Loan approval classification

Leonardo Palestra, Marco Gelsomini
Università degli studi Milano-Bicocca, CdLM Data Science

Abstract
This project tackles the challenge of identifying risky loans in a highly imbalanced dataset, where risky loans are significantly underrepresented. The goal was to minimize false negatives, as misclassifying risky loans can result in major financial consequences. The methodologies used to handle class imbalance were **SMOTE** and undersampling, models were selected based on the **F1-score** to balance precision and recall, and optimized using **ROC-AUC**. The best model achieved a **ROC-AUC score of 0.517**, demonstrating the difficulty of distinguishing risky loans, with opportunities for further improvements.

Contents

1 Introduction

The aim of this work is to construct a classifier. This classifier need to learn how to distinguish between a risky and non-risky applicants. For training this model, an open data set that contains the labels of thousands of applicants it is used with 11 variables that describe them. This is a typical scenario where for a bank it could be useful and money-saving to know in advance if one of their customers is creditworthy. For example in the paper Khandani et al. (2010)[1] a nonlinear classifier and forecaster are built assigning a risk score. As the decision to grant or not to grant a loan can have a sensitive impact on an applicant's life, the article by Addo et al. (2018)[2] that analyzes the implication of using different models in assessing and granting credit. If we think about the nature of the phenomenon, it is worth also to mention that the "sensible" scenario is the rarest one. In fact, generally the applicants are credit worthy and ask for a loan knowing if their financial situation allows it. This introduced from a technical point of view the problem of imbalance learning. In this setting the model train on the vast majority of instances labeled as non risky while only a few associated with risky category. Moreover classify a risky applicant as non-risky has a much greater cost respect to misclassify a non-risky one.

2 Data Exploration

The first thing to do in order to better comprehend the dataset is an exploratory data analysis. The data consists of 252000 rows,11 columns describing different aspects of an applicant:

1. Economic situation (*Income, House_Ownership, Car_Ownership, Current_House_Years*).

2. Job-related indicators (*Experience, Profession, Current_Job_Years*).

3. Socio-demographic (*City, State, Age, Married/Single*).

If the Id variable is excluded from the perimeters and we count the duplicates row we have 2088101 duplicates. If we remove these instances, we remain with 36007 unique instances plus and 7183 "former" duplicates. In general duplicates does not add informative power to the analysis, moreover the only discriminant is the ID column but it could be result from automatic procedure, therefore only the dataset without duplicate is considered.

As the final aim is to learn if an applicant can be labeled as risky we analyze the target variable *Risk_Flag* and the relationships with the other variables. In Figure 1 various boxplots are drawn, for each covariate, in order to have a general understanding of both the univariate distribution and the realtionships with the target.

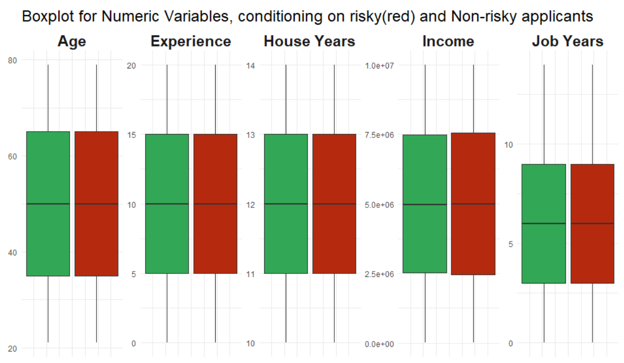


Figure 1

It is clear that, from a single covariate numeric prospective, both the general distribution and the conditional one exhibit a surprisingly symmetrical pattern, centered on the same values and with no outliers. This may suggest us that the eventual patterns with the target are non-linear and hidden between various variables together or relevant only in particular subset of data. If we concentrate on the categorical features like city we can see that in aggregate there are city that contains higher proportion of risky applicants has shown in figure 2.

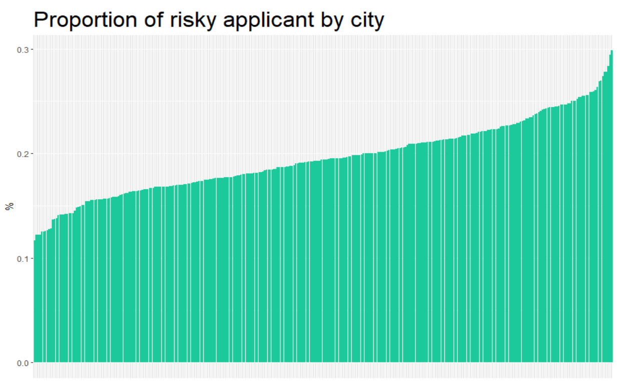
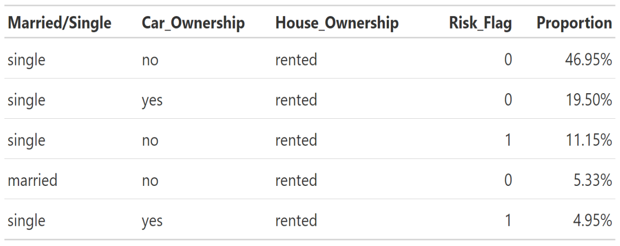


Figure 2

While for the other categorical variable see the count of occurrences in figure 3, showing concentration for a particular set of values.



Married/Single	Car_Ownership	House_Ownership	Risk_Flag	Proportion
single	no	rented	0	46.95%
single	yes	rented	0	19.50%
single	no	rented	1	11.15%
married	no	rented	0	5.33%
single	yes	rented	1	4.95%

Figure 3

Is useful also to show correlation between predictors, in this case the results are shown in the correlation plot in figure 4. This indicates almost no correlation among numerical features, except that between experience and job years.

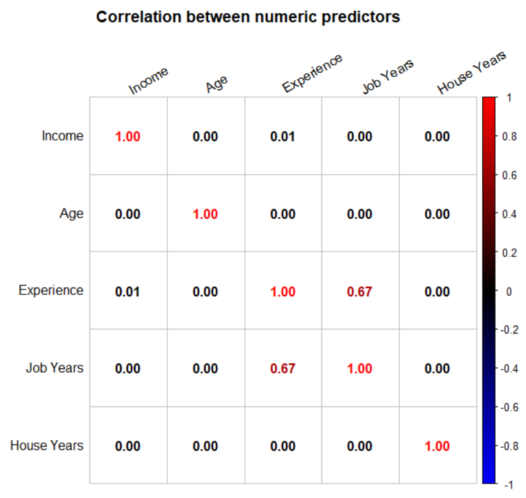


Figure 4

3 Preprocessing

In order to make the dataset more suitable for the analysis we have implemented the following changes:

3.1 Handling of duplicates

- Removed duplicate rows: To ensure the quality and reliability of the dataset, duplicate rows were identified and removed. In general duplicates does not add informative power to the analysis, moreover the only discriminant is the ID column but it could be result from automatic procedure, therefore only the dataset without duplicate is considered.These duplicates, often resulting from data entry errors or redundancies during data collection or merging processes, can introduce bias, overemphasize certain patterns, or negatively impact the performance and generalization of the machine learning model. By eliminating duplicates, the number of rows was reduced from 252.000 to 43.190.

- Removed redudand *Cities* and *States*: To improve the consistency and usability of the dataset, state and city names with minor variations (i.e., *'Tiruchirappalli[10]'* and *'Tiruchirappalli'*) were identified and consolidated. Such discrepancies often result from data collection inconsistencies or annotations, and if left unaddressed, they could lead to fragmented data points or inaccuracies in analysis

3.2 Feature Encoding

- Binary encoding of *Married/Single* and *car_own*: The binary variable was encoded as 0 and 1 to represent its two distinct categories numerically. This transformation simplifies processing by the machine learning model while preserving the original categorical distinction

- Label encoding of *house_own*: Label encoding was applied to the categorical variable `house_own` to convert its categories into numerical values. The encoding was performed as follows:

  - *no_rent_no_own* → 0
  - *rented* → 1
  - *owned* → 2

  This order was chosen to reflect an increasing degree of ownership and responsibility, ensuring that the encoded variable maintains a logical ordinal relationship among the categories. This transformation prepared the variable for input into machine learning models while preserving its meaningful distinctions.

- Target encoding *city*, *state*, and *profession*: Target encoding was applied to these categorical variables due to their high cardinality. For each category, the mean of the target variable *risk_flag* was used as the encoded value. This approach helps reduce dimensionality while preserving useful information, making it more efficient than one-hot encoding for variables with many unique categories.
  Target encoding is computed on the training set and then applied separately to the test set, ensuring no data leakage.

3.3 Normalization

All variables were normalized to bring them to the same scale. This step ensures that no feature dominates the model due to its larger magnitude, improving model performance and stability, especially for algorithms sensitive to feature scaling (i.e. KNN).
In order to avoid data leakage normalization is computed first to the training set and then used (with the mean and standard deviation of the training set) to the test set.

4 Handle unbalanced data

Each model is tested using different methodologies: Holdout (Used as baseline), undersampling and oversampling with SMOTE, the last two techniques are used to handle imbalance.

4.1 Holdout

The holdout technique was used to split the dataset into training and testing sets, with 75% of the data allocated for training and the remaining 25% for testing. To ensure that the class distribution was preserved in both sets, *stratified sampling* was applied. This method ensures that each class is proportionally represented in both the training and testing sets, preventing bias caused by imbalanced classes. Stratified sampling helps improve the model's generalization ability and ensures more reliable evaluation results.

4.2 Undersampling

After applying the holdout technique, *undersampling* was performed on the training data only. The undersampling process involved reducing the number of observations with class 0 to match the number of observations with class 1. This was achieved by using *linear sampling*, which randomly selects and removes observations from the majority class (class 0) until the dataset is balanced. This approach was applied exclusively to the training set to prevent information leakage from the testing set, ensuring a fair evaluation of model performance.

4.3 Oversampling

For handling class imbalance, *oversampling* was applied to the training data (coming from Holdout) using the SMOTE (Synthetic Minority Over-sampling Technique) algorithm[3]. SMOTE generates synthetic examples of the minority class by interpolating between existing minority class instances, creating new data points that are similar to the original ones. This technique helps improve model performance by providing the model with more balanced data, thus reducing bias towards the majority class. For this process, 10 nearest neighbors were used to generate the synthetic samples, and the minority class was oversampled to match the number of observations in the majority class. SMOTE is particularly useful in situations where the minority class has too few examples, ensuring that the model is better able to generalize to both classes.

To avoid generate too many synthetic rows, as the consequences of the class imbalance, before apply the SMOTE algorithm we have applied an under sampling which reduced the class imbalance from 80-20 to 70-30.

Also, when using cross validation, we have applied the SMOTE technique only to the training set within each fold and not to the validation set, this avoid over optimistic results because if applied before the cross validation some synthetics instances of the minority class could be present in the validation test and these generated instances, by construction, are closely related to the ones in the training. Moreover keeping an unbalanced validation tests can assess the model as in a real application scenario.

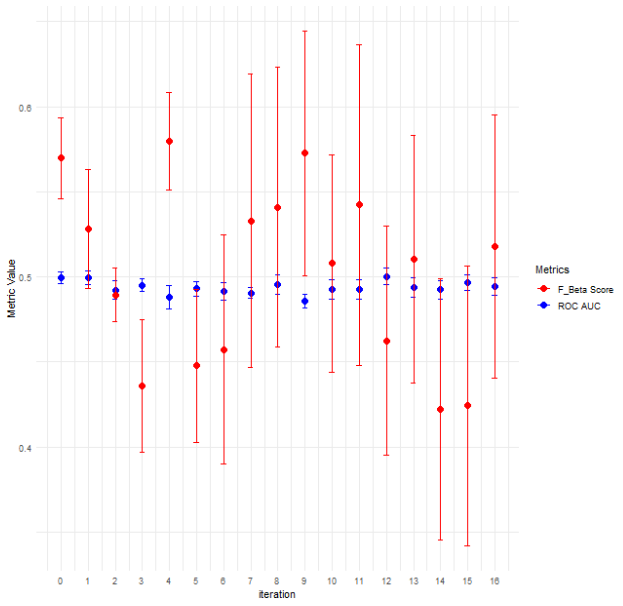
5 Train and optimization

5.1 Models

For this analysis, four different machine learning algorithms were chosen, each representing a distinct class of models. These models, implemented using Weka 3.7 in KNIME, were selected to provide a diverse set of approaches to the classification task. Each algorithm offers unique strengths, making them suitable for handling various types of data patterns and relationships.

- **Logistic Regression**[4]: A linear classification model that estimates the probability of an event based on a logistic function. Chosen as a representative of *linear models.*
- **Random Forest**[5]: An ensemble method that builds multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. Chosen as a representative of *ensemble methods.*
- **Naive Bayes**[6]: A probabilistic classifier based on Bayes' theorem, assuming independence between features. Chosen as a representative of *probabilistic models.*
- **K-Nearest Neighbors (KNN)**[7]: A non-parametric method that classifies data points based on their proximity to others in the feature space. Chosen as a representative of *instance-based learning algorithms.*

5.2 Parameter tuning



Parameter tuning results of KNN in SMOTE

For each of the models and for each of data handling technique (Holdout,Undersampling and Oversampling) we have applied a parameter tuning by constructing samples of possible configurations of parameters from a grid of values obtained with Latin hypercube sampling (LHS)[8] (for example in random forest models) and applied them to the model using a 10-fold **cross validation**[9] to obtain robust estimations. An example of one these results is shown in picture above.

To select the best set of parameters during the cross validation process we used the Fβ-score with $\beta = 2$, which places more emphasis on **recall** rather than precision. This choice was made because, in the context of loan risk classification, it is crucial to minimize false negatives—i.e., to ensure that risky loans are correctly identified. A higher β value (such as 2) allows the model to prioritize detecting these risky loans, even at the cost of potentially increasing the number of false positives. The image 5 shows an example of the process descripted.

6 Evaluation and comparison

To evaluate the performance of the models, two key metrics were used: *F1-score* and *ROC (Receiver Operating Characteristic)*

6.1 F1-Score

After prioritizing recall in the initial filtering (5.2), we evaluated the models based on their F1-score for the positive class (class 1, representing risky loans). The F1-score is the harmonic mean of precision and recall, balancing the trade-off between the two. Precision measures the proportion of correctly identified risky loans out of all loans predicted as risky, while recall captures the proportion of correctly identified risky loans out of all actual risky loans.

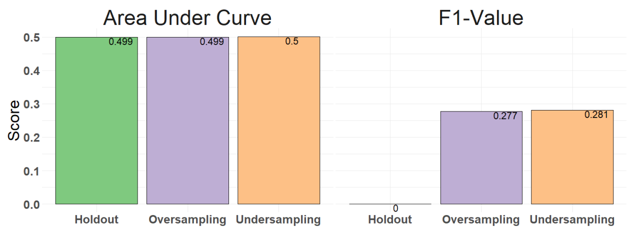
We excluded models with F1 values equal and near zero so we can avoid classifier that predicts only one class.

6.2 ROC-AUC

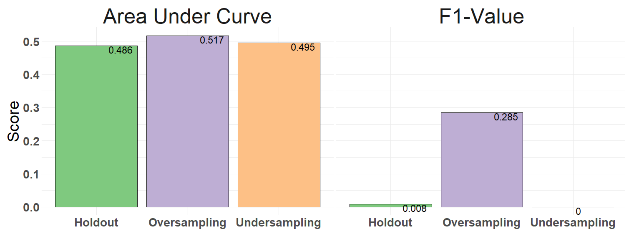
Finally, we used the ROC curve to assess the models' discriminative ability. The ROC curve (Receiver Operating Characteristic curve) plots the true positive rate (sensitivity) against the false positive rate (1-specificity) for different decision thresholds. It helps evaluate the model's ability to distinguish between the positive (risky) and negative (non-risky) classes across various thresholds, irrespective of the class distribution.

By combining the F1-score and the ROC curve, we ensured that the selected model demonstrated both balanced predictive performance and strong discriminatory power, which is essential for optimizing loan risk assessment in imbalanced datasets.

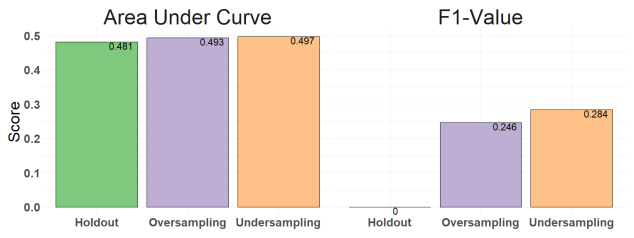
These are the best metrics, *with all the assumptions that we have made*, for each type of model and for each type of data handling (Holdout, Oversampling, Undersampling) founded by maximizing the f1-score:



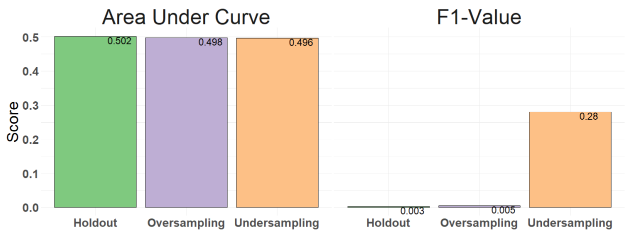
Logistic regression



KNN

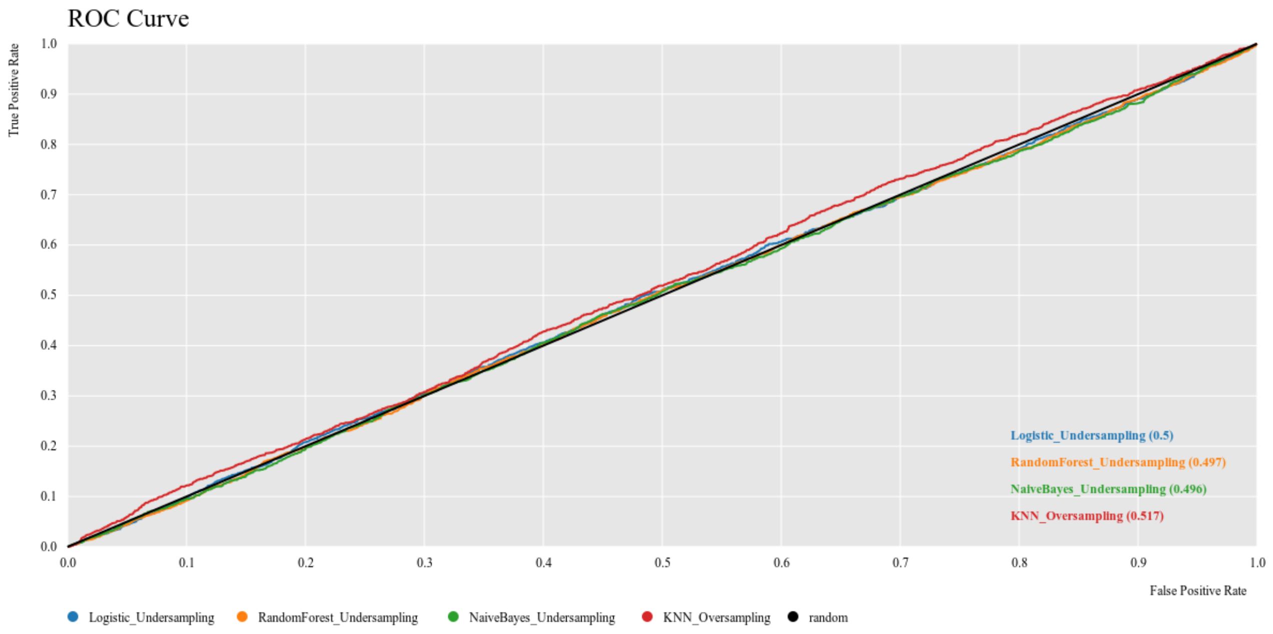


Random Forest



Naive Bayes

We can find the best possible model filtering them by considering an acceptable value of F1-score (as described before) and maximizing the ROC curve:



ROC CURVE OF THE BEST MODELS

So, the best model, is the KNN with K-optimal equal to 13 in the configuration of the Oversampling technique.

7 Conclusions

In the loan risk classification project, we addressed the problem of **class imbalance**, where the minority class (risky loans) is crucial for financial decisions. We applied various data management techniques such as **SMOTE**, **undersampling**, and **holdout**.

The models tested include **Logistic Regression**, **Random Forest**, **K-Nearest Neighbors**, and **Naive Bayes**, representing different categories of machine learning models.

During the **parameter optimization** phase, we utilized **cross-validation** combined with **Latin Hypercube Sampling (LHS)** to efficiently explore the parameter space of each model and select the best performing combinations. In this phase, we used the **F2-score** (with $\beta = 2$) to prioritize **recall**, minimizing false negatives, which are crucial in this case, as failing to identify risky loans (class 1) can have severe financial consequences.

Subsequently, in the final phase, we selected the best model based on two criteria: first, we used the **F1-score** to ensure that the models did not predict the same class (all zero or all one) consistently, thus ensuring a balanced evaluation between **precision** and **recall**. This helped exclude models with poor performance, favoring more reliable ones. Finally, we selected the best model based on the **ROC curve**, choosing the one with the highest ROC-AUC score. This approach ensured that the model could effectively separate risky loans (class 1) from non-risky loans (class 0). In practice, the 1-score allowed us to avoid models that tended to predict the same class, while the ROC curve maximized discrimination between the two classes, improving the model's overall performance.

Despite the improvements in selecting models that were more sensitive to the minority class, the final model achieved a **ROC-AUC score of 0.517**, indicating that the separation between risky and non-risky loans remains an area for improvement.

For a better performance in a real application it would be useful to have a domain expert knowledge in order to maximize the performance of the classifier by knowing the cost associated to its error.

References

[1] A. E. Khandani, A. J. Kim, A. W. Lo, Consumer credit-risk models via machine-learning algorithms, Journal of Banking Finance 34 (11) (2010) 2767–2787. doi:https://doi.org/10.1016/j.jbankfin.2010.06.001.
URL https://www.sciencedirect.com/science/article/pii/S0378426610002372

[2] P. M. Addo, D. Guegan, B. Hassani, Credit risk analysis using machine and deep learning models, Risks 6 (2) (2018). doi:10.3390/risks6020038.
URL https://www.mdpi.com/2227-9091/6/2/38

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357. doi:10.1613/jair.953.
URL http://dx.doi.org/10.1613/jair.953

[4] J. C. Stoltzfus, Logistic regression: A brief primer, Academic Emergency Medicine 18 (10) (2011) 1099–1104. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1553-2712.2011.01185.x, doi:https://doi.org/10.1111/j.1553-2712.2011.01185.x.
URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1553-2712.2011.01185.x

[5] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[6] G. I. Webb, E. Keogh, R. Miikkulainen, Naïve bayes., Encyclopedia of machine learning 15 (1) (2010) 713–714.

[7] L. E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.

[8] W.-L. Loh, On latin hypercube sampling, The annals of statistics 24 (5) (1996) 2058–2080.

[9] M. W. Browne, Cross-validation methods, Journal of mathematical psychology 44 (1) (2000) 108–132.