Predicting Loan Risk: A Machine Learning Model for Assessment

Ilaria Coccollone[1], Giovanni Noè[2]

Abstract

The application of machine learning algorithms in credit risk assessment has gained significant attention in recent years due to its ability to enhance decision-making and mitigate financial risks. The goal of this research work is to find a well-performing classification model to predict the risk flag of loan applicants making use of some personal information like their professional status, house and car ownership, age and marital status. The dataset employed for this analysis can be found on Kaggle.com and includes data about 252,000 loan requesters.

[1] *University of Milano-Bicocca | MSc Data Science*

[2] *University of Milano-Bicocca | MSc Data Science*

Contents

Introduction

The ability to accurately evaluate credit risk is fundamental for financial stability and efficient resource allocation in the banking and lending industries. Traditional methods of credit risk evaluation face limitations in dealing with the increasing complexity and volume of data. Errors in these assessments can lead to significant financial losses for lenders or a reduction in credit access for potential borrowers. For these reasons the adoption of Machine Learning algorithms has emerged as a promising solution, offering the capability to uncover hidden patterns in data and deliver more accurate and dynamic predictions. The goal of this research is to identify a machine learning model that performs well in predicting the risk associated with each applicant (a binary classification problem) maintaining also interpretability and scalability for real-world applications. Credit applicants exhibit a wide range of risk profiles influenced by various factors, including their professional status, possession of assets, and demographic attributes, so the ML process must take account of this diversity and select only the

relevant features to enhance performance and interpretability. This work will test three different classification models (Logistic Regression, Naïve Bayes and Random Forest) to find the most suitable for the classification problem just introduced. All the work has been carried out through the KNIME Analytics Platform.

1. Data Exploration and Preprocessing

1.1 Data Exploration

The dataset employed to develop the ML solution of this work is the Loan Approval Dataset, available on the Kaggle Platform. It is consisting of 252,000 records representing loan applicants and contains the following attributes for each record:

- **Id**: Unique identifier for each loan applicant (Categorical Nominal)
- **Income**: The income level of the applicant (Numeric Interval)
- **Age**: Age of the applicant (Numeric Interval)
- **Experience**: Years of professional experience (Numeric Interval)
- **Married/Single**: Marital status of the applicant (Categorical Nominal)
- **House_Ownership**: Indicates whether the applicant owns or rents a house (Categorical Nominal)
- **Car_Ownership**: Indicates whether the applicant owns a car (Categorical Nominal)
- **Profession**: Occupation or profession of the applicant (Categorical Nominal)
- **CITY**: City of residence of the applicant (Categorical Nominal)
- **STATE**: State of residence of the applicant (Categorical Nominal)
- **CURRENT_JOB_YRS**: Duration of employment in the current job (Numeric Interval)
- **CURRENT_HOUSE_YRS**: Duration of residence in the current house (Numeric Interval)
- **Risk_Flag**: Binary indicator of loan risk, where 1 represents a flagged risky applicant and 0 represents a non-risky applicant (Categorical Nominal)

1.2 Class Imbalance

The class imbalance problem is a significant issue in the deployment of classification models, it occurs when the classes of the output attribute are not equally represented in the dataset, with one class significantly outnumbering the others. This imbalance can lead to biased models that favor the majority class, resulting in poor predictive performance for the minority class. In the case of the dataset used in this research, the number of records whose value for the class attribute is 1 much smaller than the number of those with a 0 value. Therefore, the class imbalance is a

problem that will be addressed in the model deployment of this work.

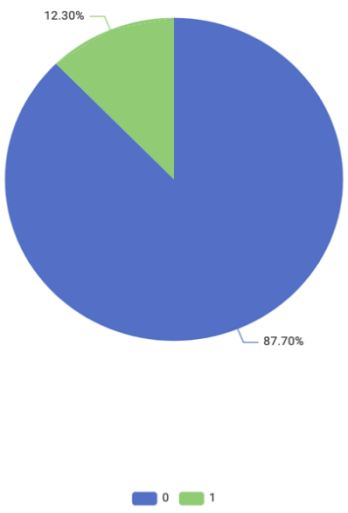

Figure 1: *Distribution of 0 and 1 values for the class attribute*

1.3 Missing Values and "Duplicates"

Before starting the Machine Learning process an important step is to handle missing values and duplicates in the dataset. Fortunately, the dataset employed in this work is free of missing values. Also, at first glance the dataset is free of duplicate rows. However, after removing the Id column, it was found that nearly 83% of the records are "copies" of just 2.85% of the remaining records. Although having so many applicants with the exact same characteristics is a very unlikely scenario, the authors of this work had no evidence of data counterfeiting. Therefore, since each "duplicate" has a unique Id, they were considered distinct applicants and none of the rows was removed.

1.4 Feature Transformation

To compute the correlation between the attributes some subtle but significant changes were made to the columns "Married/Single", "House_Ownership" and "Car_Ownership". Specifically, the "Married/Single" column became "married" (0 = not married, 1 = married), similarly "Car_Ownership" was transformed into "car_owned" (0 = no car, 1 = car owned) and "House_Ownership" came to be "house_owned" (0 = norent_noown, 1 = rented, 2 = owned). For the first two variables a simple binarization process was applied but for the third one there is something more: an order in the values was introduced. This should not happen for a nominal variable but in the opinion of the authors the order that appears makes sense with reality: in the context of house ownership, a person who rents a house stands in the middle between owning and not having any kind of power on the house they live in. Since these transformations are not expected to negatively impact the performance of the classifiers, the authors decided to keep them in both the model development and validation workflows.

1.5 Correlation Matrix

To better handle the process of feature selection, the correlation matrix of the attributes was computed. The correlation matrix reveals that the only relevant correlation exists between the "Experience" and "CURRENT_JOB_YRS" attributes.

Therefore, if one of them is chosen during feature selection, the other should be considered redundant and excluded from the process.

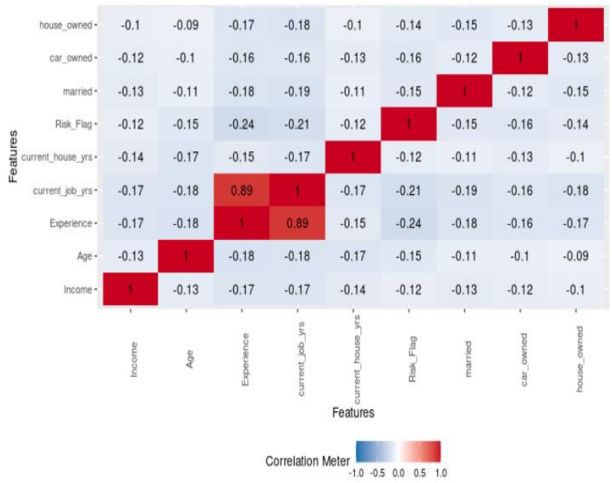

Figure 2: *Correlation matrix between attributes*

2. Feature Selection

2.1 Feature Selection Loop

Before starting the implementation of the models chosen for the analysis, a feature selection process was used to identify the most relevant features of the dataset. This allowed, in addition to increasing the performance of the model, a better interpretation of the results, obtaining a more robust model. To do this, was used a component of the KNIME software called "Feature Selection Loop". Furthermore, the data was normalized to treat all features equally during selection. The loop consists of a “Loop start node” that allows you to select a subset of features from the dataset and start the iterations. The node was configured with a generic algorithm. Inside the loop, a workflow builds a classification model, in this case the Decision Tree model is chosen. The trained model is used by the “Decision Tree Predictor node” to generate predictions, which are then evaluated by a “Scorer node” to measure its performance. The process continues iteratively optimizing the selection of the most relevant features thanks to the Score results. To evaluate the performance, the Cohen's Kappa metric was chosen, as having an unbalanced dataset, a metric such as Accuracy could be misleading, while the first considers that class 0 is in the majority and therefore easier to predict. At the end, the results are collected in the “Feature Loop End node” where is detected the combination of features that best performed.

2.2 Final Features

The final features obtained from the loop, which best optimize the data, were: Id, Income, Age, Experience, Profession and STATE. Since the id is a feature that is not relevant for a relationship with the target, inserting it could lead to overfitting of the data. This is why it was chosen to remove it from the dataset analysis.

3. Model Training

3.1 Cross-validation

In this analysis, three different Machine Learning models were chosen for predicting the value of Risk_Flag. The Logistic Regression, the Naïve Bayes and the Random Forest classifier. To guarantee the robustness

and efficiency of the chosen models, it was decided to use a Cross-validation process. At the beginning of this part of the workflow the "X-Partitioner node" allows to divide the dataset into subsets for training and validation. The number of chosen folds was five. In this way, each observation was used for both training and validation in different iterations. After dividing the dataset into k folds (k = 5), k-1 folds, in each iteration, were used for training, while the remaining partition (fold) was used for validation. This process was repeated five times, such that each partition was used, at least once, for validation. Finally, the results are aggregated in the "X-Aggregator node" to have an optimal overall evaluation. The main reasons why it was decided to use the Cross-validation method are the reduction in the risk of overfitting, as the three models are tested on a fold with data that don't correspond to the training data, allowing for more robust and reliable estimates to be obtained of performances. Furthermore, having an unbalanced dataset, this approach allows us to consider the distribution of classes correctly in each training and validation phase. It also helps to understand if the model is learning the patterns of minority class, without favoring the majority, avoiding bias. The last reason why it was chosen is to help manage "duplicate" data, which was discussed in paragraph 1.3. It can avoid the presence of "duplicates" in both training and validation within the same fold, allowing to check whether the model has generalized or whether it is only storing similar data (in this case possible duplicates).

3.2 Class Balancing

To further manage class balancing and improve model performance, it was decided to use a balancing technique within the cross-validation process. Through the "Row Splitter node" the dataset was divided to separate the two Risk_Flag classes. Subsequently, through the "Counting Loop node", was performed an oversampling to balance the two classes. The choice to include it in the cross validation was made to avoid that information from the test could influence that of the training. In this way the balancing occurs separately for each fold, ensuring greater rigidity without the process being influenced by a balancing done at the start. It also ensures greater generalizability on new data.

3.3 Logistic Regression Training

The first model implemented was Logistic Regression. In KNIME the fundamental nodes to implement the model are the "Logistic Regression Learner node" and the "Logistic Regression Predictor node" to get the final scores. This model was chosen as it is a simple and easy to interpret model for binary targets. It was chosen as a starting point to subsequently evaluate more complex models.

3.4 Naïve Bayes Training

The second model implemented was Naïve Bayes. In KNIME the fundamental nodes to implement the model are the "Naïve Bayes Learner node" and the "Naïve Bayes Predictor node" to get the final scores. The choice of this second model was mainly to test a probabilistic model. Naive Bayes was a scalable model, quick to implement and considered useful for making comparisons.

3.5 Random Forest Training

The last model implemented was Random Forest. In KNIME the fundamental nodes to implement the model are the "Random Forest Learner node" and the logistic "Random Forest Predictor node" to get the final scores.

The Random Forest being an ensemble method is able to handle unbalanced datasets very well, thanks to the random sampling of data, reducing the possibility that the model considers only the majority class. Furthermore, the model reduces the impact of possible duplicates, since each tree can see different information, ensuring greater robustness and greater accuracy of the estimate. All these reasons led to the choice of this optimal model for the analysis.

4. Model Evaluation

4.1 Model Comparison

The last part of the analysis focused on the evaluation of the three implemented models.

The best predictive model was found to be the Random Forest. Table 1 showed the results of the main metrics of the models. It showed that the first two models, Logistic Regression and Naive Bayes, performed well in class 0, almost always identifying positives. The problem, in both models, was in the management of class 1, with very weak results. In summary, the two models fail to correctly predict the minority class. The best model turns out to be the Random Forest. The classifier has an excellent performance for the majority class, all metrics (Recall, Precision and F-measure) are high, indicating that the model is completely reliable in predicting class 0. With higher values for class 1, it is better than the other two models and suitable for offering the best performance.

Model	Recall	Precision	F measure
LR (class 0)	0.577	0.894	0.702
LR (class 1)	0.514	0.146	0.227
NB (class 0)	0.597	0.886	0.713
NB (class 1)	0.454	0.136	0.21
RF (class 0)	0.913	0.966	0.938
RF (class 1)	0.769	0.553	0.643

Table 1: *Model Performance Metrics by class*

4.2 The ROC Curve

Another very useful measure used to compare models is the ROC curve (Receiver Operating Characteristic Curve). The ROC curve illustrates the relationship between a model's sensitivity (True Positive Rate, **TPR**) and 1-specificity (False Positive Rate, **FPR**). This graph is widely used to compare different models based on the AUC (Area Under the Curve), a measure of performance. The AUC ranges from 0 to 1: AUC = 1 perfect model, AUC = 0.5 the model is no better than a random classification, AUC < 5 the model is terrible, with worse performance than a random classification. Since the AUC is independent of the threshold chosen to classify the predictions, it offers a robust and useful way to evaluate and compare models across different scenarios. Figure 3 showed the different AUC for each model.

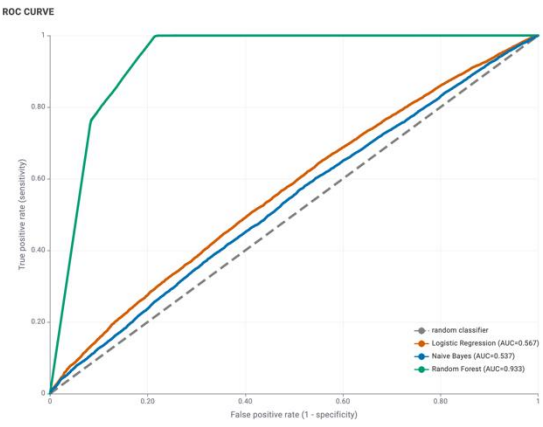

Figure 3: *ROC curve for each model*

Even in this case, the Random Forest (green one) was found to be the best model, with an AUC = 0.933. The Logistic Regression (red one) and the Naive Bayes (blue one) obtained values of AUC = 0.567 and AUC = 0.536 respectively, not very far from those of a random classifier.

Conclusions

At the end of the analysis, it was obtained that a machine learning model such as Random Forest, can be a good ally in predicting the risks of bank loans and all the possible risks that they entail. This analysis is proposed as a beginning of a possible evaluation of models capable of managing challenges such as those of bank loans. For the future, a further implementation of more complex and suitable models is reserved to obtain a further optimization of the predictions of risks and bank credits, which nowadays represent an important point in continuous evolution of modern society and economy.

References

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112, p. 18). New York: springer.

Kaggle. (2021). Loan Approval Dataset. https://www.kaggle.com/datasets/rohit265/loan-approval-dataset

KNIME https://www.knime.com

KNIME Community Hub https://hub.knime.com

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer.