Automation of the loan eligibility process based on customer detail

by

Sofia Borchardt, Arushi, Ana Ccarita, Uiara Valente, Gayatri Dandawate, Anshul Ghormade

The Quest for Loan Eligibility: Exploring the Depths of Customer Details

RWTH University,
Aachen

Febraury 2023

Contents

1 Introduction to the Loan Prediction Mission

1.1 The Challenge: Defining the Problem Statement

The problem at hand is to develop a machine learning model that can predict whether or not a customer will receive a loan. The model will be trained on a dataset containing customer information such as demographics, purchase history and among others.

1.2 Data Analysis: Gathering and Assessing Vital Information

The dataset used for training and evaluating the model contains 614 observations corresponding to different customers, which includes the following information:

- Loan_ID : Unique Loan ID
- Gender : Male or Female
- Married : Applicant married (Y/N)
- Dependents : Number of dependents
- Education : Applicant Education (Graduate/ Under Graduate)
- Self_Employed : Self employed (Y/N)
- ApplicantIncome : Applicant income
- CoapplicantIncome : Coapplicant income
- LoanAmount : Loan amount in thousands of dollars
- Loan_Amount_Term : Term of loan in months
- Credit_History : Credit history meets guidelines yes or no
- Property_Area : Urban or Semi Urban or Rural
- *Loan_Status* : Loan approved (Y/N), this is the dependent variable.

1.3 Methodology Overview: Charting Our Course

To tackle this problem, supervised learning techniques will be used to train a binary classification model. Several functions have been developed and implemented, which are used to navigate and iterate through the problem. The first step is to perform an exploratory analysis to understand the variables and their correlation with the target variable. Following this, different steps are taken to train and test machine learning models using the pipeline concept, which includes preprocessing (Missing values treatment, onehot encoding and scaling), feature selection, cross-validation, and hyperparameter optimization to obtain the best results. Finally, different attempts will be made to improve the outcomes, such as creating new features.

1.4 Model Selection: Deciding Our Path

The performance of several commonly used binary classification algorithms such as Logistic Regression (LR), Random Forest (RF), and Support Vector Machines (SVM) will be compared. Hyperparameters tuning technique such as Grid search will also be used to optimize the model performance.

1.5 Results and Conclusion: Charting Our Success

The final results of the model performance will be presented, and the implications of the findings will be discussed. The potential limitations of the analysis will be discussed as well as the areas for future growth, will be suggested.

2 The Methodology of Loan Prediction

Overall, the methodology used in this code includes:

- Exploratory Data Analysis (EDA).
- Modeling of the data.
- Evaluation and further possibilities.

2.1 Exploratory Data Analysis: Scanning the Data

2.1.1 Exploring the Variables: Purifying the Signal

To gain a comprehensive understanding of the data, we conducted exploratory data analysis, which involved analyzing categorical and numerical variables separately and evaluating outliers.

The first step involved setting the data types based on the nature (discrete or continuous) of the variables, as follows:

- 'Gender': 'category'
- 'Married': 'category'
- 'Dependents': 'category'
- 'Dependents': 'category'
- 'Education': 'category'
- 'Self_Employed': 'category'
- 'Credit_History': 'category'
- 'Property_Area': 'category'
- 'Loan_Amount_Term': 'category'
- 'ApplicantIncome': 'float64'
- 'CoapplicantIncome': 'float64'
- 'LoanAmount': 'float64'
- 'Loan_Status': 'category'

In order to explore the numerical variables, two functions are created:

- *UVA_numeric* which conducts a descriptive statistical analysis including minimum, maximum, range, mean, median, standard deviation, skewness, and kurtosis. The function runs a loop for each variable, calculates its descriptives, and plots its KDE (Kernel Density Estimation) with all the information. A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.
- *Normality_check* which checks the normality of the distribution of the variables in the input data using a Q-Q plot. A Q-Q plot is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight. If the scatter plot is not a straight line then the distribution is not normal and we have a skewed distribution with outliers.

On the other side, for the exploration of categorical variables two other functions were created:

- *UVA_category* calculates the frequency of occurrence of each category and plots a bar chart showing both: the percentage that each category represents out of the total observations and the count of each category.
- *BVA_categorical_plot* does a bi-variate analysis of two categorical variables and forms a cross-tabulation table between them. The chisquared test is performed on this table to check if the relationship between the two variables is statistically significant. In this case, statistically, significance was defined using a p-value equal to 0.05. It is used for hypothesis testing. Then, a grouped count plot is plotted showing the distribution of the target variable for each category of some of the categorical variables. Additionally, a per cent stacked bar plot shows the distribution of the target variable for each category of some of the categorical variables as a percentage of the total number of observations.

2.1.2 Missing and Outlier Values: Identifying the Unknown

The treatment of the NAs has been carried out in a simple way, using the median for the continuous variables and the mode (e.g. the most frequent value) for the categorical variables. This will be part of the prepossessing pipeline, described in the following sections.

To analyze outliers, first of all, it is conducted a qualitative analysis of the values and checked if there were unrealistic or impossible values in the data set (e.g: a negative period of the loan). After concluding that all values were valid, since none of the continuous numerical variables presented a normal distribution, outliers were selected using the IQR (Interquartile Range). The IQR was based on the fact that most of the data in a sample are expected to be within the range of Q1 - 1.5 * IQR to Q3 + 1.5 * IQR.

Q1 and Q3 represent the first and third quartiles, respectively, and IQR is the interquartile range, which is the difference between Q3 and Q1. Any data point that falls outside this range could be considered an outlier. The outliers treatment is checked with the functions:

- *UVA_outlier*: performs univariate analysis on the continuous variables. For each variable in the group, the function calculates its descriptive statistics (i.e., median, interquartile range, and whiskers), identifies the number of outliers based on the calculated whiskers, and either plot a boxplot with outliers or replaces the outliers with the corresponding whisker value and plots a boxplot without outliers.
- *Outliers_Distribution*: stores the outliers in a new dataframe. It then groups the outliers by 'Loan_Status' and calculates the percentage of outliers that fall into each category.

Once the subset of outliers was defined, the distribution of the target variable was analyzed within this subset and compared with the rest of the dataset to evaluate the behaviour of the outliers towards the target variable to better understand the added value of this subset of the observations.

2.2 Modeling of the data: Navigating the Data, testing our hypothesis, and optimizing our course.

This chapter is the heart of this study, which is based on two main functions (*Preprocessing_Pipeline*, *ML_Models*). These functions condense several essential concepts and tools for predicting loan eligibility.

Pipelines are used to organize and automate the data processing and modeling steps in a consistent and reproducible manner. This is crucial because it helps prevent data leakage, which can occur when information from the test set leaks into the training set during the modeling process, leading to overly optimistic performance estimates. By using pipelines, the data processing and modeling steps are exclusively applied to the training data, and the test data is kept separate until the final evaluation. This approach ensures that the performance estimates are more realistic and can be trusted for decision-making.

The *Preprocessing_Pipeline* function is a data transformation pipeline that inputs data for the missing values, performs one-hot encoding on categorical variables, and scales on numerical variables using MinMaxScaler. The pipeline is defined using the ColumnTransformer and Pipeline classes from scikitlearn. The ColumnTransformer applies different transformers to different columns of the data, while the Pipeline groups these transformers together into a single model. Since the data treatment for numerical and categorical values is not exactly the same, ColumnTransformer was conducted separately for numerical and categorical values inside this pipeline.

Onehot encoding is used to convert categorical variables into numerical values. Each category of a categorical variable is transformed into a new binary column, with a value of 1 indicating the presence of that category in the original variable and 0 otherwise. This encoding allows the machine learning algorithm to understand the categorical variable as numerical and avoids the bias that could be introduced by considering the categories as ordered values.

MinMaxScaler scales the feature values to a specific range, usually between 0 and 1. It scales the values by subtracting the minimum value of the feature and dividing by the range (maximum value - minimum value). Logistic regression and SVC are sensitive to the scale of the input features because these models assume that the input features are on the same scale, and weight each feature equally in the model.

The *Preprocessing_Pipeline* function will be used then as part of the pipeline in the function *ML_models.*

The *ML_Models* is designed to train the machine learning models using different feature selection methods and hyperparameter tuning with a cross-validation (CV) strategy. The function first separates the dataframe into features X and target y variables and splits them into training and testing sets using a ratio of 80% to 20% respectively. Then, it applies the preprocessing pipeline from the function *PreProcessing_Pipeline* to X_train. Next, it defines three models: SVC, LR and RF, and two feature selection methods: Ki2 and RFE. The number of features to consider is also defined (5,7). The function loops over each model, feature selection method, and number of features, creating a pipeline for feature selection and the model itself. A grid search is conducted with CV to optimize the hyperparameters of the model pipeline. Then, the model is evaluated using CV and its performance metrics are recorded. The final results are stored in a dataframe and returned as output, which is shown in the Results section.

It is important to notice that this function trains the models based on the training data and utilizes the k- fold crossvalidation (CV) strategy with a k=10. In optimal computational conditions, k should also be optimized as a hyperparameter. This is omitted in this project for practical reasons of restraint in the availability of computational resources and thus k is set to a common value in the literature that has shown a very good bias-variance trade-off with reasonable computational capability. In the book "Introduction to Statistical Learning with Application in R" by Garett James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, the impact of the number of folds in k-fold cross-validation is discussed in detail. The authors note that a higher value of n-splits generally leads to a more accurate estimation of the test error, but also results in longer computation time. Furthermore, increasing k can reduce bias in the estimation of the test error, but at the same time increase the variance due to the correlation introduced between the averaged errors.

Another point to highlight is that regularization hyperparameters are also considered for logistic regression where the penalties l1 and l2 are considered hyperparameters of the model. The penalty l1 corresponds to the lasso regularization method and the penalty l2 to the ridge regularization method.

Regarding the hyperparameter tunning, Grid search was selected because it is a simple and widely used technique that can work well for small or medium-sized datasets and models with a small number of hyperparameters as is this case.

The trained models are then tested using the test data points with the accuracy, F1 score, and confusion matrix calculated by comparing the predicted labels of X_test compared to the real labels "y_test" of the X_test data points. Before production, the model with the best performance should be retrained using the entire data set. In this case, this was omitted for simplicity and practical reasons.

2.2.1 Description of ML used Models

- SVC (Support Vector Classifier): SVC is a type of support vector machine (SVM) that performs binary classification by finding the hyperplane in the feature space that separates the two classes with the largest margin allowing for some miss-classifications.
- LR (Logistic Regression): Logistic Regression is a statistical model used for binary classification that predicts the probability of a binary outcome given a set of input features. It is an important algorithm to consider because it is simple and easy to implement, and it works well with a limited number of observations as is the case in this project.
- RF (Random Forest): Random Forest is an ensemble learning algorithm that combines multiple decision trees to form a more robust and accurate prediction. It is important to consider because it can handle both categorical and continuous input variables, it's easy to interpret, and it can handle non-linear boundaries and complex interactions between features.

Overall, the combination of SVC, LR, and RF covers a range of problem types and provides a good basis for evaluating the performance of different machine-learning algorithms on the given dataset.

2.2.2 Considered hyperparameters in the respective ML Models

For all the models mentioned above the following hyperparameters were considered and tuned:

- Logistic Regression (LR):
  - The C hyperparameter in Logistic Regression controls the regularization strength. A small C value will result in a wider margin and a more general model (underfitting), while a large C value will result in a narrower margin and a more complex model (overfitting).
  - The penalty hyperparameter in Logistic Regression specifies the type of regularization applied to the coefficients. The common penalties are 'l1' (Lasso) and 'l2' (Ridge), which add an L1 or L2 regularization term to the loss function, respectively. L1 regularization leads to sparse coefficients, meaning some of the features may be completely ignored by the model, while L2 regularization tries to spread out the coefficients' magnitude across all features.
- Random Forest (RF):
  - n_estimators: This hyperparameter represents the number of trees in the forest. Increasing the number of trees can lead to a decrease in variance, but also to an increase in bias and computing time. A typical value for n_estimators is 100.
  - max_depth: This hyperparameter controls the maximum depth of each tree in the forest. Deeper trees can lead to overfitting, while shallower trees can lead to underfitting. A common value for max_depth is between 3 and 11.
  - min_samples_split: This hyperparameter sets the minimum number of samples required to split an internal node. Setting this value higher can reduce overfitting but also decrease the model's ability to capture complex relationships. A common value for min_samples_split is 2.
  - min_samples_leaf: This hyperparameter sets the minimum number of samples required to be at a leaf node. Setting this value higher can reduce overfitting, but also decrease the model's ability to capture complex relationships. A common value for min_samples_leaf is 1.
  - max_features: This hyperparameter sets the number of features to consider when looking for the best split. Using auto means that the number of features to consider is equal to the square root of the total number of features, using sqrt means the number of features to consider is equal to the square root of the total number of features, and using log2 means the number of features to consider is equal to the logarithm of the total number of features. A common value for max_features is auto.
- Support Vector Classifier (SVC):

  - C: is the regularization parameter that determines the trade-off between achieving a low training error and a low testing error. A smaller C value allows some misclassification of the training data, while a larger C value aims to minimize the training error and increase the chances of overfitting.
  - gamma: is the kernel coefficient for 'rbf' (Radial basis function), 'poly' and 'sigmoid'. Larger values of gamma lead to a more complex model and smaller values of gamma lead to a simpler model.
  - kernel: is the type of kernel used in SVC. Some common types of kernels are 'linear', 'rbf' (Radial basis function), 'poly' (polynomial) and 'sigmoid'.

2.2.3 Description of the used performance measure methods

- The *accuracy* score is one of the most commonly used metrics for classification problems, as it measures the proportion of correct predictions made by the model. It is a simple and intuitive metric that is easy to interpret and compare across models. However, accuracy can be misleading if the classes are imbalanced, meaning that some classes have a much larger number of samples than others. In such cases, a model that always predicts the majority class may have high accuracy, but it may not be useful in practice.
- *F1 score* is a more balanced metric that takes into account both precision and recall. Precision measures the proportion of true positives among all positive predictions, while recall measures the proportion of true positives among all actual positive samples. The F1 score is the harmonic mean of precision and recall, and it is a good metric to use when the classes are imbalanced, which could be the case as has been seen in the EDA.
- The *confusion matrix* shows the number of true positives, true negatives, false positives, and false negatives for each class.

It is important to mention, that this problem has the particularity that not all errors are equally "expensive" for the client since a false positive (i.e: the bank gives a loan to someone that should not have receive it) is worse than a false negative (i.e: the bank denies a loan to someone that should have receive it). For this reason, other ways of measuring performance were included besides the accuracy of the classification. The main measures of performance that we are going to consider are the accuracy and the F1 score. These methods combined give a good performance indication for our particular business case. However the F1 score, a high or low F1 score does not give information of weather the recall or the precision are being high or low. For this reason a qualitative such as the confusion matrix is also considered.

2.3 Other steps: Looking for Alternative Approaches

Some features were manually created using domain knowledge with the goal of identifying valuable correlations with the target variable that would improve the performance of our model. These variables were added before running the dataframe through the function *ML_Models*. Given that no improvements were shown, it was decided to not include these variables in production.

The manually-created variables were created using transformations and combinations of the original variables shown below, and they aim to capture more non-linear relationships between the variables and the target variable. However, it is essential to keep in mind that creating too many new features can lead to overfitting, which can result in poor generalization performance on new data.

- 'LoanAmount_log': Taking the logarithm of the LoanAmount variable can help normalize its distribution and remove outliers. Logarithmic transformation helps in dealing with skewness in the data.
- 'ApplicantIncome_log': Similar to the first one, taking the logarithm of the ApplicantIncome can help normalize its distribution and remove outliers.
- 'Total_Income_log': Same as above, taking the logarithm of the Total_Income can help normalize its distribution and remove outliers.
- 'LoanAmountAppIncome': This variable represents the ratio between the logarithm of LoanAmount and the logarithm of ApplicantIncome. It may capture information about the loan affordability based on the applicant's income.
- 'LoanAmountTotalInc': This variable represents the ratio between the logarithm of LoanAmount and the logarithm of Total_Income. It may capture information about the loan affordability based on the total household income.
- 'LoanAmount_ratio': This variable represents the ratio between LoanAmount and Total_Income. It may capture information about the loan affordability based on the total household income.
- 'Income_education': This variable represents the product of the logarithm of ApplicantIncome and the Education variable. It may capture information about how education affects the applicant's income.
- 'Income_employed': This variable represents the product of the logarithm of ApplicantIncome and the Self_Employed variable. It may capture information about how self-employment affects the applicant's income.
- 'ApplicantIncome_log_sq': This variable represents the square of the logarithm of ApplicantIncome. It may capture non-linear relationships between the ApplicantIncome and the target variable.

- 'LoanAmount_monthly': This variable represents the LoanAmount divided by the Loan_Amount_Term. It provides information about the monthly repayment amount.

Another approach taken was to create several new variables automatically from the already given variables combining two or more variables to create a new feature through polynomial transformation (degree 2). After the addition of these variables, the function *ML_Models* was fed with this new dataframe. But again the results were not better than the ones described for the original dataframe (without the addition of new variables).

Finally to improve the predictive results, a simple neural network (NN) was tested to determine if the results could be improved with a more complex solution. The proposed NN was programmed with two hidden layers in the model, one with 128 neurons and a ReLU activation function and the other with 64 neurons and also a ReLU activation function. Another layer of the model includes a sigmoid activation function for binary classification. The model is compiled with the optimizer *adam*, the loss function (binary cross-trapping), and the same metrics used to evaluate the model (accuracy, f1, and confusion matrix). This NN did not improve the performance of the model and was therefore not considered to move to production without further analysis.

Other strategies could be discussed like domain knowledge by adding features that are known to be related to the target variable, for example, the loan purpose. Another possibility would be to collect additional data, either new observations and/or new variables (i.e. more demographic information about the applicant).

3 Results and Discussion of the Loan Prediction Mission

In this section results are discussed considering the following parts:

- Exploratory Data Analysis (EDA).
- Modeling of the data.
- Other steps.

3.1 Exploratory Data Analysis: The Data Tells Its Story

Below are shown the output of the functions *UVA_numeric* and *Normality_check*:

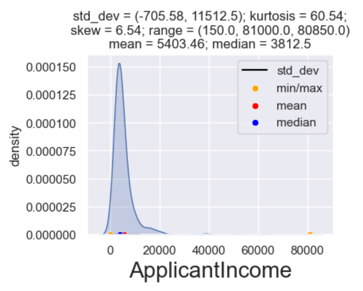


Figure 1: Application Income distribution

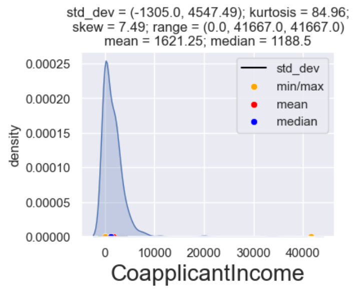


Figure 2: Co-application Income distribution

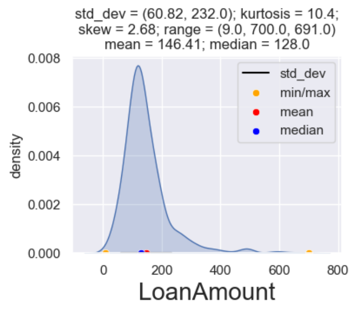


Figure 3: Loan Amount distribution

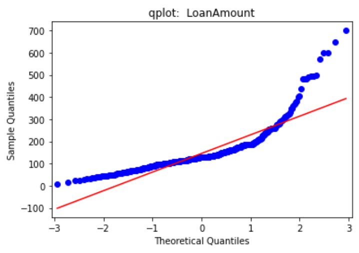


Figure 4: QQ Plot for the variable *LoanAmount*

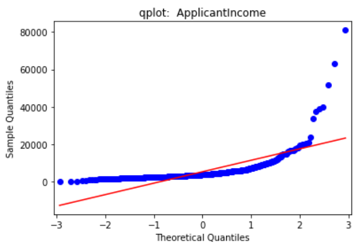


Figure 5: QQ Plot for the variable *ApplicantIncome*

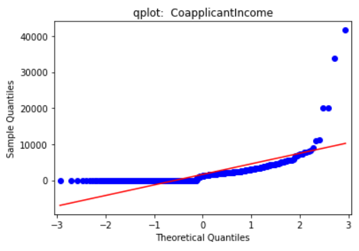


Figure 6: QQ Plot for the variable *CoapplicantIncome*

Basically, we can conclude that neither of the three numerical values is following a normal distribution and some potential outliers are already visualized at this point.

Considering the IQR described in the section 2, 79 observations were considered outliers. An example of the variable Loan Amount can be seen below.

The distribution of the subset of outliers was similar to the original dataset regarding the target variable (63,4 % YES – 36,4 % NO). This would indicate that removing them would not affect the training of ML in the future. However, the subset of outliers

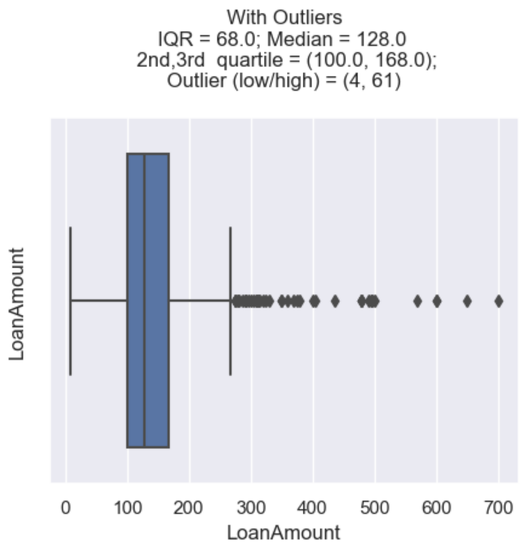


Figure 7: Boxplot for variable *LoanAmount* with outliers

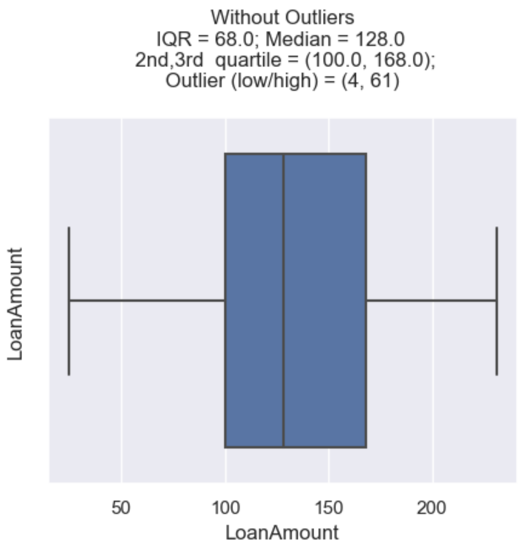


Figure 8: Boxplot for variable *LoanAmount* without outliers

represents more than 10% of the total data set. Considering all factors it was decided that the added informative value that the outliers could provide could enrich the model and improve the prediction of our model and thus have a better performance for the client's use case. For this reason, the outliers were not eliminated from the data.

The following distribution was distinguished after analyzing categorical variables in the original dataset:

- 81% are males.
- 78% are graduated.
- 86% are non-self-employed.
- 85% asked for a loan of 30 years.
- 84% have a credit history.
- *69% are provided with the loan and 31% no.*

Below is shown some examples of the output of the functions $UVA_categoryand$

-

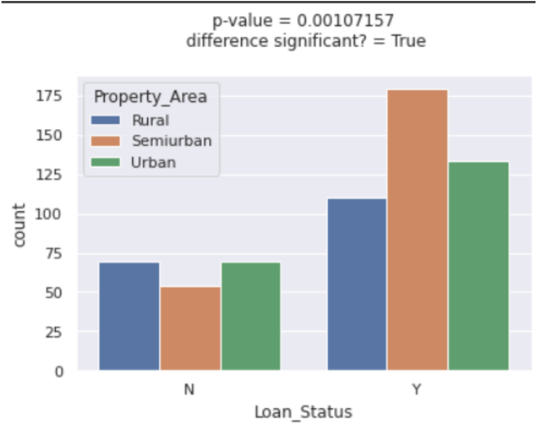


Figure 9: Histogram Plot for Property Area and Loan Status

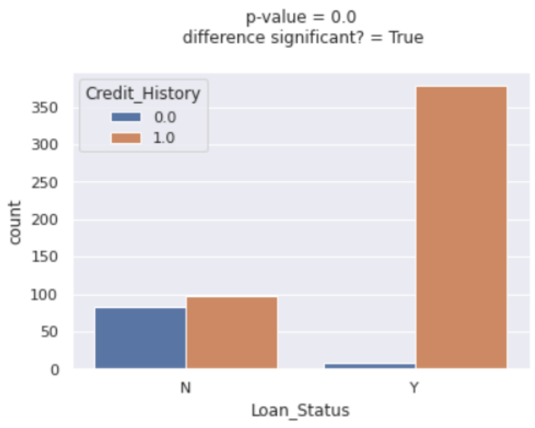
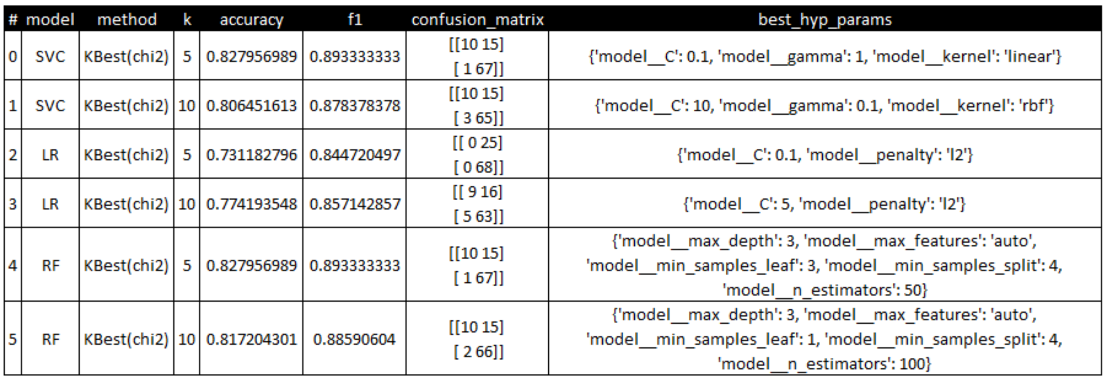


Figure 10: Histogram Plot for Credit History and Loan Status

3.2 Model Selection: Choosing Our Best Path

As mentioned in the previous section, the results of the code indicated the performance of the different machine-learning models that were evaluated. The evaluation metric used in this case is accuracy, F1 score and confusion matrix, which measures the proportion of correctly classified instances in the dataset.



#	model	method	k	accuracy	f1	confusion_matrix	best_hyp_params
0	SVC	KBest(chi2)	5	0.827956989	0.893333333	[[10 15] [1 67]]	{'model__C': 0.1, 'model__gamma': 1, 'model__kernel': 'linear'}
1	SVC	KBest(chi2)	10	0.806451613	0.878378378	[[10 15] [3 65]]	{'model__C': 10, 'model__gamma': 0.1, 'model__kernel': 'rbf'}
2	LR	KBest(chi2)	5	0.731182796	0.844720497	[[0 25] [0 68]]	{'model__C': 0.1, 'model__penalty': 'l2'}
3	LR	KBest(chi2)	10	0.774193548	0.857142857	[[9 16] [5 63]]	{'model__C': 5, 'model__penalty': 'l2'}
4	RF	KBest(chi2)	5	0.827956989	0.893333333	[[10 15] [1 67]]	{'model__max_depth': 3, 'model__max_features': 'auto', 'model__min_samples_leaf': 3, 'model__min_samples_split': 4, 'model__n_estimators': 50}
5	RF	KBest(chi2)	10	0.817204301	0.88590604	[[10 15] [2 66]]	{'model__max_depth': 3, 'model__max_features': 'auto', 'model__min_samples_leaf': 1, 'model__min_samples_split': 4, 'model__n_estimators': 100}

Figure 11: Results

The results show that all models achieved relatively high accuracy, ranging from 73% to 83%. The F1 score is also high, ranging from 84% to 89%, indicating a good balance between precision and recall.

Looking at the confusion matrix, we can see that all models correctly classified most of the negative cases in the observations, but had more difficulty with positive cases in the observations. The models tended to have more false negatives than false positives, which means that some customers were denied for the loan requested with the model which in reality were not denied.

Among the models, the SVC model with k=5 and the RF model with k=5 achieved the highest accuracy and F1 score, and had similar confusion matrices with relatively few false positives and false negatives. The best hyperparameters found for these models are also reported.

Overall, the results suggest that the SVC and RF models with KBest feature selection using chi2 test with k=5 are good options for predicting the loan status. Considering these results it is recommended to use SVC with k=5 for feature selection using chi square given the lower computational capability that is required. It is worth to mention that the grid of hyperparameters was of a limited size in the used models to accelerate the computation.

Further evaluation and testing may be necessary to determine the robustness of these models. As mentioned before it would be also recommendable to train the winning model, in this case SVC or RF using the entire dataset before deploying the model in production.

The code was originally prepared to also run feature selection with RFE (recursive feature elimination) and with more hyperparameters, but due to a lack of computing resources, it was run only with the chi2 feature selection method and with fewer hyperparameters, especially for the RF model. In addition, more values of K could be included for feature selection. As mentioned before, the k of k-fold cross validation could also be optimize if the needs of the business case are more restrictive with regards to acceptable errors.

3.3 Other steps: In the search to Enhance Performance

In this project, the motivation for creating new features was to improve the performance of the machine-learning models. However, it should be noted that creating new features is a time-consuming and resource-intensive process, especially in real-world scenarios where data collection can be expensive. Furthermore, although several new variables were created through polynomial transformations and combining multiple variables, no significant improvement in the model's performance was observed. Therefore is not recommended to repeat this procedure should someone face the proposed ML problem in the future.

It may be necessary to explore alternative techniques or to try different parameters because the feature creation techniques used were, according to the results, not the most appropriate for the data.

Regarding to the NN, the model evaluation was performed with a different number of epochs (10, 20, 30) and batch sizes (32, 64, 128), however, the results did not exceed 72% accuracy. Many other things could be explored in neural networks like that different layers could produce varied degrees of precision, but this is not in the scope of this work, for our case, it seems that the problem may be more suitable for traditional ML models than for NN models.

4 The Conclusion of the Loan Prediction Mission

4.1 Recap of Our Journey: The Success of Our Mission

In general, the code effectively evaluates different machine learning models and determines the best-performing model based on accuracy. The results obtained from these steps show that the model is able to make accurate predictions and that the choice of techniques used in the cleaning, transformation, feature selection, and hyperparameterisation steps has a significant impact on the performance of the model.

4.2 Final Thoughts: Charting the Path for Future Predictive Adventures.

As we conclude this loan prediction mission, it is important to reflect on the knowledge and skills we have acquired along the way. We have gained a deeper understanding of data analysis and machine learning techniques, which will prove useful in future predictive adventures.

Another key takeaway is the importance of trying out various machine learning algorithms to identify the best-performing one. As we have seen, different algorithms have different strengths and weaknesses, and it is crucial to evaluate them all to find the most accurate model.

Our success in this project has shown us the power of machine learning and the importance of careful selection and tuning of techniques. We can apply this knowledge to other domains, such as fraud detection, customer segmentation, and sentiment analysis. We can also explore new frontiers, such as deep learning and reinforcement learning, to improve the accuracy and robustness of our models. Overall, we are confident that our journey has paved the way for new and exciting possibilities in predictive analytics.

List of Figures