Loan Approval: Exploring Loan Applicant Characteristics and Risk Assessment

Team 31: Aurora Castelnovo[1], Nicole Gemelli[1], Davide Tortorella[1]

Abstract

Loan approval is a vital process in the financial sector, where institutions assess applicants to determine their eligibility for a loan. This evaluation is based on a thorough analysis of the applicant's financial status and demographic details, ensuring informed decisions that balance risk and opportunity.

The "Loan Approval Dataset" provides comprehensive information about loan applicants, including demographic and financial attributes such as age, marital status, employment history, ownership of assets, and income levels.

This study aims to leverage machine learning algorithms to train a predictive model capable of classifying applicants as high-risk or low-risk. Such models can assist financial institutions in enhancing their loan approval strategies, reducing default rates, and improving overall efficiency in credit risk management.

Keywords

Machine Learning—Loan Approval—Risk Assessment—Applicants

[1]Università degli Studi di Milano Bicocca, CdLM Data Science

Contents

Introduction

What defines a risky loan applicant? At its core, a risky loan applicant is someone who has a higher probability of defaulting on loan repayments. This determination is typically based on a combination of financial, employment, and personal factors. Some of the primary indicators of risk include financial stability, job security, and ownership of assets, as these factors reflect an individual's ability to manage debt responsibly and maintain consistent payments over time.

While these factors provide a measurable basis for assessing risk, loan approval decisions also involve more nuanced and context-dependent considerations. For instance, macroeconomic conditions such as inflation, unemployment rates, or economic downturns can influence the overall risk landscape. Similarly, personal circumstances (such as unexpected medical expenses, family responsibilities, or a history of adverse events) can significantly impact an individual's financial behavior.

With advancements in data analysis and machine learning, financial institutions can now incorporate vast amounts of data to better assess loan risk. These technologies analyze complex patterns across numerous variables, enabling more accurate and fair decision-making while reducing biases inherent in manual assessments.

Despite the inherent challenges, it is valuable to investigate whether it is possible to predict the loan risk flag starting from measurable attributes. The dataset chosen to answer this research question is the Loan Approval Dataset, available

from a lending-related data source. It consists of 252,000 records, each with the following 13 features:

- *Id* (categorical-nominal): Unique identifier for each loan applicant
- *Income* (numeric-ratio): The income level of the applicant
- *Age* (numeric-ratio): Age of the applicant
- *Experience* (numeric-ratio): Years of professional experience
- *Married/Single* (categorical-binary): Marital status of the applicant
- *House_Ownership* (categorical-binary): Indicates whether the applicant owns, rents a house, or none of them
- *Car_Ownership* (categorical): Indicates whether the applicant owns a car
- *Profession* (categorical-nominal): Occupation or profession of the applicant
- *CITY* (categorical-nominal): City of residence of the applicant
- *STATE* (categorical-nominal): State of residence of the applicant
- *CURRENT_JOB_YRS* (numeric-ratio): Duration of employment in the current job
- *CURRENT_HOUSE_YRS* (numeric-ratio): Duration of residence in the current house
- *Risk_Flag* (categorical-binary): Binary indicator of loan risk, where 1 represents a flagged risky applicant and 0 represents a non-risky applicant

The goal of our analysis is to predict the risk flag using machine learning tools and evaluate the goodness of the predictions.

This report is organized as follows:

1. **Data Exploration**: we examine the dataset features and initial insights, focusing on *Risk_Flag*.
2. **Preprocessing**: we remove some columns, transform some features (in particular using binarization) and verify if there are some missing values. This is done in order to obtain a more suitable dataset for further analysis.
3. **Dataset configuration and challenges**: we outline the dataset setup and the strategies we used to address class imbalance in the target variable, *Risk_Flag*, ensuring fair model evaluation and improved detection of the minority class.
4. **Models:** we analyze and evaluate various models to predict the target variable, *Risk_Flag*, comparing their performance across multiple metrics to identify the most effective approach. Particular attention is given to Random Forest.
5. **Interpretability:** we explore the interpretability of the model by analyzing the significance of variables using the InfoGain method, providing insights into the key factors influencing predictions.
6. **Conclusion**

All steps and analyses in this report were conducted using **KNIME**, a versatile and intuitive platform for data analytics. KNIME's drag-and-drop interface allowed us to efficiently preprocess the data, implement various Machine Learning models, and evaluate their performance without requiring extensive coding. Its flexibility and powerful capabilities made it an ideal choice for managing the entire workflow, from data exploration to model comparison.

1 Data Exploration

The first step in exploring the dataset involves assessing its completeness and quality. Notably, the dataset contains no missing values, which eliminates the need for imputation and ensures that all variables are fully available for analysis.

Another notable characteristic of the dataset is the presence of duplicates. Out of the original 252,000 rows, removing duplicates would reduce the dataset to approximately 40,000 rows. However, we have chosen to keep these duplicates for several reasons. First, the larger dataset size provides the model with more data to learn patterns effectively, which can improve its

generalization ability. Additionally, removing duplicates significantly impacts model efficiency, as observed during initial experiments where a smaller dataset led to poorer outcomes in training. Retaining these duplicates allows us to preserve variability in the data, which is especially beneficial in scenarios where some duplicates might reflect genuine repeated observations from the real-world data collection process.

Around 90% of applicants are categorized as *Single*, with the remaining only 10% being *Married*. Approximately 90% of them live in rented accommodations, while only a small fraction own houses. Similarly, about 70% of applicants rent their cars, while the remaining 30% own them outright.

Since our goal is to find the best classification model in order to classify applicants based on how risky the loan will be, we consider *Risk_Flag* to be a dependent variable, the target for our analysis. The *Risk_Flag* variable indicates whether a loan applicant is considered high-risk (1) or low-risk (0). One of the primary challenges of this dataset is the significant class imbalance: we notice that approximately 88% of applicants are labeled as low-risk, while, on the other hand, only 12% are categorized as high-risk (as shown in Figure 1). This imbalance poses a potential challenge for the model, as it could lead to biased predictions favoring the majority class. To mitigate this, the evaluation phase will focus on metrics like *Precision, Recall, F1-score, and ROC-AUC*, which are more suited for imbalanced datasets than simple accuracy.

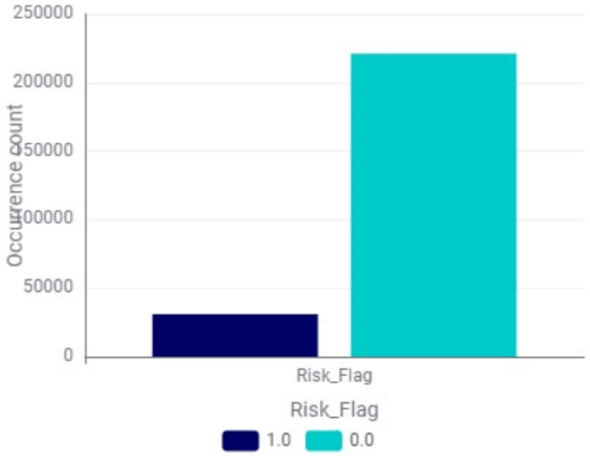


Figure 1. Partition of Risk_Flag

2 Preprocessing

In order to make the dataset more suitable for analysis and to enhance model performance, we implemented the following preprocessing steps.

2.1 Feature Selection

Out of all the 13 original features we have decided to remove the Id column. It is a column that uniquely identifies each row and has no meaningful relationship with the target variable Risk_Flag. Removing it prevents the model from learning spurious patterns that could lead to overfitting without contributing to predictive performance.

We have also checked the correlation between the variables, by using the correlation matrix. While slight correlations were observed, these were not sufficient to warrant removal, ensuring that all relevant features remained for modeling.

Additionally, features related to state, city, and profession were initially encoded using *K-Fold Encoding*, which replaced categorical values with the mean of the target variable for each category. However, these features were later removed because they were observed to cause significant overfitting in the Random Forest model (as we'll discuss further). This decision highlights the importance of carefully evaluating feature contributions to avoid degrading the generalizability of the model.

2.2 Feature Transformation

We applied the following transformations to enhance the dataset's utility for predictive modeling.

Income column: it has been discretized into integer ranges, effectively grouping individuals based on income levels. This transformation makes it easier for models to capture patterns related to income without being overly sensitive to minor variations. Additionally, avoids scaling issues, as income values are orders of magnitude larger than other variables.

Risk_Flag column: the target variable, initially represented as a numerical column (0 for low risk and 1 for high risk), was transformed into a string format. This transformation was necessary to

ensure compatibility with specific machine learning models that required categorical input for the target variable.

2.3 Transformation of Categorical Variables

The dataset is also composed of categorical variables, and those were transformed to ensure they were suitable for machine learning models. In particular, we use two different techniques, depending on the number of values the variable can take.

Married/Single and *Car_Hownership* were binarized by assigning a value 1 for, respectively, "*married*" and "*yes*", and 0 for "*single*" and "*no*". This transformation ensured compatibility with numerical models while preserving their informational value.

House_Ownership includes three categories (*'rented', 'norent_noown', 'owned'*) that are treated by using *one-hot encoding* technique. In this way, for each *n* unique values of the given feature, *n* columns were added to the dataset. The main advantage of one-hot encoding is that it considers each unique value independent from the others, avoiding the addition of correlations not present in the original data.

By performing these transformations, we ensured that categorical features were properly represented without introducing biases or correlations that were not present in the original data.

3 Dataset configuration and challenges

The dataset was divided into two parts:

- **Training set (67%)**
- **Test set (33%)**

The split was performed using *stratified sampling* to ensure that the proportions of *Risk_Flag* classes were maintained.

The dataset used in this analysis exhibits a significant imbalance between the classes of the target variable, *Risk_Flag*. Approximately 12% of instances are labeled as high-risk (1), while 88% belong to the low-risk (0) class. This imbalance poses challenges for classification algorithms, as they tend to prioritize the majority class, potentially neglecting the minority class.

In imbalanced datasets, high accuracy can be misleading. For instance, a model that predicts all instances as belonging to the majority class would achieve an accuracy of 88%, yet it would fail to identify any high-risk applicants, rendering it ineffective.

To address this issue, we relied on metrics better suited for imbalanced datasets, including:

- **Precision:** Measures the proportion of true positive predictions among all positive predictions.
- **Recall (True Positive Rate):** Captures the proportion of actual positive instances correctly identified by the model.
- **F1-Score:** The harmonic mean of Precision and Recall, balancing the trade-off between the two.
- **ROC-AUC:** Represents the model's ability to distinguish between positive and negative classes across different thresholds.

To mitigate the effects of imbalance, the following approach was used: *Stratified Sampling***.** It ensured proportional representation of classes in the training and test sets. This strategy, combined with the use of robust evaluation metrics, ensured a fair assessment of model performance and addressed the challenges posed by the imbalanced nature of the dataset.

4 Models

To predict the **Risk_Flag** variable, we applied several classification techniques based on Machine Learning algorithms. The selected algorithms include:

Probabilistic Models: Include **Naive Bayes**, which assumes attribute independence for efficient tasks like text classification, and **Bayesian Networks**, which allow attribute dependencies for more accurate modeling.

Heuristic Models: Include **Random Forest**, an ensemble method that builds multiple decision trees to handle non-linear relationships, and **J48**, a decision tree algorithm based on C4.5, known for interpretability and reliability.

Regression Models: **Logistic Regression** predicts binary outcomes by applying a logistic function, offering simplicity and interpretability.

Artificial Neural Networks: Include **MLP (Multi-Layer Perceptron)**, a feedforward network optimized via backpropagation, and **SpegaSOS**, which enhances learning through structured optimization for complex data modeling.

Each of these algorithms was selected to address different aspects of the classification problem. Heuristic models like Random Forest excel at handling feature interactions and non-linear patterns, while probabilistic models offer interpretability and simplicity. Regression models provide a straightforward approach to capturing linear relationships, and neural networks bring the capability to learn complex, non-linear representations.

This combination of methods ensures a comprehensive exploration of the data, enabling us to identify the most suitable approach for predicting the *Risk_Flag* variable effectively.

In this part of the analysis we decided to not use Cross-validation (but we'll use it further), as stratified sampling already ensures a balanced representation of classes in both subsets and because we weren't interested yet in a validated value of the performance metrics but we were looking for the general best algorithm.

Additionally, at this stage of the analysis, the primary metric used to evaluate model performance was the *ROC Curve*. Other metrics will be considered in later stages for a more comprehensive evaluation.

All the algorithms discussed in this study were evaluated using a dataset where the columns **City**, State, and Profession were transformed using K-Fold Encoding to ensure robust handling of categorical data. This encoding method prevents data leakage and provides a more accurate representation of feature interactions during training and testing.

Each algorithm was tested with its default parameters, under three distinct configurations to assess their performance:

1. **Using all the columns** in the dataset, providing the models with the full feature set.
2. **Using features selected through *InfoGain***, a method that ranks attributes based on their information gain relative to the target variable.
3. **Using features selected through *CfsSubsetEval***, which evaluates subsets of attributes based on their correlation with the target class and minimal redundancy among themselves.

The results consistently showed that the highest *ROC curve* values were achieved when all columns were used, highlighting the importance of leveraging the full feature set for optimal performance. This suggests that neither *InfoGain* nor *CfsSubsetEval* captured all the nuances and interactions present in the complete dataset, reinforcing the value of comprehensive data inclusion in these models.

4.1 Algorithms

J48:

- Best Configuration: Using all features.
- ROC-AUC: 0.792
- Observations: Delivered consistent results and a good balance between interpretability and performance. When feature selection was applied using CfsSubsetEval, the performance slightly declined, suggesting J48 benefited from all available features.

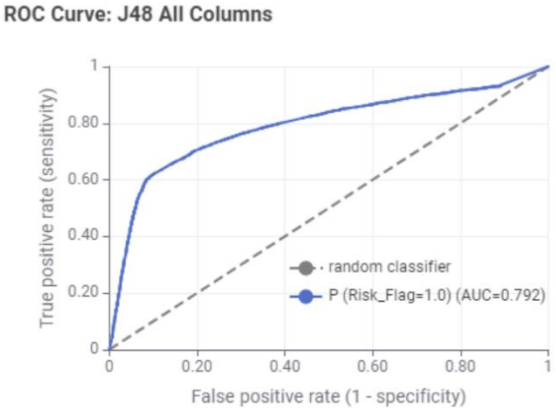


Figure 2. ROC Curve J48 with all features

Logistic Regression

- Best Configuration: The information gain method confirmed that all features contributed to the model and were thus included.
- ROC-AUC: 0.628
- Observations**:** Showed lower ROC-AUC compared to tree-based models.

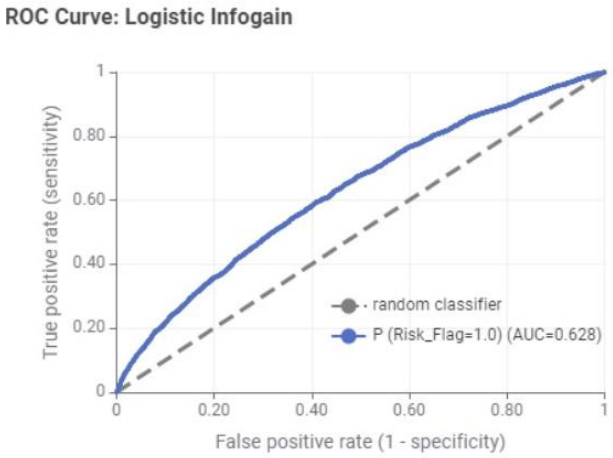


Figure 3. ROC Curve Logistic Regression with all features

Multi-Layer Perceptron (MLP)

- Best Configuration: Using all features.
- ROC-AUC: 0.642
- Observations: It could require more extensive tuning to improve performance. The network underperformed compared to tree-based models.

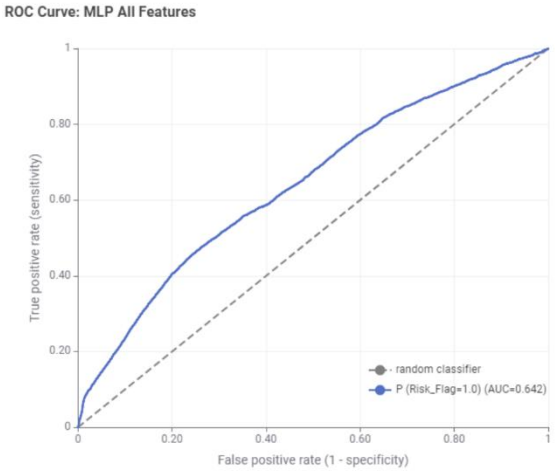


Figure 4. ROC Curve MLP with all features

Spegasos

- Best Configuration: Using all features.
- ROC-AUC: 0.5
- Observations: it didn't outperform random guessing

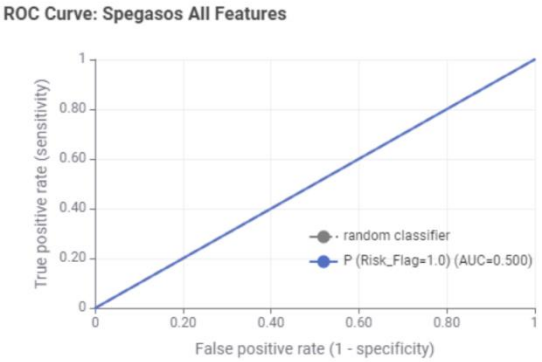


Figure 5. ROC Curve Spegasos with all features

Bayesian Net

- Best Configuration: Using all features.
- ROC-AUC: 0.624
- Observations: It seems better than others but it wasn't already the best one.

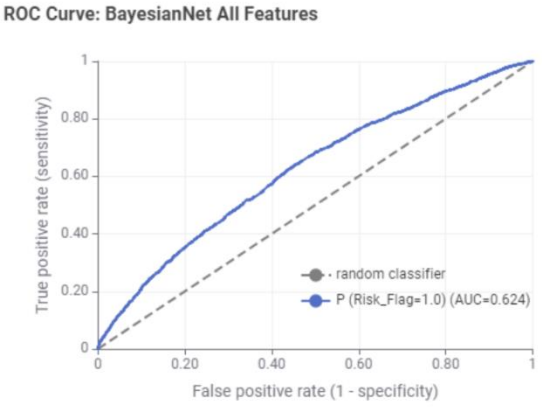


Figure 6. ROC Curve Bayesan Net with all features

Naive Bayes

- Best Configuration: Using all features.
- ROC-AUC: 0.616

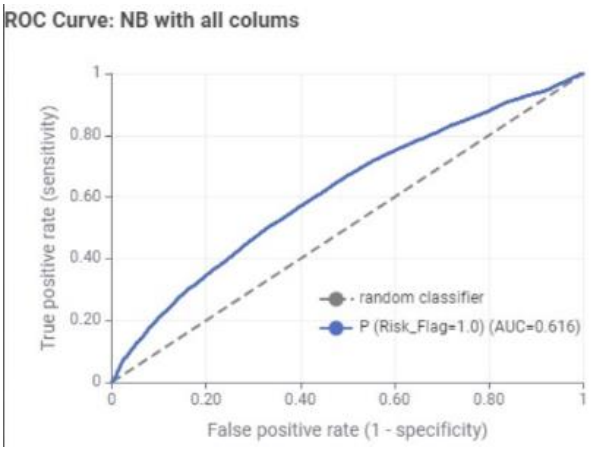


Figure 7. ROC Curve Naïve Bayes with all features

4.2 Random Forest

- Best Configuration: Using all features.
- ROC-AUC: 0.897 (test), 0.987 (training).

Initial tests with Random Forest yielded strong performance, achieving high ROC-AUC values, but significant overfitting was observed, as evidenced by the disparity between training (0.987) and test (0.897) scores. The primary cause of this overfitting was identified as the variables transformed *with K-Fold Encoding* (*City, State, and Profession*). These variables, while initially contributing to model performance, also introduced noise and increased complexity, exacerbating overfitting.

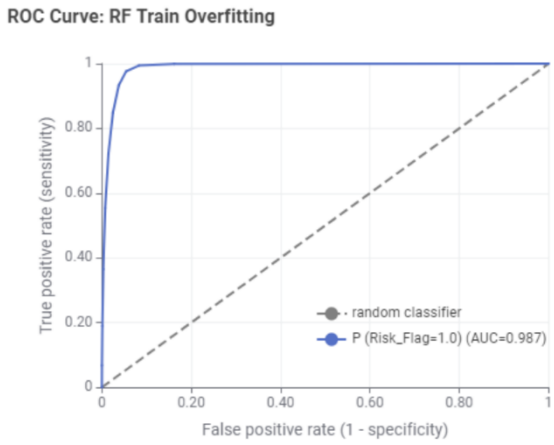


Figure 8. ROC Curve Random Forest Train with all features

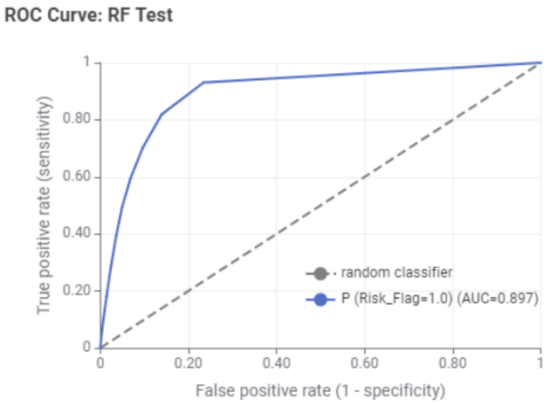


Figure 9. ROC Curve Random Forest Test with all features

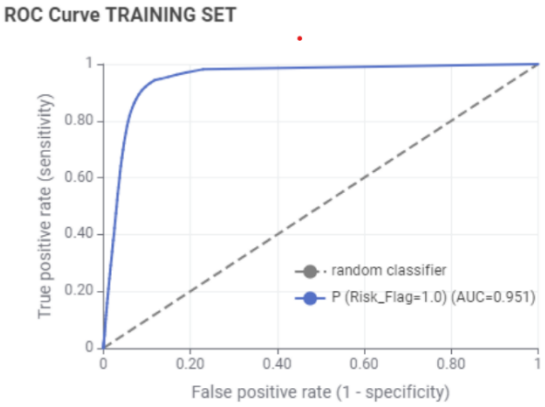
To address this, we tested alternative approaches for handling these columns. First of all, we tried to apply *OneHotEncoding***,** but unsuccessfully because the number of dummies was too high and KNIME crushed multiple times. We thought that a possible solution would be to group *CITY and STATE* based on socio-economic zone, but we didn't have enough knowledge to do that. Additionally, we can infer that the information about profession is contained in Income. Consequently, the decision was made to remove these three columns to simplify the dataset and reduce overfitting.

With the revised dataset, Random Forest remained the focus due to its strong baseline performance and capacity to handle complex relationships. We conducted extensive hyperparameter tuning, targeting parameters such as the number of trees, maximum tree depth, to achieve two objectives:

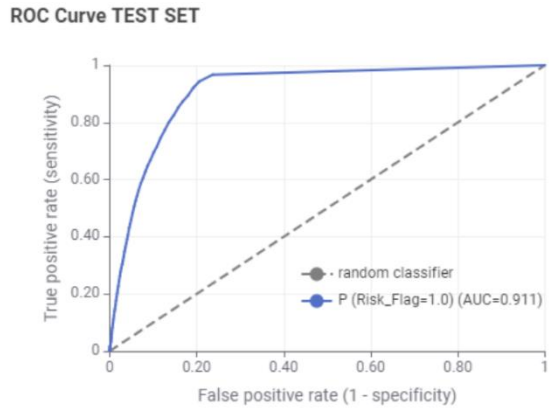
1. Enhance overall predictive power.
2. Reduce overfitting.

Specifically, the *Parameter Optimization Loop* node in KNIME was used. This node employs the Hill Climbing algorithm to search for the pair of parameters that maximizes the number of correct classifications. To ensure effective optimization, lower and upper boundaries for parameter values required by the very node were defined based on preliminary manual tests. These tests helped identify the most appropriate range for the algorithm to explore.

This process led to a better-balanced model, achieving a ROC-AUC of **0.947** on training and **0.903** on the test set (obtained using 5-fold cross-validation), effectively mitigating overfitting.



* **Figure 10.** ROC Curve Random Forest Test without *STATE, CITY* and *Profession*



*** Figure 11.** ROC Curve Random Forest Test without *STATE, CITY* and *Profession*

*(The numbers in these charts differ slightly from those mentioned earlier, as they reflect the model's performance before cross-validation. Notably, the values in the charts are higher than the cross-validated results, further underscoring the importance of cross-validation in ensuring reliable performance assessment.)

These findings emphasize the robustness of Random Forest and its ability to leverage a comprehensive feature set effectively. By addressing the challenges of overfitting and fine-tuning the model, we achieved a configuration that balanced performance and generalization, confirming Random Forest as a highly effective algorithm for this dataset.

5 Models interpretability

To try to understand the significance of the variables, we used the *AttributeClassifier* node with the *InfoGain* method. This analysis was conducted on both a Random Forest model with default parameters and the fine-tuned Random Forest model (optimized as described earlier). The results were nearly identical for both models, with both approaches consistently identifying the most significant variables as ***Current_Job_Years, Age*** *and* ***Experience.***

Interestingly, and somewhat counterintuitively, *Income* is not a primary variable in our model. However, the selected features are closely related to income, as they likely reflect applicants' financial stability and job security—factors crucial for risk assessment.

6 Conclusion

Throughout our analysis, the primary metric we focused on and optimized was the AUC. However, considering that we are dealing with an imbalanced problem, it is important to also pay attention to other performance metrics. In particular, we observed that our Random Forest model showed a very low Recall for the *Risk_Flag=1* class (0.09 after 5-fold cross-validation).

Recall measures how well the model identifies actual "risky" customers, so a low Recall for the *Risk_Flag=1* class indicates a high number of False Negatives compared to True Positives. In other words, we are classifying risky customers as non-risky, which poses a significant risk for the bank: we may be approving loans for individuals who are unlikely to repay them.

One possible alternative is to prioritize models that offer better Recall. The **J48** decision tree algorithm is one such model. For this algorithm, we also performed hyperparameter optimization using the *Parameter Optimization Loop* node. This time, the key parameters were Confidence Factor (a value between 0 and 1 used to calculate the

confidence interval for error estimation at each node in the tree) and *MinNumObjects* (the minimum number of objects required in a node before further splitting). After setting the optimal parameters, we achieved a **Recall** of 0.473 (cross-validated with k=5) for the *Risk_Flag=1* class, which is much higher than that of the Random Forest model. However, as a trade-off, the **ROC-AUC** dropped to 0.83, losing about 8 points compared to Random Forest.

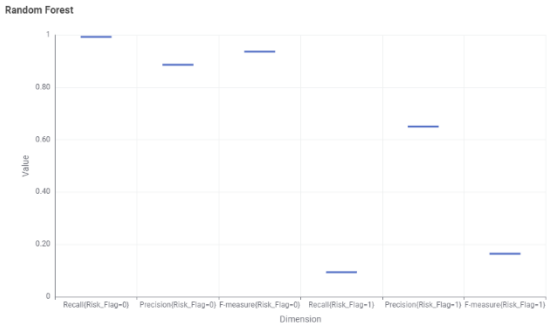


Figure 12. Random Forest Recall, Precision and F-measure cross-validate values

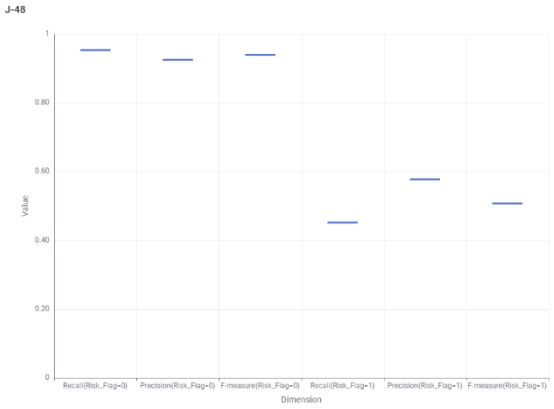


Figure 13. J48 Recall, Precision and F-measure cross-validate values

Ultimately, we chose to keep the Random Forest model to maximize the AUC, but we are aware that other approaches, such as focusing on Recall, could also be valid strategies depending on the specific business priorities.

The final Random Forest model, fine-tuned to mitigate overfitting, successfully leverages the dataset's comprehensive feature set to deliver robust predictions. This model is particularly suited to the bank's objective of achieving accurate and balanced risk assessments, making it the optimal choice for deployment in a real-world loan approval system.

References

Kaggle, Loan approval dataset from: *https://www.kaggle.com/datasets/rohit265/loan-approval-dataset*

Random Forest Breiman, L. (2001 from: *https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf*

Guyon, I. and Elisseeff (2003) from: https://jmlr.org/papers/volume3/guyon03a/guyon03a.pdf

https://www.mdpi.com/2227-9717/9/10/1713