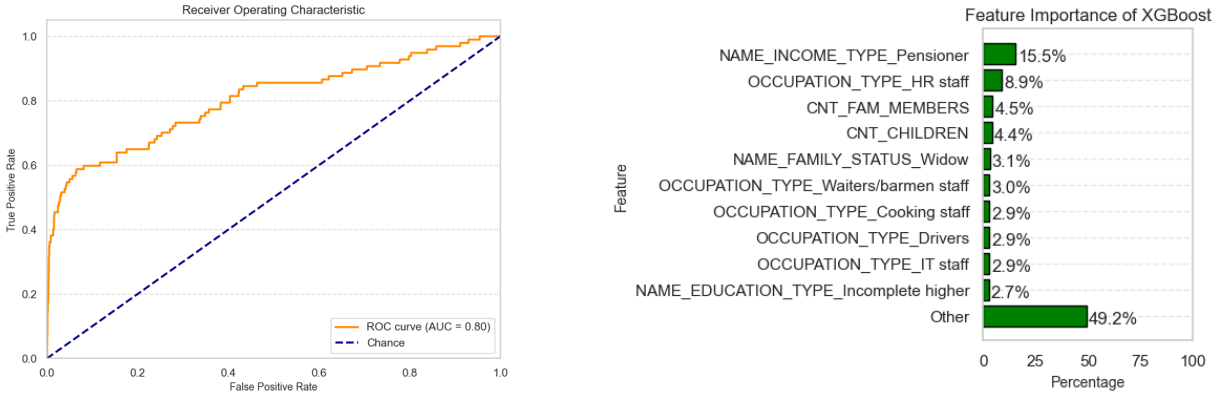
Table 1 presents the performance of the two models evaluated in this study. The precision, recall, and F1 score metrics reflect the models' ability to accurately classify good and bad customers. While all models

classify good customers almost perfectly (which is expected given the level of imbalance in the dataset), XGBoost outperforms all models in most metrics, indicating strong overall classification, particularly when accounting for class imbalance and particularly in terms of precision (0.38) and F1 score (0.37) of the bad class. The bottom two rows display the XGBoost's performance under the alternative sampling methods of choice. Although there is some variation it still performs relatively better than other models, but it scores relatively less in most metrics compared to its SMOTE-only counterpart. Given its superior performance and no noteworthy performance increases being derived from alternative sampling methods, the XGBoost model under pure SMOTE sampling will be examined in greater detail in the subsequent analysis.

4.2 Detailed Analysis of XGBoost Results

Following the previous section, we provide a detailed analysis of the XGBoost model's results below.

4.2.1 Additional metrics



(a) XGBoost - ROC Curve and AUC

(b) XGBoost - Feature Importance

Figure 7: XGBoost performance and key features

- ROC Curve and AUC: the ROC curve displays the trade-off between the true positive rate and false positive rate at various threshold settings. The ROC plot in Figure 7a shows that the curve is well above the diagonal chance line, indicating that the model distinguishes between good and bad customers effectively. The computed AUC, which summarizes the overall discriminative ability of the model, reinforces this observation. A high AUC (close to 1) implies that the model reliably ranks high-risk customers higher than low-risk ones.

- KS Statistic: the KS statistic of 0.522 with a p-value of 0.000 indicates that there is a maximum difference of about 52% between the cumulative distributions of the predicted probabilities for the positive and negative classes. This substantial difference suggests that the model has a strong ability to separate the two classes. Essentially, a high KS value confirms that the model's scoring is effective in ranking customers by risk, which is a key requirement in credit risk prediction.

- The Feature Importance Chart (Figure 7b) for XGBoost illustrates the relative contribution of various features to the model's predictions. The most influential features include whether a customer is a pensioner, HR staff, and the number of members in their family, accounting for 15.5, 8.9, and 4.5 percent of the model's decision-making, respectively. These findings suggest that demographic factors such as income type and occupation, along with family size, are critical in predicting the target variable. Furthermore, the 'Other' category, which aggregates the importance of less prominent features, constitutes 49.2 percent of the total importance, emphasizing the cumulative effect of multiple factors. This analysis underscores the complexity of the XGBoost model, where both individual features and the collective influence of a broad range of factors contribute to the prediction process.

4.3 Comparative Analysis

We also attempted to run the XGBoost model configured with the original article's suggested hyper-parameters. It still demonstrates robust performance on the majority class, achieving an overall accuracy of around 97%. This confirms that XGBoost is highly effective at classifying the bulk of "good" customers by leveraging the extensive features and patterns in the data. However, the precision for the minority (bad) class is noticeably lower (at 16%, compared to the 38% shown above for our optimal parameters), this outcome is expected given the inherent data imbalance, difficulty in predicting rare events in credit risk contexts and highlights the importance of hyper-parameter tuning.

When comparing our results to those reported in the article, we note that the original study boasted near-perfect precision and recall values, approximately 99% for good customers and 99.3% for bad customers, with an AUC close to 1.0. Our inability to replicate such flawless performance strongly suggests that there may be fundamental flaws in their methodology. In particular, an examination of their classification report indicates that the test set was oversampled as well, as evidenced by nearly equal support figures (around 7422 for good customers and 7406 for bad customers). Oversampling the test set introduces data leakage and is unrealistic for practical applications, since in real-world scenarios models are deployed on authentic, unaltered data.

Despite the lower performance on the minority class in our results, the high overall accuracy and strong performance on the majority class indicate that XGBoost is learning useful distinctions. The discrepancy for the bad class underscores the well-known challenge of imbalanced datasets, where even a powerful classifier may struggle to capture rare events accurately, but our results nonetheless reflect the typical trade-offs in genuine credit risk scenarios. Identifying even an additional smaller fraction of high-risk customers can lead to meaningful improvements in financial decision-making.

In conclusion, although neither our optimised hyper-parameters nor our replication using the original ones achieve the near-perfect scores reported in the article, our findings provide a more plausible and credible benchmark for real-world credit risk prediction. The high accuracy and robust majority-class metrics highlight the strength of XGBoost, while the more realistic (albeit imperfect) performance on the minority class reveals areas for further refinement.

5 Final Remarks

Our work set out to replicate and extend the methodology of a published article on credit risk prediction, with a focus on employing sound data-handling practices and rigorous evaluation techniques. We began by carefully pre-processing the data ensuring proper normalisation, applying oversampling only to the training set to prevent data leakage, and converting categorical variables to one-hot encoded formats to create a robust foundation for our models. By comparing our results with the extreme metrics reported in the original study, we highlighted significant methodological discrepancies, notably the unrealistic oversampling of the test set. Our approach provided more plausible and reliable performance estimates that align with practical expectations in a real-world credit risk context.

Although we do not claim to have developed a foolproof methodology, the insights gained throughout this process have been invaluable. They not only enabled us to construct a more credible model evaluation framework but also established a solid foundation for critically assessing published research. Ultimately, our work demonstrates the importance of adhering to sound data processing practices and rigorous evaluation methods, setting a benchmark for future research and providing a basis for constructive criticism of existing methodologies in the field of credit risk prediction.

References

[1] Chang, V., Sivakulasingam, S., Wang, H., Wong, S.T., Ganatra, M.A., & Luo, J. (2024). Credit Risk Prediction Using Machine Learning and Deep Learning: A Study on Credit Card Customers. *Risks, 12*, 174. https://doi.org/10.3390/risks12110174.

[2] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785.

[3] Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning, 20*(3), 273–297.

[4] Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5–32.

[5] Chawla, N.V., Bowyer, K.W., Hall, L.O., & Kegelmeyer, W.P. (2002). SMOTE: Synthetic Minority Oversampling Technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

[6] Bach, M., Werner, A., & Palt, M. (2023). The Proposal of Undersampling Method for Learning from Imbalanced Datasets. *Proceedings of the 23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*. Silesian University of Technology, Gliwice, Poland.

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

A Appendix - Data and Software

A.1 Dataset Summary

We utilised the "Credit Card Approval Prediction" dataset available on Kaggle at https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction/data.

A.2 Software

Our analysis was implemented using Python, and we employed the following libraries and packages:

- Pandas and NumPy: For data manipulation, cleaning, and numerical operations.
- Scikit-learn: For model evaluation, data splitting, and hyper-parameter tuning (GridSearchCV), as well as computing metrics like accuracy, ROC/AUC, and more [7].
- XGBoost: For building the gradient boosting model that was a key component of the original study.
- Imbalanced-learn (imblearn): Specifically, the SMOTE algorithm was used to address class imbalance by oversampling the minority class in the training data.
- TensorFlow/Keras: For implementing and tuning multi-layer perceptron (MLP) models, including the use of early stopping and the Adam optimiser.
- Joblib and tqdm: For parallel processing and progress visualisation during hyperparameter grid searches.
- Matplotlib and Seaborn: For creating visualisations, including histograms, ROC curves, and various other plots that helped in EDA and model evaluation.

B Appendix - Variable Descriptions

B.1 Application Record

The Application Record dataset includes several variables that provide a profile of each applicant:

- **ID**: is a unique identifier assigned to every applicant.
- **CODE_GENDER**: indicates the applicant's gender (commonly "M" for male or "F" for female).

- **FLAG_OWN_CAR**: shows whether the applicant owns a car ("Y" for yes, "N" for no).
- **FLAG_OWN_REALTY**: shows whether the applicant owns property ("Y" for yes, "N" for no).
- **CNT_CHILDREN**: the number of children the applicant has.
- **AMT_INCOME_TOTAL**: reports the total annual income (currency unknown).
- **NAME_INCOME_TYPE**: describes the source or type of income (for example, "Working" or "Pensioner").
- **NAME_EDUCATION_TYPE**: the highest level of education achieved by the applicant.
- **NAME_FAMILY_STATUS**: records the applicant's marital status.
- **NAME_HOUSING_TYPE**: provides information about the type of housing the applicant occupies.
- **DAYS_BIRTH**: the applicant's age in negative day counts (e.g., –10000, approximately a 27 years old).
- **DAYS_EMPLOYED**: similarly indicates the duration of employment as a negative day count.
- **FLAG_MOBIL**, **FLAG_WORK_PHONE**, **FLAG_PHONE**, and **FLAG_EMAIL**: indicate whether the applicant has a mobile phone, work phone, personal phone, and email address respectively.
- **OCCUPATION_TYPE**: variable categorises the type of occupation, for instance "Labourers" or "Managers".
- **CNT_FAM_MEMBERS**: denotes the family size.

B.2 Credit Record

The Credit Record dataset tracks each client's monthly loan performance over a designated period:

- **ID**: unique identifier for each client, matching the application record.
- **MONTHS_BALANCE**: indicates the relative month of record extraction (0 for the current month, –1 for the previous month, and so on).
- **STATUS**: represents the client's monthly payment performance. Numeric values indicate days overdue: 0 for 1–29 days, 1 for 30–59 days, 2 for 60–89 days, 3 for 90–119 days, 4 for 120–149 days, and 5 for write-offs or bad debts (over 150 days overdue). The alphabetic value "C" denotes a fully paid month, while "X" indicates no active loan for that month.

C Appendix - Correlation Matrix

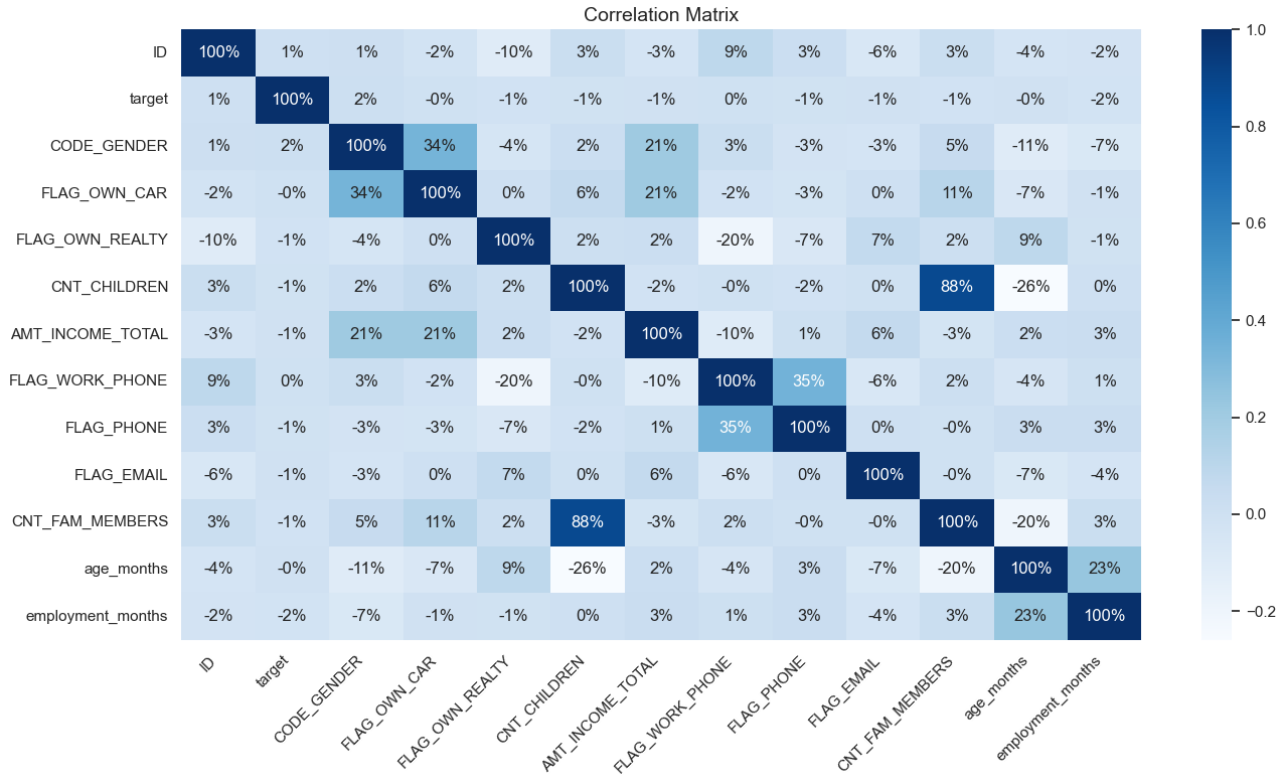


Figure 8: Correlation Matrix

D Appendix - Classification Metrics

Below are the formulas used to compute the classification metrics for each model:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}, \qquad \text{Precision} = \frac{TP}{TP + FP},$$

$$\text{Recall} = \frac{TP}{TP + FN}, \qquad \text{Specificity} = \frac{TN}{TN + FP}, \qquad \text{F}_1 = \frac{2\,(\text{Precision})(\text{Recall})}{\text{Precision} + \text{Recall}}.$$