By: Igord Diamandi

Loan Approval Prediction  
Using Machine Learning

Contents

[0](#_Toc175755063)

[Introduction 0](#_Toc175755064)

[Data Overview 0](#_Toc175755065)

[General information 0](#_Toc175755066)

[Target Variable Description 0](#_Toc175755067)

[Project stages and approach 0](#_Toc175755068)

[Data preparation 0](#_Toc175755069)

[Exploratory Data Analysis (EDA) 0](#_Toc175755070)

[Features Evaluation 0](#_Toc175755071)

[Model selection 0](#_Toc175755072)

[Model fine tuning 0](#_Toc175755073)

[Implementation 0](#_Toc175755074)

[EDA: Empty values elimination 0](#_Toc175755075)

[Outliers handling 0](#_Toc175755076)

[Feature Evaluation 0](#_Toc175755077)

[Model Comparison 0](#_Toc175755078)

[XGBoost Model Fine Tunning 0](#_Toc175755079)

[Grid search 0](#_Toc175755080)

[Model Performance After Fine-Tuning 0](#_Toc175755081)

[Model usage 0](#_Toc175755082)

# Introduction

This project leverages historical loan approval data to develop a machine learning model capable of predicting the likelihood of future loan approvals. This model aims to provide accurate predictions that can support decision-making processes in financial institutions, by analysing patterns and trends embedded within past loan applications.

The dataset used as the foundation for this project contains records of previous loan applications, including details about applicant demographics, financial histories, loan parameters and the outcomes of their loan requests. Through careful examination and preparation of this data, key features have been identified and engineered to serve as inputs for the predictive model.

In the following sections, this document will provide a thorough overview of the project, beginning with a detailed description of the data and the steps taken to prepare it for analysis. We will then define the target variable, outline the design of the machine learning model, and discuss the methodologies employed in its implementation. The document concludes with a comparison of model performances and the selection of the final model for deployment.

By systematically addressing each of these components, this document aims to provide a clear and comprehensive understanding of the project’s scope, objectives, and outcomes.

# Data Overview

## General information

**Dataset Overview**

The loan approval dataset consists of 13 fields, each providing critical information about the loan application and the applicant. Below is a detailed description of each field:

1. **Loan\_ID (object)**  
   This is a unique identifier for each loan application. Since it serves solely as an identification field with no predictive value, it will be excluded from the training dataset.
2. **Gender (object)**  
   This textual field indicates the gender of the applicant. It captures whether the applicant is male or female. Will be converted to binary categorical during the preprocessing.
3. **Marital Status (Married) (object)**  
   Another textual categorical field, this one reflects the marital status of the applicant, indicating whether the applicant is married or not. Will be converted to binary categorical during the preprocessing.
4. **Dependents (object)**  
   This textual categorical field specifies the number of dependents the applicant has. The values range from 0, 1, 2, to '3+' dependents.
5. **Education (object)**  
   A textual field that indicates whether the applicant is a graduate or not. This field will be transformed into a binary categorical variable during the data preparation stage.
6. **Self\_Employed (object)**  
   This textual field denotes whether the applicant is self-employed. Like the Education field, it will be converted into a binary categorical variable during preprocessing.
7. **ApplicantIncome (int64)**  
   A continuous numerical field (int64) representing the applicant’s income. This field will be used to assess the financial capacity of the applicant.
8. **CoapplicantIncome (float64)**  
   This is another continuous numerical field (float) that captures the income of the co-applicant, if any. It complements the applicant's income in determining the total financial standing.
9. **LoanAmount (float64)**  
   A continuous numerical field that indicates the amount of loan requested by the applicant. This field plays a crucial role in determining the loan approval outcome.
10. **Loan\_Term\_Amount (float64)**  
    This categorical numerical field represents the term of the loan in months. The values will be grouped during data preparation to facilitate analysis.
11. **Credit\_History (float64)**  
    A numerical field that records the applicant's credit history. It contains binary values (1.0 and 0.0) that will be converted into a categorical field in later stages of the project.
12. **Property\_Area (object)**  
    A textual categorical field that describes the area type (Urban, Semi-Urban, Rural) of the property for which the loan is being requested.
13. **Loan\_Status (object)**  
    This is the target variable for the model, represented as a textual categorical field. It has two possible values: "Y" for approved loans and "N" for rejected loans. Will be converted to binary categorical during the preprocessing.

There are total of 614 rows in the data set.

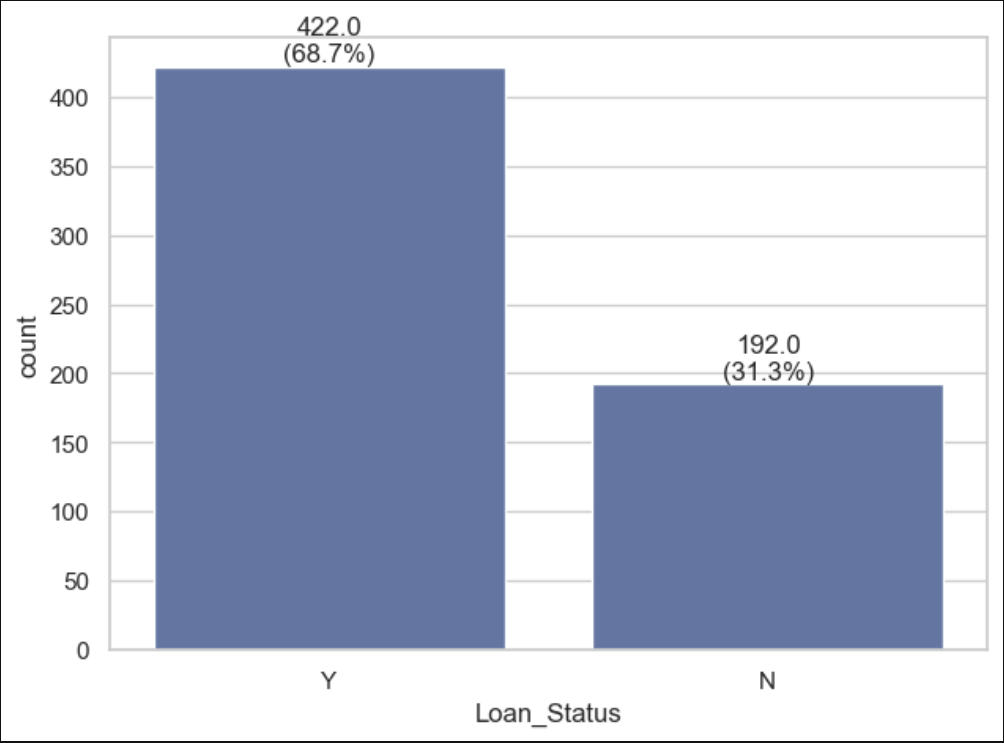
## Target Variable Description

The primary objective of this project is to develop a predictive model that can accurately determine the approval status of loan applications. The target variable for this model is the LoanStatus field, which captures whether a loan was approved or not. This field is a binary categorical variable, indicating either an approval or a rejection, and is crucial for training the machine learning model.

The LoanStatus field is encoded with two possible values:

* **Approved (Y)**: Indicates that the loan application was successful.
* **Rejected (N)**: Indicates that the loan application was not successful.

The field is well-prepared for modelling, with no missing values, ensuring that all records in the dataset contribute to the learning process. Additionally, the distribution of the LoanStatus field is balanced, which is advantageous as it prevents the model from being biased towards one outcome. Below is the distribution of the values within this field:



# Project stages and approach

## Data preparation

In this stage, we will thoroughly examine the initial dataset to identify and address any apparent errors.   
The outcome of this stage is a refined flat file, primed for Exploratory Data Analysis (EDA), data cleansing, and outlier handling in subsequent phases.   
This process is documented in the DataPrep.ipynb Jupyter notebook.

## Exploratory Data Analysis (EDA)

This stage is dedicated to an in-depth exploration of the dataset, with the goal of uncovering trends, distributions, correlations, outliers, and other key characteristics.   
During this phase, we will also determine the appropriate strategies for handling missing values and outliers.

To conduct EDA, the following libraries will be utilized:

* **AutoViz v0.1.905** in combination with **Seaborn** and **Matplotlib** for comprehensive data analysis and visual representation.
* **Missingno** for visualizing and analysing patterns in missing data.
* **LabelEncoder** for encoding categorical (non-numerical) fields into a numerical format suitable for modelling.

Regarding missing values, our primary approach will be to impute them using existing data or, if necessary, external data sources.   
For cases where logical imputation is not feasible, we will employ imputation models such as KNN or MICE to fill in the gaps.

For handling outliers, we will assess the impact of outlier reduction on key metrics such as distribution and correlation.   
This evaluation will guide our decision on whether outlier treatment is appropriate for each field where outliers are detected.

The outcome of this stage will be a clean, well-prepared dataset, ready for model training.

This stage is represented by the EDA\_and\_cleansing.ipynb Jupyter notebook.

## Features Evaluation

This stage focuses on assessing the importance of various features (fields) for model training.   
To determine which features are essential and which can be excluded, we will employ the following models:

* Lasso
* SVM
* Gradient Boost
* Random Forest
* Elastic Net
* Ridge
* RFE (Recursive Feature Elimination)

Given the potential time-consuming nature of this process, the evaluator code will be run in a multi-threaded manner to optimize efficiency.   
This stage is documented in the FeatureEvaluator.ipynb Jupyter notebook.

## Model selection

During this stage, various models will be evaluated and compared based on several key performance metrics,   
including accuracy, precision, F1 score, recall, AUC, and log-loss.

By analysing these metrics, we will identify the model that best meets our criteria for advancing to the fine-tuning stage.

Following models will be included in the comparison:

* Logistic Regression
* ADA Boosd
* XGBoost
* GBM
* Random Forest
* SVC

This stage is documented by the ModelComparison.ipynb Jupyter notebook.

## Model fine tuning

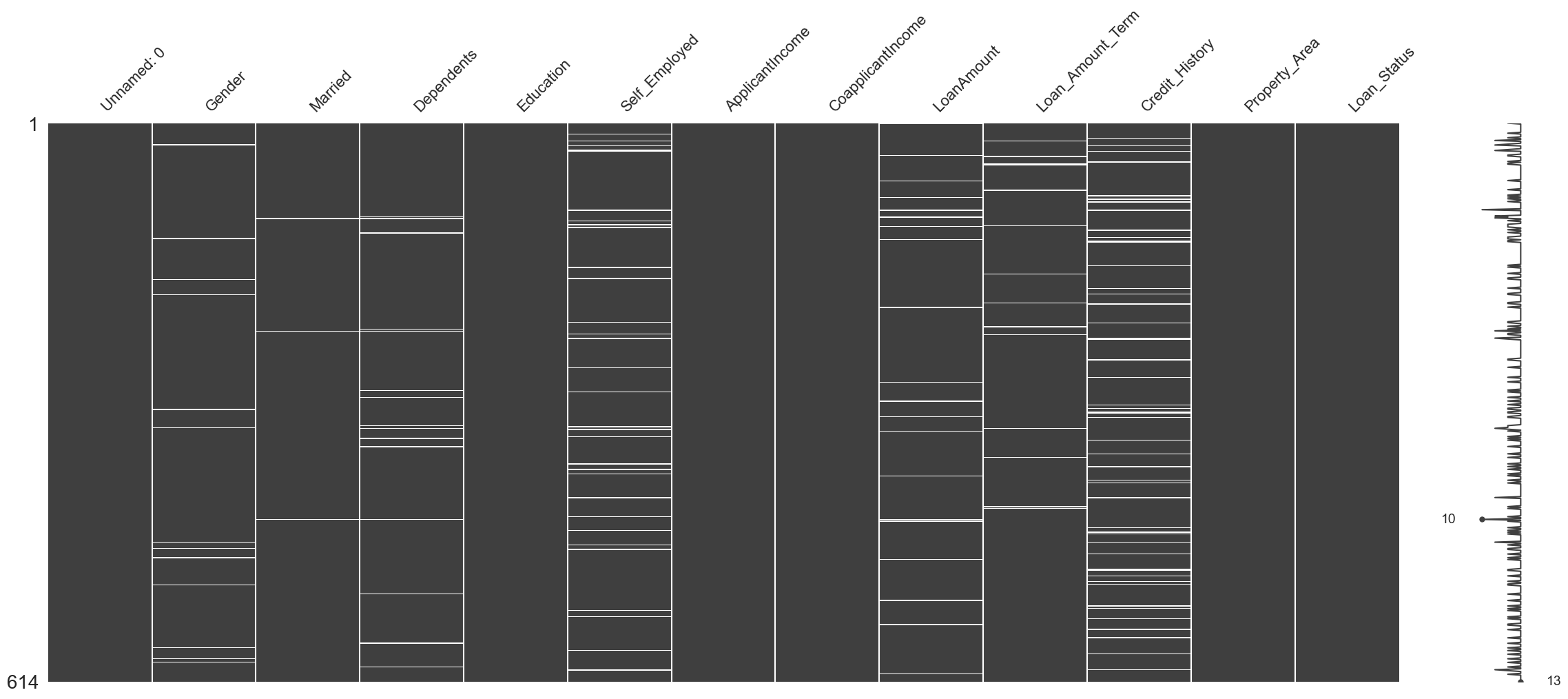
In this final stage, we will determine the optimal set of hyperparameters for the model through Grid Search, ensuring the best possible performance before deployment.

It’s represented by the ModelFineTuning.ipynb Jupyter notebook.

# Implementation

## EDA: Empty values elimination

Following snapshot was created using MSNO library, it is illustrating the empty values state in each of the data frame’s fields:



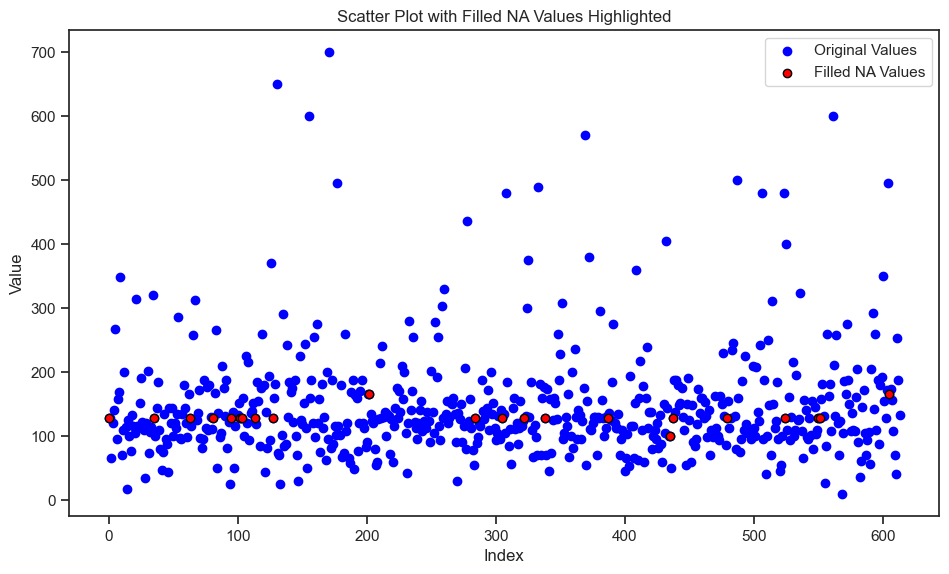
Numerical representation:

|  |  |  |
| --- | --- | --- |
| **Column** | **NaN Count** | **NaN Percentage** |
| Gender | 13 | 2.117264 |
| Married | 3 | 0.488599 |
| Dependents | 15 | 2.442997 |
| Self\_Employed | 32 | 5.211726 |
| LoanAmount | 22 | 3.583062 |
| Loan\_Amount\_Term | 14 | 2.28013 |
| Credit\_History | 50 | 8.143322 |

Columns with empty values are split into 2 groups:

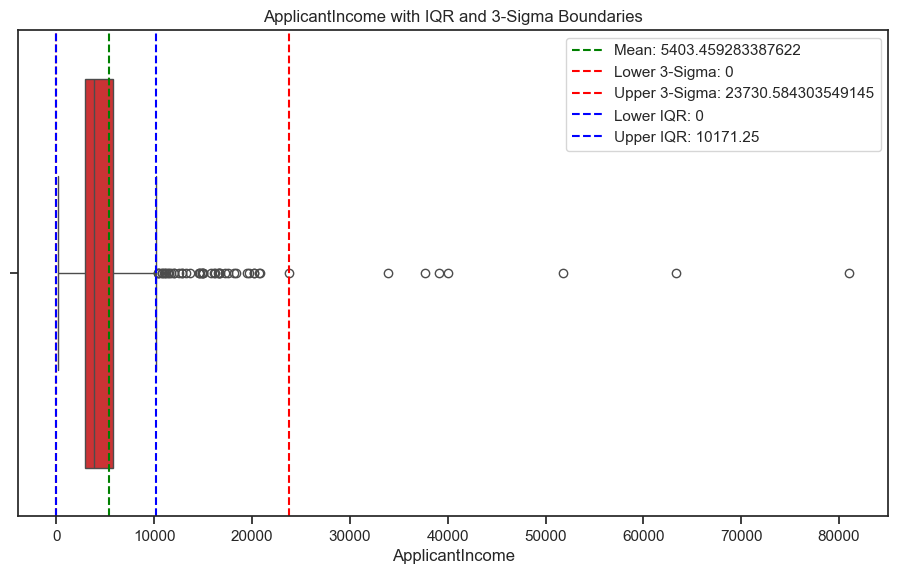
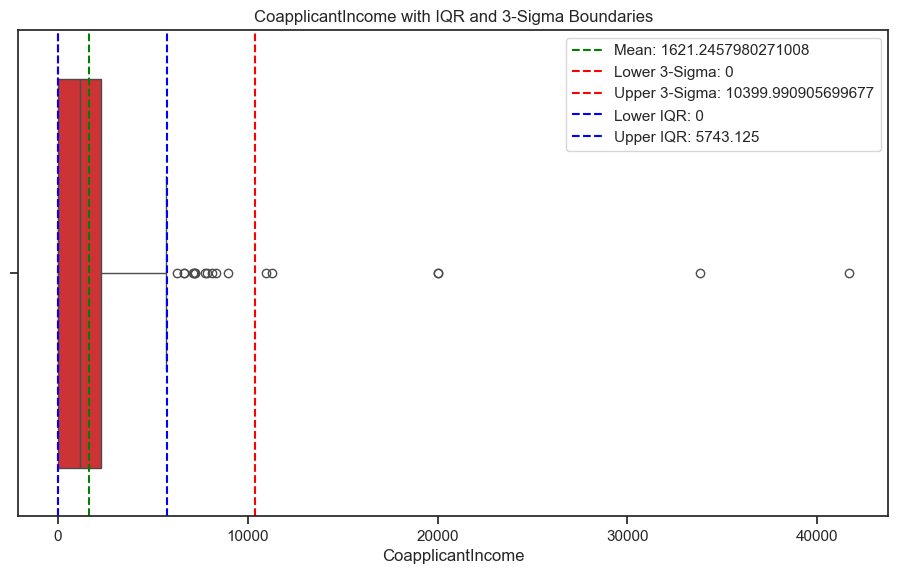
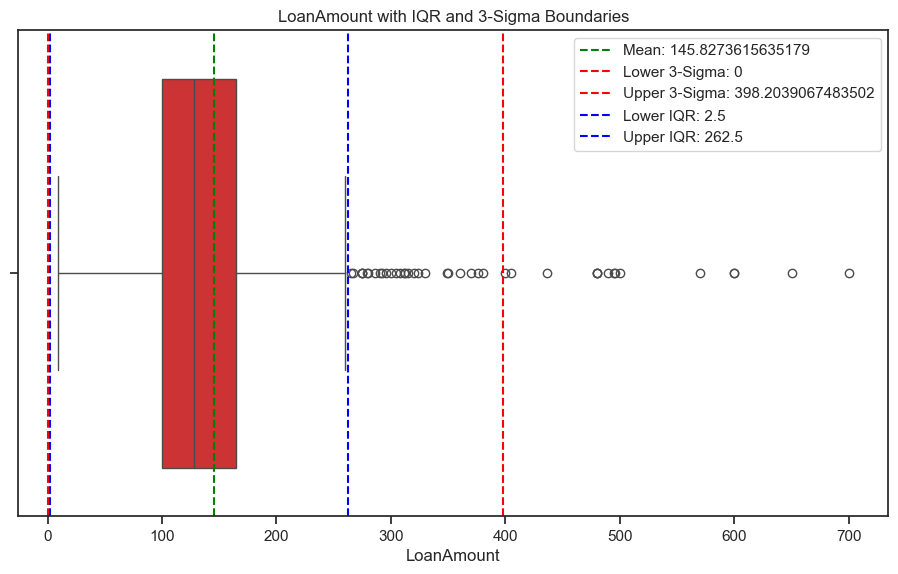
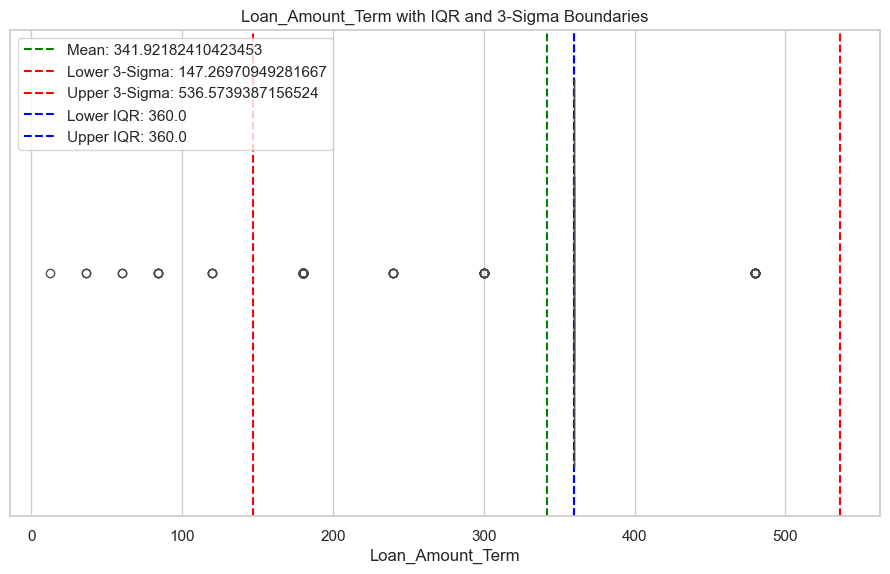
1. Personal data columns such as Gender, Married or Dependents.

2. Financial data columns, such as LoanAmount and Loan\_Amount\_Term.

There's no additional source of information which may help fill the empty values.   
There's also no way to conclude the missing data using an external source, as the data frame does not provide an identification of applicants or loan provider.  
This way the best possible approach for taking care of empty values is to use an imputation model such as KNN,   
here’s a graphical example of how KNN model handled empty values imputation for the LoanAmount field:  


## Outliers handling

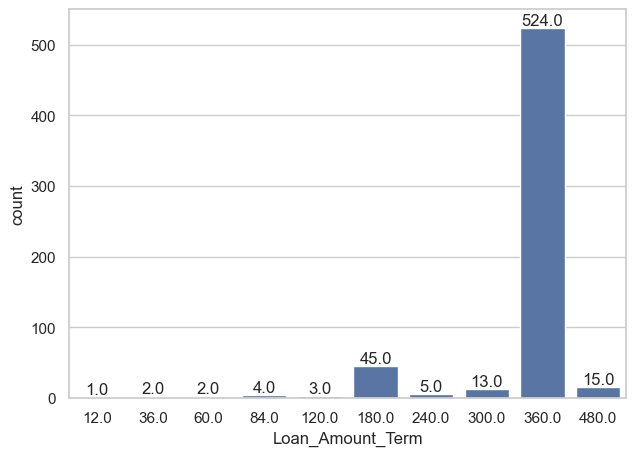
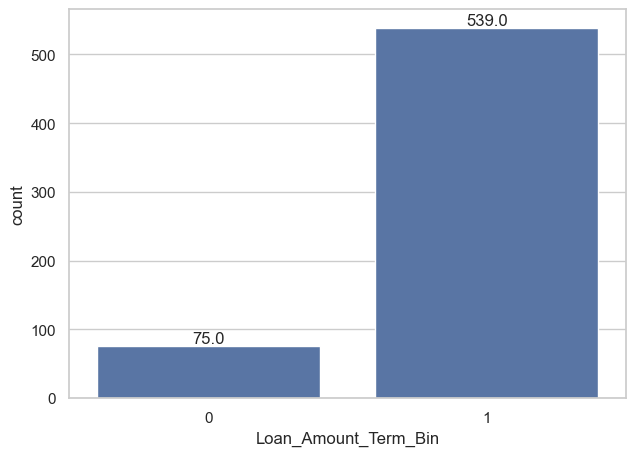
There are 4 continuous fields with outliers which require investigation and handling:

* ApplicantIncome  
  
* CoapplicantIncome  
  
* LoanAmount  
  
* Loan\_Amount\_Term  
  

Any data point falling outside the interquartile range is considered an outlier. We have developed a module that calculates the number and   
percentage of outliers and assesses whether removing these outliers impacts the correlation with the target variable and the overall distribution.   
The distribution has been evaluated using a probability-value threshold of < 0.05. The results are summarized in the table below:



Although removing them wouldn't impact the correlations, outliers were retained in the continuous variables because of their substantial presence.   
This indicates that these values are not just random anomalies, but are crucial elements of the dataset, vital for making accurate predictions.

The Loan\_Amount\_Term field is more categorical than continuous, given that it contains only 10 unique values.   
The distribution is uneven, with most entries (524 out of 614) falling within the 360-month category.   
Therefore, it would be logical to simplify this column by reducing the categories to just two: 360+ months and others.  
Following figures illustrate the before and after distribution:  
 

# Feature Evaluation

We have developed a function for each model that accepts the training and target data frames as arguments and returns an array of "importance" values (0/1) for each feature.   
These functions were executed in parallel using ThreadPoolExecutor(), allowing for faster and more efficient processing through a multi-threaded approach.   
The following table presents the aggregated results of this evaluation:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Feature** | **Lasso** | **SVM** | **GradientBoost** | **RandomForest** | **ElasticNet** |  | **Ridge** | **RFE Sum** |
| Married | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| Education | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| Credit\_History | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| ApplicantIncome | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 6 |
| CoapplicantIncome | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 6 |
| LoanAmount | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 6 |
| Dependents | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 5 |
| Self\_Employed | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 5 |
| Property\_Area | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 5 |
| Loan\_Amount\_Term\_Bin | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 5 |
| Gender | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 4 |

It's evident that the Gender field has the least importance, followed by Loan\_Amount\_Term, Property\_Area, Self\_Employed, and Dependents.   
Considering that the dataset is already small, removing these fields could negatively impact the model's predictive accuracy.   
Therefore, it is reasonable to retain these fields.

# Model Comparison

The objective of this stage is to determine which model will be deployed in production now that the data is fully prepared for training.   
The models chosen for this evaluation are widely recognized for their effectiveness in predicting categorical targets.

The following criteria are used to assess model performance:

* **Accuracy**: The proportion of correctly classified data points out of the total number of data points.
* **Precision**: The accuracy of true positive predictions, expressed as the ratio of correctly classified true positives to the total number of positive predictions.
* **Recall**: The ratio of true positive predictions to the actual number of positive data points.
* **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of both.
* **Log-Loss**: A performance metric that evaluates the accuracy of probabilistic predictions, with a focus on penalizing incorrect classifications.
* **AUC (Area Under the Curve)**: Measures the overall ability of the model to discriminate between positive and negative classes, by plotting the recall rate against the false positive rate.

The following table presents the performance metrics of the models under evaluation:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **f1-score** | **Log-loss** | **AUC** |
| SVC | 1 | 1 | 1 | 1 | 2.22E-16 | 1 |
| XGB | 0.95928 | 0.95423 | 0.98815 | 0.9709 | 1.47E+00 | 0.94199 |
| RandomForest | 0.9202 | 0.89597 | 1 | 0.94513 | 2.88E+00 | 0.8724 |
| GBM | 0.88111 | 0.85979 | 0.98815 | 0.91952 | 4.29E+00 | 0.81699 |
| ADABoost | 0.83225 | 0.81964 | 0.96919 | 0.88817 | 6.05E+00 | 0.75022 |
| Logistic Regression | 0.78176 | 0.80252 | 0.90521 | 0.85078 | 7.87E+00 | 0.70782 |

Based on the evaluation data, it is evident that XGBoost is the optimal candidate model, as it delivers the best performance metrics while avoiding overfitting.

# XGBoost Model Fine Tunning

## Grid search

To optimize the XGBoost model, we employed an explicit grid search strategy to identify the best set of hyperparameters for training.  
The following hyperparameters and their respective ranges were explored:

* **booster**: gbtree, dart
* **eta**: 0.01, 0.04, 0.08, 0.1, 0.2, 0.3
* **max\_depth**: 7, 8, 9, 11
* **min\_child\_weight**: 1, 4, 8, 12
* **gamma**: 0, 1, 4, 6, 8, 12
* **subsample**: 0.5, 0.7, 1

We evaluated the model's performance across 20 combinations that produced the highest accuracy.   
The optimal set of hyperparameters was determined to be:

{

'booster': 'dart',

'eta': 0.3,

'eval\_metric': 'auc',

'gamma': 1,

'max\_depth': 11,

'min\_child\_weight': 4,

'subsample': 0.7

}

## Model Performance After Fine-Tuning

The final performance measurements of the XGBoost model, after fine-tuning, are as follows:

* **Accuracy**: 0.9691
* **Precision**: 0.9632
* **Recall**: 0.9929
* **F1-Score**: 0.9778
* **Log-Loss**: 1.1154
* **AUC**: 0.9548

These results demonstrate that the model performs exceptionally well across all key metrics, confirming the effectiveness of the selected hyperparameters.

# Model usage

**Unlock Smarter Loan Approvals with Our Advanced Predictive Model**

Discover the potential of our cutting-edge predictive model, designed to revolutionize the loan approval process.

This demo model, developed on a limited dataset, showcases the power of advanced machine learning techniques. Imagine the possibilities when applied to your real-world data.

**Who Can Benefit?**

* **Banks and Financial Institutions**: Enhance your loan approval process with data-driven insights, reducing risk and improving decision accuracy.
* **Private Lending Companies**: Streamline approvals and increase confidence in your lending decisions, ensuring you back the right clients.
* **Loan Advisors and Consultants**: Empower your consultations with a tool that helps adjust loan parameters, improving the likelihood of approval and tailoring loans to meet clients’ needs.

**Why Choose Our Model?**

* **Scalable**: Adaptable to any dataset, this model can be customized to fit your specific needs, ensuring maximum accuracy and efficiency.
* **Proven Approach**: Built on advanced algorithms, our model is designed to deliver reliable predictions, making your approval process faster and smarter.
* **Versatile Applications**: Whether you're approving loans or advising clients, this model offers unparalleled support in making informed, data-backed decisions.

Take your loan approval process to the next level with our innovative predictive model. Contact us today to learn how we can help you integrate this technology into your operations.