Loan Approval Prediction Using Machine Learning Techniques

Assessment Report

Bishesh Giri

October 26, 2024

Contents

List of Tables

List of Figures

I. Introduction

I.I. Background Problem Definition

Overview of the Loan Approval Process

The loan approval process is a crucial decision-making step for financial institutions, including banks and lending agencies. When an individual or business applies for a loan, the institution must evaluate several factors to assess the applicant's financial health and the risk associated with lending. (Arun et al. 2016) Some of the primary factors considered in this process are:

1. **Credit History:** A borrower's past credit behavior, such as timely payments or defaults, plays a significant role in determining their creditworthiness.
2. **Income:** The applicant's income level helps evaluate whether they can afford to repay the loan over the agreed period.
3. **Loan Amount:** The size of the loan requested in relation to the applicant's income and assets is crucial in assessing risk.
4. **Assets and Liabilities:** Lenders also consider the applicant's assets and liabilities, which can provide security in case of default.
5. **Other Financial Metrics:** Additional parameters like the loan-to-value ratio, debt-to-income ratio, and employment status are also factored into the decision-making process.

After reviewing these details, the institution decides whether to approve or reject the loan application based on predefined thresholds and risk tolerances.

Challenges in Manual Loan Approval

The traditional manual approach to loan approval presents several challenges that make it both inefficient and inconsistent. One major issue is the time-consuming nature of the process. Loan officers must review numerous documents and financial reports, leading to delays that can frustrate applicants and slow down the operations of lending institutions. Moreover, manual loan approval is often subjective, with human judgment playing a significant role. This subjectivity can result in inconsistencies, as different loan officers might assess the same information differently, leading to biases or errors in judgment that cause unfair approvals or rejections. Another challenge is the risk of human error, such as incorrect data entry or misinterpretation of financial information, which can result in poor loan decisions and increase financial risk for the institution. Additionally, the manual approach struggles with scalability. As the number of loan applications grows, it becomes increasingly difficult to keep up with demand, causing backlogs and further delays. These issues highlight the need for an automated, data-driven loan approval system that can reduce time, eliminate bias, minimize errors, and improve scalability for financial institutions.

I.II. Importance of Loan Approval Prediction

Need for Automation

Automating loan approval decisions has become increasingly vital in today's fast-paced financial environment. Manual loan processing is not only time-consuming but also prone to biases and human errors. (Pušnik et al. 2019) With advancements in technology, automating the loan approval process offers several critical benefits:

1. Automation significantly reduces the time it takes to process loan applications. Financial institutions can handle a larger volume of applications in a shorter time, enabling faster decision-making and improving customer satisfaction.
2. Fewer human resources are needed to review and process applications, reducing the overall cost associated with loan approvals.

3. Automated systems apply the same decision-making criteria uniformly across all applications, eliminating the subjectivity and biases that can arise from human intervention. This ensures that all applicants are assessed fairly based on objective factors, such as credit history, income, and assets.

4. Automation, particularly through machine learning models, allows financial institutions to use large datasets for more accurate and data-driven decision-making. These models can identify patterns in borrower behavior and assess risk more effectively, leading to more informed loan approval decisions.

Impact on Financial Institutions

Accurate loan approval predictions go beyond improving efficiency; they have a direct impact on a financial institution's bottom line. One of the key benefits is better lending decisions, as accurate predictions allow institutions to effectively differentiate between high- and low-risk applicants. This leads to more informed decisions, with loans being granted to creditworthy individuals while reducing the risk of defaults. (Aphale & Shinde 2020) Additionally, predictive models enable institutions to assess the likelihood of defaults based on historical data and trends, helping to mitigate risk and protect the institution's financial health. Automation in loan approval also significantly improves the customer experience, as applicants can receive decisions within minutes rather than waiting days or weeks. This quick response time enhances customer satisfaction and fosters loyalty. Furthermore, automated systems offer scalability, allowing institutions to handle an increased volume of loan applications without requiring additional resources or time. This scalability is particularly valuable for institutions looking to grow and expand their customer base while maintaining operational efficiency.

I.III. Role of Machine Learning

Machine learning models play a crucial role in predicting loan approval outcomes by uncovering patterns in historical data that may be challenging for humans to identify. These models can analyze vast datasets, evaluating factors such as credit score, income, loan amount, and more, to provide accurate predictions. By learning from past loan approvals and rejections, machine learning algorithms make data-driven decisions that enhance the reliability of loan approval processes. (Kadam et al. 2021)

Machine learning models transform the decision-making process by leveraging historical data. They analyze past loan applications, learn relationships between various factors, and predict future loan outcomes with greater accuracy. This leads to more informed, objective, and efficient decisions, reducing human error and bias.

I.IV. Objective

- **Objective:** The goal of this project is to develop a predictive model that determines whether a loan application will be approved or rejected. This prediction is based on various factors from the applicant's profile, such as credit history, income, loan amount, and other financial attributes.

- **Potential Benefits:**

    1. **Automation:** Automate the loan approval process, reducing the need for manual intervention and speeding up decision-making.
    2. **Improved Accuracy:** Enhance decision-making by accurately predicting loan outcomes based on historical data patterns.
    3. **Risk Reduction:** Help lending institutions minimize the risk of approving loans that may result in default.
    4. **Efficiency:** Streamline operations, saving time and resources while ensuring consistent and fair decisions.

II. About the Dataset

The Loan Approval dataset contains financial records and essential details related to loan applications, designed to predict loan approval outcomes made by lending institutions. Each record in the dataset represents an applicant's financial and personal profile, encompassing several key features.

Feature	Description
CIBIL Score	A credit score that assesses the creditworthiness of the applicant.
Income	The applicant's total income, typically used to gauge their ability to repay the loan.
Employment Status	Indicates whether the applicant is employed, self-employed, or unemployed.
Loan Term	The duration for which the loan is taken.
Loan Amount	The total amount of loan applied for.
Assets Value	Represents the value of the applicant's assets, which may be considered as collateral.
Loan Status	The target variable indicating whether the loan was approved or denied.

Table 1: Loan Approval Dataset Feature Descriptions

(The dataset used for this analysis, can be accessed here[1].)

This dataset is commonly used in machine learning to develop predictive models that assist financial institutions in evaluating loan approval risks. By analyzing the relationships between these features and the loan status, predictive models can be trained to forecast the likelihood of loan approval, supporting more accurate, data-driven lending decisions.

II.I. Library Used

During the analysis process, we utilized "Python Programming" language to visualize and train the model. To make the most of our dataset, various libraries available in python was utilized. These libraries helped to easily manipulate, transform, visualize and the train the data. As a result, meaningful conclusions and observations from our analysis were drawn.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder, OneHotEncoder,
    StandardScaler
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.model_selection import train_test_split, StratifiedKFold,
    cross_val_score
from sklearn.metrics import classification_report, accuracy_score,
    precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix
import optuna
from sklearn.ensemble import StackingClassifier
from lime.lime_tabular import LimeTabularExplainer
```

The integration of these libraries allowed us to streamline the data preprocessing pipeline, explore intricate patterns within the dataset, and enhance the overall accuracy and robustness of our predictive model. As a result, our analysis provided actionable insights into loan behavior, contributing to a deeper understanding of the system's dynamics.

[1]Link to Dataset: `https://www.kaggle.com/datasets/architsharma01/loan-approval-prediction-dataset`

III. Plan of Action

The following figure outlines the steps involved in making predictions. Each step is crucial for ensuring the accuracy and effectiveness of the predictive model. By following this Data Science Life Cycle process (Wing 2019), we can systematically approach the problem and derive meaningful model.

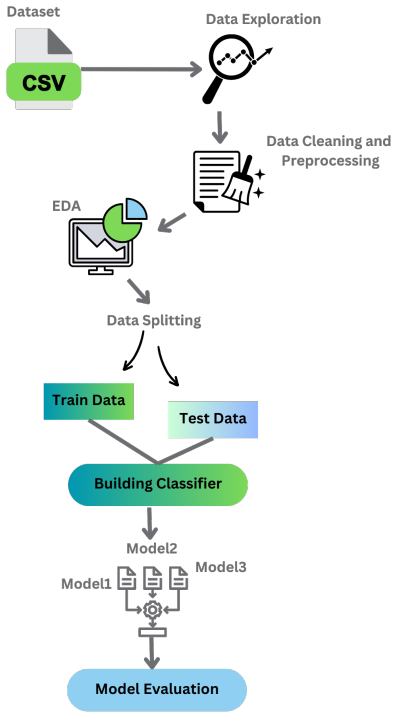


Figure 1: The proposed approach for machine learning

The figure above outlines the complete workflow for the machine learning project. It illustrates the sequential plan of action, starting with data exploration and cleaning, followed by exploratory data analysis (EDA) and preprocessing steps. After preparing the data, it is split into training and testing sets. Various classifiers are then built and evaluated to identify the best-performing models. The top-performing models are selected for further hyperparameter tuning and are finally combined using a stacking ensemble to enhance predictive performance.

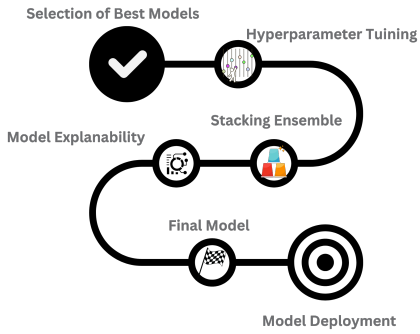


Figure 2: The proposed approach for model development

In the development process, the top-performing models are first selected based on their evaluation

results. These models are then fine-tuned using Optuna to optimize their hyperparameters. After tuning, the best models are combined through a stacking classifier to enhance performance. The stacked model is further interpreted using LIME (Local Interpretable Model-agnostic Explanations) to gain insights into its predictions. This final model is then prepared for deployment, ensuring it is both accurate and interpretable for real-world applications.

IV. Data Exploration

The dataset comprises approximately 4269 entries and includes 13 features. It is well-maintained with no null values and no duplicate entries that need to be addressed. The table presented below describe the features and its data type.

Column	Dtype	Description
loan_id	int64	Unique identifier for each loan application
no_of_dependents	int64	Number of dependents the applicant has
education	object	Educational qualification of the applicant
self_employed	object	Indicates if the applicant is self-employed (Yes/No)
income_annum	int64	Annual income of the applicant
loan_amount	int64	Total amount of loan applied for
loan_term	int64	Loan repayment period in months
cibil_score	int64	Credit score of the applicant assessing creditworthiness
residential_assets_value	int64	Value of the applicant's residential assets
commercial_assets_value	int64	Value of the applicant's commercial assets
luxury_assets_value	int64	Value of the applicant's luxury assets
bank_asset_value	int64	Value of the applicant's bank account and other liquid assets
loan_status	object	Target variable indicating loan approval status (Approved/Rejected)

Table 2: Dataset: Column Overview and Descriptions

Categorical Columns:

- **education:** Educational qualification of the applicant (Graduate or Not Graduate).
- **self_employed:** Employment status of the applicant (Yes or No).
- **loan_status:** Target variable indicating whether the loan was Approved or Rejected.

IV.I. Missing Values

- There are **no missing values** in any of the columns, ensuring that all data is complete and ready for analysis.

IV.II. Summary Statistics

- The **numerical columns** display a wide range of values, providing the following insights:
  - The mean of **income_annum** is approximately **5.06 million**, with a maximum value nearing **9.9 million**.
  - The **cibil_score** ranges between **300 and 900**, with a mean of around **600**.
  - The **loan_amount** varies significantly, from a minimum of **300,000** to a maximum of nearly **40 million**.

IV.III. Distribution of the Target Variable (*loan_status*)

- Analyzing the distribution of **loan_status** will help assess whether the dataset is **balanced or imbalanced**, a crucial factor for model training and evaluation.

IV.IV. Column Name Issues

- There are **leading spaces** in some column names (e.g., *' no_of_dependents'*), which will need to be addressed during the data cleaning process to avoid errors during analysis and model training.

This dataset provides comprehensive loan approval patterns, capturing various activities and associated loan status. The detailed features enable a thorough analysis of loan behavior. By addressing and utilizing these features, we can gain valuable insights into network traffic patterns, identify potential security threats, and enhance firewall rules and configurations.

V. Data Cleaning

The data cleaning process started by removing the loan_id column, as it was not relevant for analysis. Following this, leading spaces in some column names were identified and corrected to ensure consistency and avoid potential errors during further analysis and modeling.

```
df.drop(columns=['loan_id'], inplace=True)

df.rename(columns=lambda x: x.strip(), inplace=True)
```

V.I. Outlier Detection and Removal

To detect outliers, both the Interquartile Range (IQR) and Z-score methods were employed. As a result, the z-score method was used outliers detection in several columns were identified and subsequently removed from the dataset to enhance the integrity of the analysis.

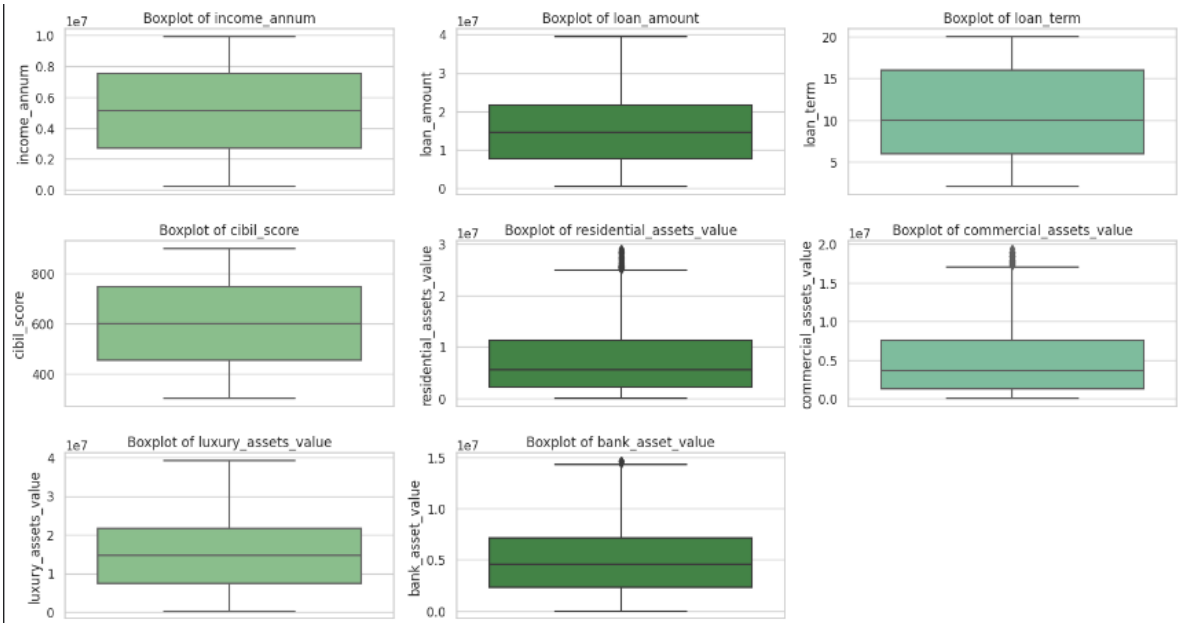


Figure 3: Boxplot for outlier detection

```
def detect_outliers_zscore(df, columns, threshold=3):
    outlier_indices = []
    for col in columns:
        z_scores = np.abs(stats.zscore(df[col]))
        outlier_indices.extend(np.where(z_scores > threshold)[0])
    return set(outlier_indices)

outliers_zscore = detect_outliers_zscore(df, numeric_columns)
print(f'Number of outliers detected using Z-scores: {len(outliers_zscore)}')
```

Total of 33 outlier were detected and finally dropped. Finally, observed skewness in certain columns, which will be addressed in the preprocessing step to ensure a more balanced dataset for further analysis.

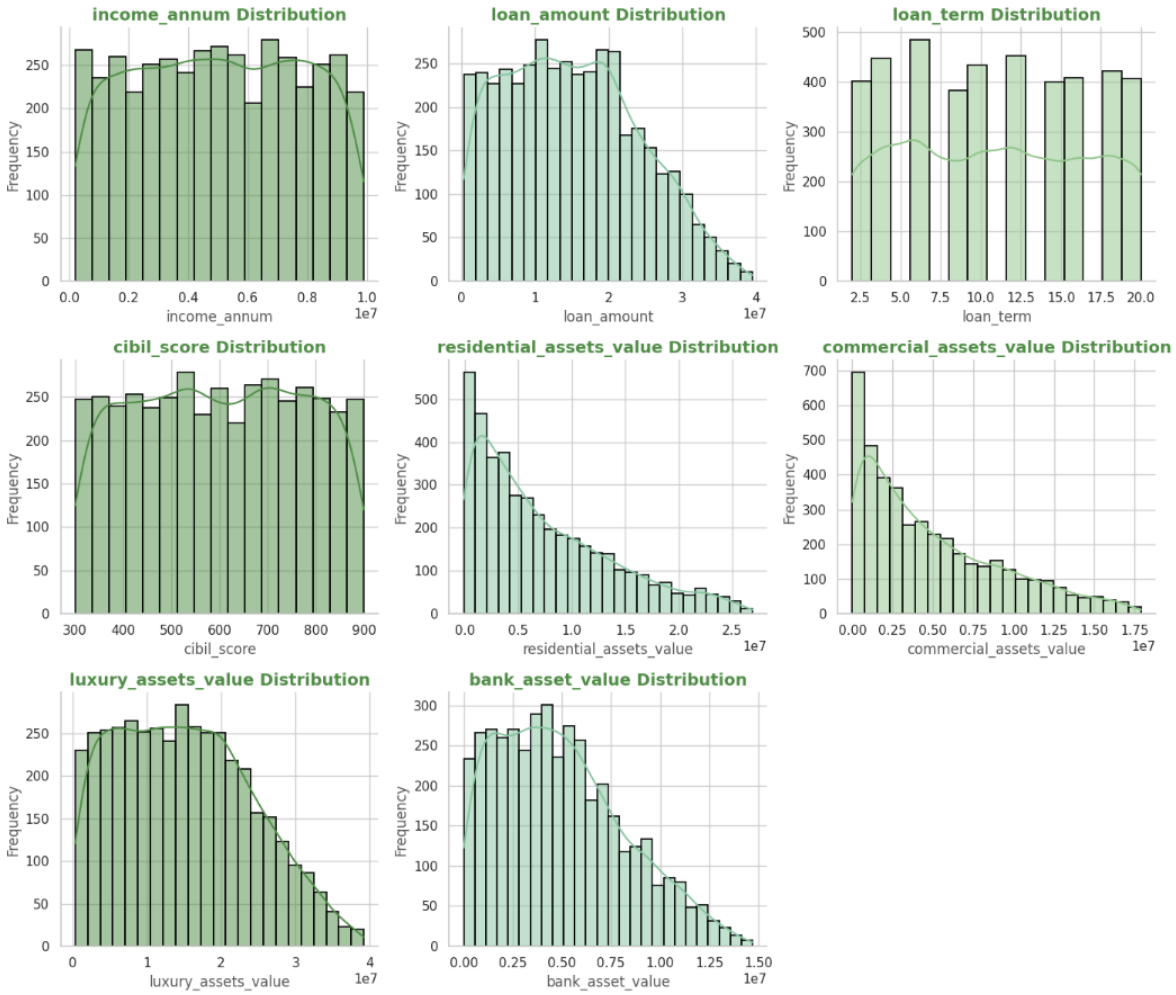


Figure 4: Histogram for Skewness

VI. Exploratory Data Analysis

For this project, conducting a comprehensive Exploratory Data Analysis (EDA) is crucial to uncover valuable insights from the dataset and to understand the relationships between features and the target variable (*loan_status*).

The EDA process will be organized into several key steps:

- **Univariate Analysis:** Focuses on exploring individual features to understand their distributions and summary statistics.
- **Bivariate Analysis:** Examines the relationships between two variables, particularly how each feature relates to the target variable.
- **Multivariate Analysis:** Explores interactions among multiple features to identify patterns or dependencies.
- **Correlation Analysis:** Measures the strength and direction of relationships between numerical features, helping to detect multicollinearity or significant connections.

- **Feature Combinations:** Investigates how combinations of features influence the outcome, offering deeper insights into feature interactions.

Each of these steps will contribute to building a strong foundation for modeling and further analysis.

1. **Univariate Analysis:**

   In the univariate analysis, we focused on examining the categorical columns. Specifically, the distribution of the **education** and **self_employed** columns was analyzed, revealing that the data is relatively evenly distributed across these categories. However, when analyzing the **loan_status** column, it was observed that the number of approved loans significantly exceeds the number of rejections. This imbalance in the target variable suggests that the dataset is skewed toward approved loans, which may influence the model's predictions.

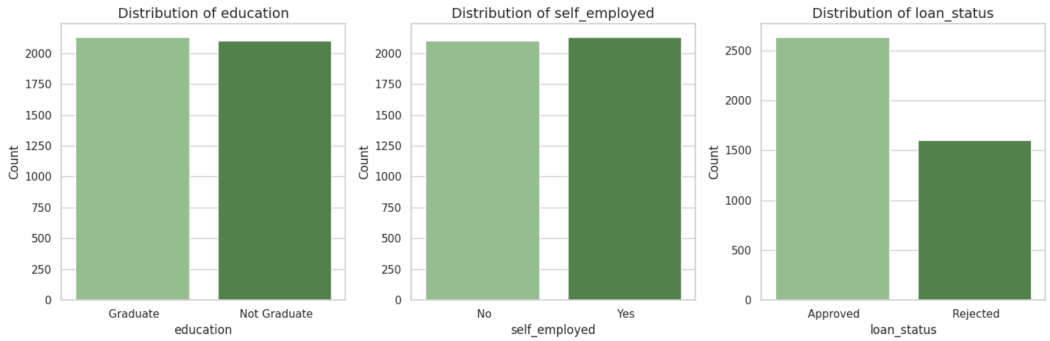


Figure 5: Barplot Categorical Features

2. **Bivariate Analysis:**

   For the bivariate analysis, a series of box plots were created to explore the relationship between numerical features and the target variable, **loan_status**.

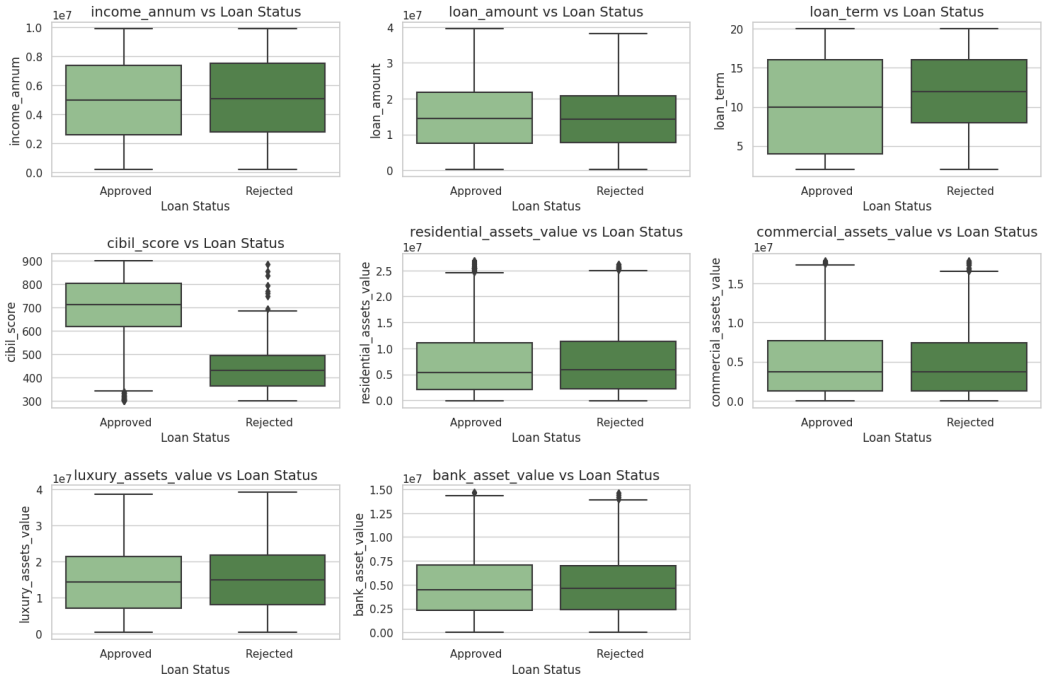


Figure 6: Boxplot Bivariate Analysis

This approach helps identify which numerical features have a significant impact on loan approval and can offer insights into how these features vary for different loan outcomes. This is particularly useful here for visualizing the spread, central tendency (like medians), and the presence of outliers across the two categories of the target variable.

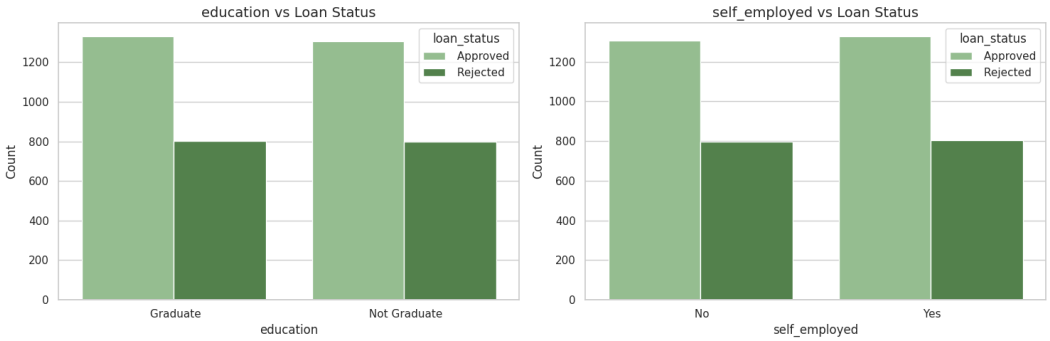


Figure 7: Bargraph Bivariate Analysis

Similarly for categorical columns, count plots were generated to visually compare the distribution of categorical variables with respect to the target variable, **loan_status**. Each plot shows the count of loan approvals and rejections for the given categorical feature, helping us observe any trends or patterns in the data.

The analysis revealed that the distribution of values for loan approvals and rejections is quite similar across both categorical columns. In both the education and self_employed columns, there are no significant imbalances, indicating that these factors do not drastically influence the loan approval outcome on their own. This suggests that other features may play a more prominent role in determining loan status.

3. **Correlation Analysis:**

   The correlation matrix below reveals several important insights into the relationships between the numerical features related to loan approval. First, a very strong positive correlation exists between income_annum and loan_amount (0.93), suggesting that applicants with higher annual incomes tend to apply for larger loan amounts. Additionally, asset-related features, such as luxury_assets_value and bank_asset_value, also exhibit a strong relationship with income. For instance, luxury_assets_value is highly correlated with both income_annum (0.93) and loan_amount (0.86), indicating that individuals with higher incomes are more likely to own luxury assets. Similarly, bank_asset_value shows a notable correlation with income (0.85), reinforcing the link between income and asset ownership.

   In contrast, the cibil_score, a key feature in loan approval decisions, demonstrates relatively low correlations with other features, indicating that it operates independently of variables such as income, loan amount, and asset values. This suggests that an applicant's creditworthiness, as represented by their CIBIL score, is evaluated separately from their financial standing in terms of income and assets.

   Furthermore, the loan_term variable exhibits minimal correlation with all other features, implying that the length of the loan term does not depend significantly on an applicant's income or asset values. Finally, the analysis also highlights strong intercorrelations between different asset categories, such as residential_assets_value, commercial_assets_value, and luxury_assets_value, suggesting that individuals who own one type of asset are more likely to own others as well.

   Overall, the correlation analysis underscores the importance of income and asset values in determining loan amounts, while factors like CIBIL score and loan term seem to be evaluated on their own.

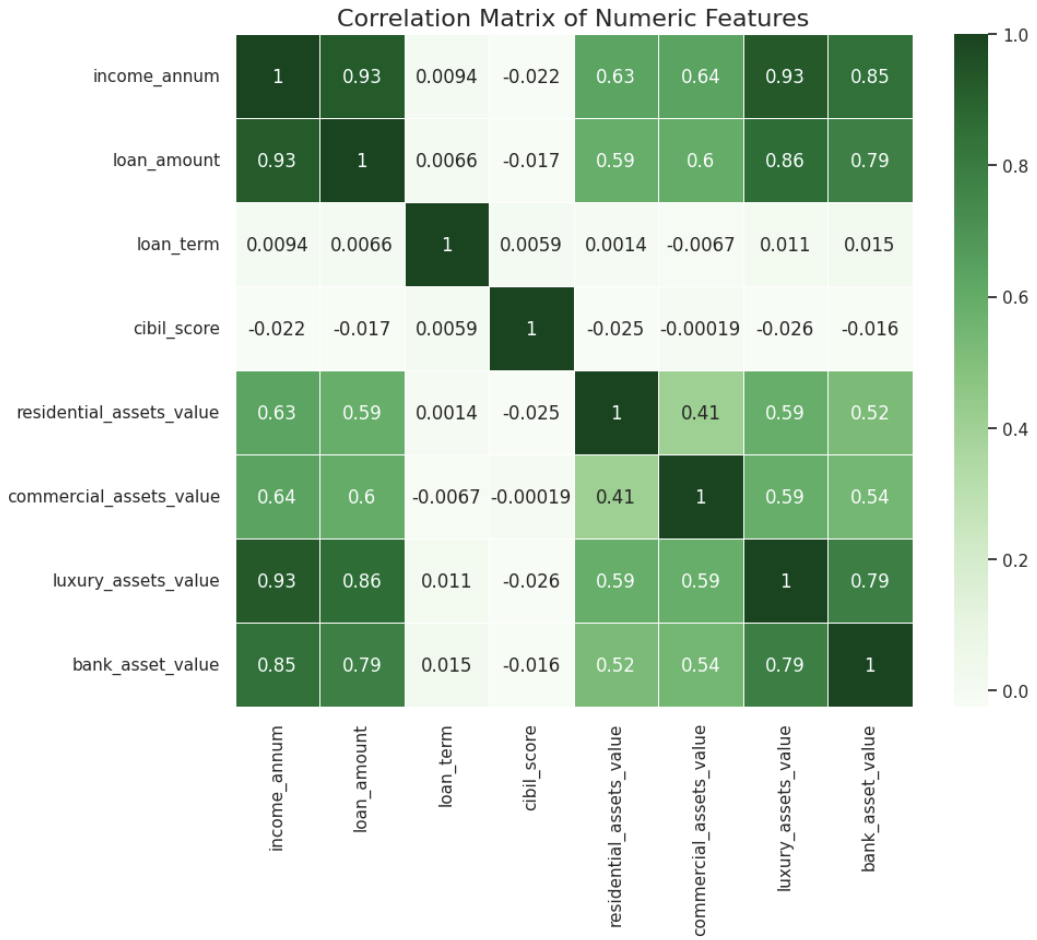


Figure 8: Analysis features through heat map of the correlation

4. **Understanding Feature Combinations:**

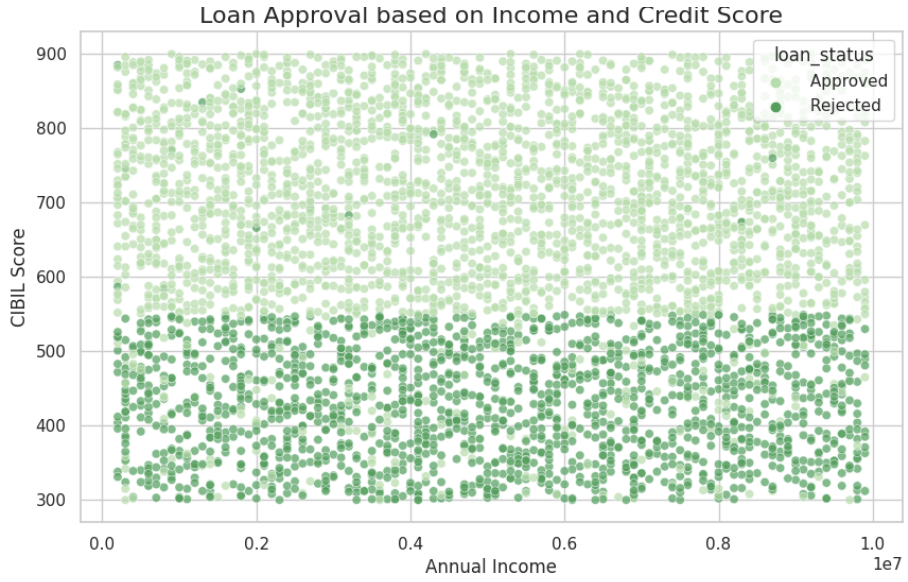


Figure 9: Scatter plot for loan approval based on income and credit score

The scatter plot illustrates the relationship between annual income and CIBIL score with respect to loan approval status. Data points are color-coded to distinguish between approved and rejected loans, with approved loans shown in lighter green and rejected loans in darker green.

From the visualization, it is evident that loan approval tends to be associated with higher CIBIL scores, as the majority of approved loans are clustered in the upper part of the plot. Conversely, loan rejections are more concentrated among applicants with lower CIBIL scores, especially below 500. Interestingly, annual income does not appear to have a direct or strong impact on loan approval or rejection, as both approved and rejected loans are spread across various income levels without a clear pattern. This suggests that while income plays a role, CIBIL score is a more decisive factor in determining loan approval in this dataset.

Overall, the plot highlights that a good CIBIL score significantly increases the likelihood of loan approval, while income has a more limited or indirect effect. This analysis emphasizes the importance of evaluating both income and credit score together when assessing an applicant's creditworthiness.

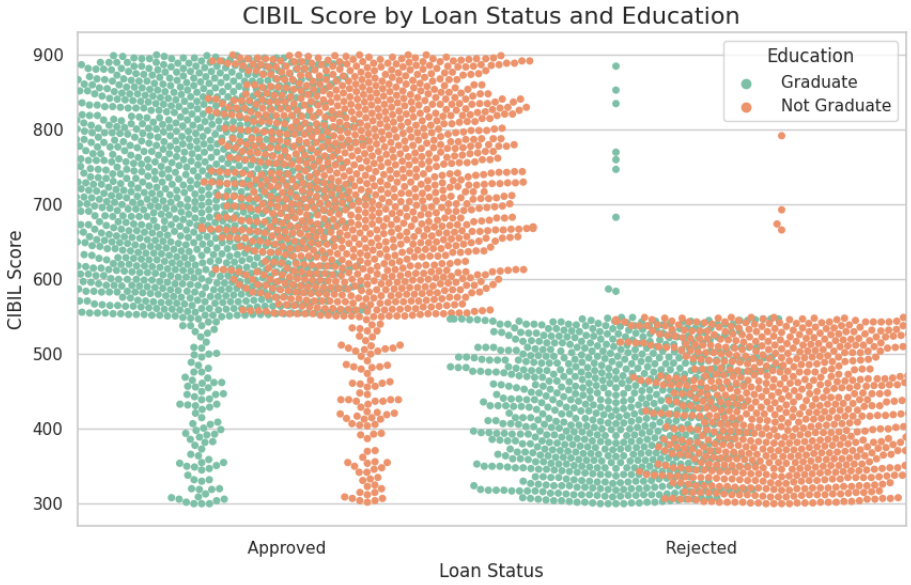


Figure 10: Scatter plot for CIBIL Score

The plot illustrates the relationship between CIBIL score, loan status, and education level. Graduates (green) and non-graduates (orange) both see more approvals with higher CIBIL scores. However, non-graduates tend to have a slightly higher proportion of lower CIBIL scores, especially in the rejected category. Overall, higher CIBIL scores are linked with loan approvals, regardless of education.

5. **Multivariate Analysis: Interaction Between Features**

   The plot below presents a pair plot analyzing the relationship between annual income, loan amount, CIBIL score, and loan approval status. The diagonal plots show the distribution of each variable, while the scatter plots illustrate the correlations between them, with lighter dots indicating approved loans and darker ones indicating rejected loans.

   (a) **Income and Loan Amount:** A positive correlation exists between income and loan amount, with higher incomes generally corresponding to higher loan amounts. Both approved and rejected loans span a similar range, but rejections are more prevalent for higher loan amounts.

   (b) **Income and CIBIL Score:** There isn't a clear correlation between annual income and CIBIL score. Both approved and rejected loans are distributed across similar ranges of income and CIBIL scores, though approved loans are more frequent for higher CIBIL scores.

(c) **Loan Amount and CIBIL Score:** No strong correlation between loan amount and CIBIL score is evident. Both approved and rejected loans are distributed evenly across different CIBIL scores for various loan amounts.

(d) **CIBIL Score Distribution:** Loans are more likely to be approved for applicants with CIBIL scores above 600, while lower CIBIL scores (around 300-500) are associated with a higher rate of rejections.

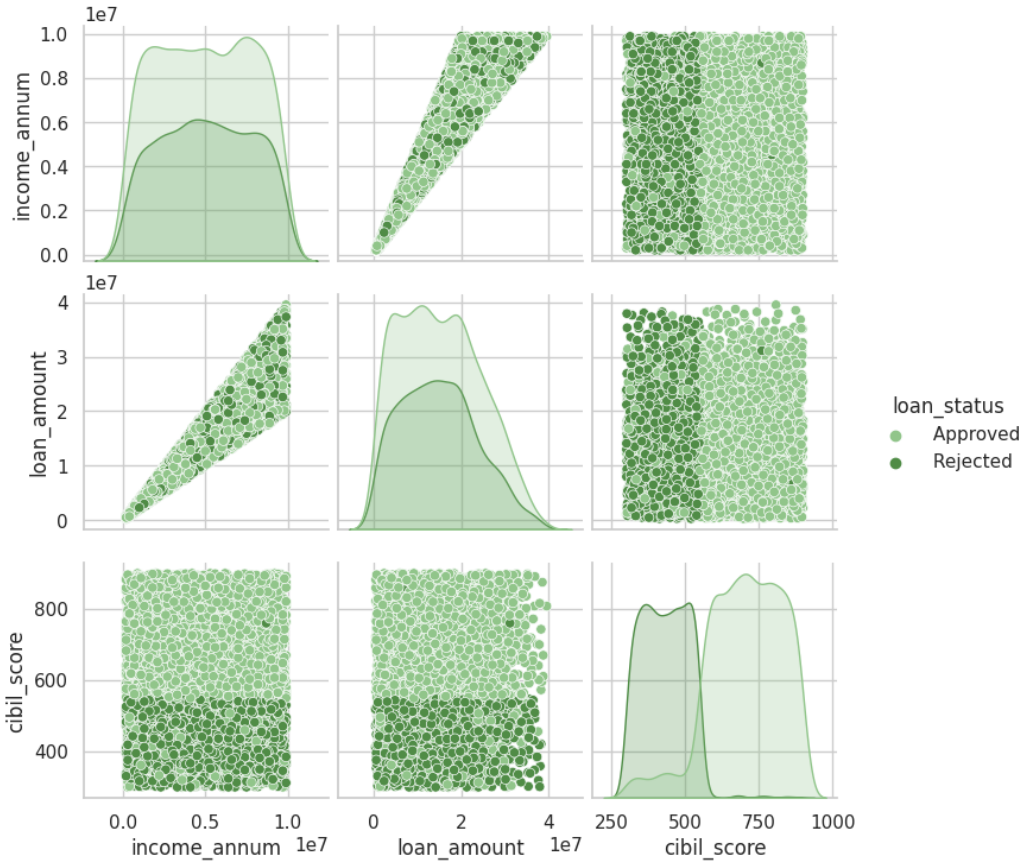


Figure 11: Pair plot for Multivariate Analysis

VI.I. Summary of EDA

The exploratory data analysis (EDA) process provided valuable insights into the factors influencing loan approval decisions. By examining relationships between variables such as annual income, loan amount, and CIBIL score, we observed several important trends. Higher CIBIL scores and moderate-to-high incomes are positively associated with loan approvals, while lower CIBIL scores (below 600) contribute to higher rejection rates. Loan amounts do not show a significant direct influence on approval status, but very high loan amounts are more often rejected, especially when paired with lower CIBIL scores.

The analysis also revealed that income and loan amount are positively correlated, as expected, but neither is a strong independent predictor of loan approval without considering the CIBIL score. Visualizations further demonstrated the importance of creditworthiness (CIBIL score) in the decision-making process, while income and loan amount seem to act as supplementary factors. Overall, the EDA has provided a clear direction for model building, highlighting the key predictors and their relationships with loan approval outcomes.

VII. Data Preprocessing

Approach to Data Preprocessing

With insights gained from the exploratory data analysis (EDA), we are now prepared to proceed to the data modeling phase. However, before that, it is essential to preprocess the data to ensure optimal model performance. The data preprocessing process will involve several key steps.

First, **Feature Engineering** will be conducted to create new features based on existing data, enhancing the model's predictive power. Next, all categorical columns will undergo **One Hot Encoding**, as these features consist of boolean values or contain only two unique values. This ensures that the model can effectively interpret and utilize these features.

```
def feature_engineering(X):
    X['loan_to_income_ratio'] = X['loan_amount'] / X['income_annum']

    X['total_asset_value'] = (X['residential_assets_value'] +
                              X['commercial_assets_value'] +
                              X['luxury_assets_value'] +
                              X['bank_asset_value'])

    return X
```

In the feature engineering process, two new columns were created to enhance the dataset by capturing additional insights from existing features. These columns are designed to help improve the predictive power of the model by offering more meaningful relationships between variables.

1. **Loan to Income Ratio (loan_to_income_ratio):**

   This feature is computed by dividing the loan amount by the applicant's annual income:

   $$\text{loan\_to\_income\_ratio} = \frac{\text{loan\_amount}}{\text{income\_annum}}$$

   This ratio reflects the proportion of the loan amount relative to the applicant's annual income, providing insights into their debt-to-income burden. A higher ratio indicates that the applicant is borrowing a larger portion of their income, which could imply higher risk in terms of loan repayment.

2. **Total Asset Value (total_asset_value):**

   This feature aggregates the values of all the applicant's assets, including residential, commercial, luxury, and bank assets:

   $$\text{total\_asset\_value} = \sum_{\text{asset} \in \{\text{residential,commercial,luxury,bank}\}} \text{asset\_value}$$

   This feature offers a comprehensive view of the applicant's financial strength by combining various asset types into a single metric. It provides the lender with a clearer picture of the total collateral available, which may influence the likelihood of loan approval.

In addition to scaling and encoding, binning was applied to the cibil_score feature to improve model robustness by grouping the scores into meaningful categories, which can help reduce noise and enhance interpretability. Additionally, a new feature, income_per_dependent, was created to provide insights into income distribution relative to dependents, offering the model a more nuanced understanding of financial stability in the context of loan eligibility.

For encoding the categorical features, the `LabelEncoderTransformer` was tested by extending the `BaseEstimator` and `TransformerMixin` classes from `sklearn.base`. This custom transformer was used to perform label encoding on the categorical columns. However, OneHotEncoding was preferred for this project. Label encoding was chosen as the most suitable method since all categorical features contained only two unique values. For example, the `education` column had values such as 'Graduated' and 'Non-Graduated,' which were efficiently converted into numeric representations to facilitate model training.

```
def feature_engineering(X):

    X['income_per_dependent'] = X['income_annum'] / (X['no_of_dependents'] +
         1)
    def cibil_category(score):
        if score < 500:
            return 'Low'
        elif 500 <= score <= 700:
            return 'Medium'
        else:
            return 'High'

    X['cibil_category'] = X['cibil_score'].apply(cibil_category)
    return X
```

Following this, we will construct a **Preprocessing Pipeline** to handle the transformation and scaling of the data. Numerical data will be normalized using `StandardScaler()`, ensuring that all features contribute equally during model training. Additionally, a one hot encoder transformer will be integrated into the pipeline to streamline the preprocessing of categorical variables. Finally, a fully preprocessed DataFrame will be generated, which will serve as the foundation for all subsequent data modeling activities. This structured approach to data preprocessing will facilitate smoother transitions to model development and enhance the overall performance of the models.

```
numerical_features = [
    'income_annum', 'loan_amount', 'loan_term',
    'cibil_score', 'residential_assets_value',
    'commercial_assets_value', 'luxury_assets_value',
    'bank_asset_value', 'income_per_dependent']

categorical_features = ['education', 'self_employed', 'cibil_category']
selected_features = numerical_features + categorical_features

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ],
    remainder='passthrough'
)

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor)
])
```

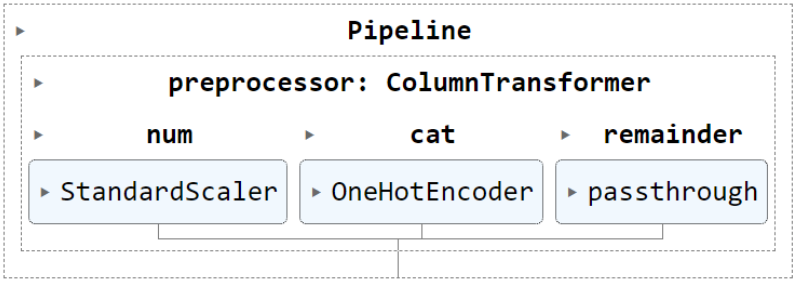


Figure 12: Data Preprocessing Pipeline

VIII. Data Splitting and Model Training

The dataset was split into training and testing sets in an 80:20 ratio, ensuring that 80% of the data was utilized for model training, while the remaining 20% was reserved for testing. This approach allows the model to learn from a substantial portion of the data while still leaving enough data to evaluate its performance effectively.

An adequate amount of training data is crucial for reliable parameter estimation, as insufficient data can lead to high variability in the estimates. Conversely, having fewer test cases can result in more consistent performance measures, though it may limit the robustness of the evaluation.

```
X = processed_df.drop(columns=['loan_status'])
y = processed_df['loan_status']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)
```

To further enhance the accuracy of the model, cross-validation techniques were employed. This method helps mitigate overfitting and provides a more reliable assessment of the model's performance across different subsets of the data.

VIII.I. Description of Classification Models

This section provides an overview of the different models used in this analysis:

Model	Description
Logistic Regression	A linear model for binary classification that estimates the probability of a class label based on one or more predictor variables.
Decision Tree	A non-linear model that makes decisions based on a series of rules derived from feature values, visualized as a tree structure.
Random Forest	An ensemble method that combines multiple decision trees to improve accuracy and control overfitting by averaging their predictions.
XGBoost	An optimized gradient boosting framework that is efficient, scalable, and offers superior performance for classification and regression tasks.
LightGBM	A gradient boosting framework that uses a histogram-based approach to accelerate training and reduce memory usage, particularly effective for large datasets.
CatBoost	A gradient boosting library designed to handle categorical features directly, preventing overfitting and offering strong performance with minimal preprocessing.
SVC (Support Vector Classifier)	A powerful classifier that finds the optimal hyperplane for separating classes in high-dimensional space, capable of handling non-linear boundaries using kernel functions.
KNN (K-Nearest Neighbors)	A simple, instance-based learning algorithm that classifies samples based on the majority class among the K closest training examples.
Naive Bayes	A probabilistic classifier based on Bayes' theorem, assuming independence among predictors, often used for text classification tasks.
MLP (Multi-layer Perceptron)	A type of neural network with one or more hidden layers, capable of capturing complex relationships in data through non-linear activation functions.
AdaBoost	An ensemble technique that combines multiple weak classifiers, adjusting the weights of misclassified instances to improve overall model accuracy.
Extra Trees	An ensemble learning method that builds multiple decision trees with random feature selection, promoting diversity among trees to enhance performance and reduce overfitting.

Table 3: Overview of Classification Models

VIII.II. Model Training and Evaluation

To assess the performance of the different models, we evaluated each model based on accuracy, roc_auc_score, MCC and detailed classification metrics. The evaluation was conducted using a test set to ensure that the performance metrics are representative of how the models perform on unseen data.

Performance Metrics

To evaluate the model performance model accuracy and classification report were printed. The classification report shows Recall, F1 Score, and Precision respectively. These metrics depend on True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The following metrics were used to evaluate the models:

- **Accuracy:** The proportion of correctly classified instances out of the total instances.
- **Precision:** The proportion of true positive predictions among all positive predictions made by the model.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** The proportion of true positive predictions among all actual positives.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score:** The harmonic mean of precision and recall, providing a single metric to evaluate the model's performance on each class.

$$F_1 \text{ Score} = 2 \times \frac{P \cdot R}{P + R}$$

- **ROC AUC Score:** The Area Under the Receiver Operating Characteristic Curve (ROC AUC) measures the model's ability to distinguish between classes. It quantifies the trade-off between true positive rate and false positive rate at various thresholds.

$$\text{AUC} = \int_0^1 \text{TPR}(FPR)\, dFPR$$

Where: - TPR = $\frac{TP}{TP+FN}$ (True Positive Rate) - FPR = $\frac{FP}{FP+TN}$ (False Positive Rate)

- **Matthews Correlation Coefficient (MCC):** A correlation coefficient that takes into account true and false positives and negatives, providing a balanced measure of performance even for imbalanced datasets.

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Where: - TP = True Positives - TN = True Negatives - FP = False Positives - FN = False Negatives

- **Confusion Matrix:** A table that is often used to describe the performance of a classification model, showing the counts of true positive, true negative, false positive, and false negative predictions.

Confusion Matrix =

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

```
models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "XGBoost": XGBClassifier(),
    "LightGBM": LGBMClassifier(),
    "CatBoost": CatBoostClassifier(verbose=0),
    "SVC": SVC(probability=True),
    "KNN": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
    "MLP": MLPClassifier(),
    "AdaBoost": AdaBoostClassifier(),
    "Extra Trees": ExtraTreesClassifier()
}

skf = StratifiedKFold(n_splits=5)


def evaluate_model(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)

    cv_score = cross_val_score(model, X_train, y_train, cv=skf, scoring='
        accuracy').mean()

    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1] if hasattr(model, "
        predict_proba") else None

    metrics = {
        'CV Accuracy': cv_score,
        'Test Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1 Score': f1_score(y_test, y_pred),
        'MCC': matthews_corrcoef(y_test, y_pred),
        'ROC-AUC': roc_auc_score(y_test, y_prob) if y_prob is not None else
            'N/A',
        'Confusion Matrix': confusion_matrix(y_test, y_pred).ravel()  # (tn,
            fp, fn, tp)
    }
    return metrics
```

```
results = {}

for model_name, model in models.items():
    print(f"Training and evaluating {model_name}...")
    metrics = evaluate_model(model, X_train, y_train, X_test, y_test)
    results[model_name] = metrics

results_df = pd.DataFrame(results).T

results_df = results_df.sort_values(by='F1 Score', ascending=False)
results_df
```

The model training process involved the evaluation of twelve distinct models. To ensure robust predictions on the test set, we employed StratifiedKFold with five splits. This approach allowed for a comprehensive assessment of each model's performance across various metrics, which will be analyzed to inform subsequent steps. Following this analysis, the four best-performing models will be selected for further hyperparameter tuning and stacking. This strategy aims to enhance overall model performance and achieve more accurate predictions.

	CV Accuracy	Test Accuracy	Precision	Recall	F1 Score	MCC	ROC-AUC	Confusion Matrix
LightGBM	0.981108	0.978774	0.981763	0.964179	0.972892	0.955557	0.998819	[507, 6, 12, 323]
CatBoost	0.979042	0.975236	0.978659	0.958209	0.968326	0.948143	0.997981	[506, 7, 14, 321]
XGBoost	0.979928	0.975236	0.981595	0.955224	0.96823	0.948182	0.997992	[507, 6, 15, 320]
Random Forest	0.974908	0.96934	0.963964	0.958209	0.961078	0.935798	0.996314	[501, 12, 14, 321]
Decision Tree	0.97373	0.96934	0.966767	0.955224	0.960961	0.935766	0.966891	[502, 11, 15, 320]
AdaBoost	0.969597	0.964623	0.966361	0.943284	0.954683	0.925858	0.994324	[502, 11, 19, 316]
Extra Trees	0.952771	0.945755	0.936556	0.925373	0.930931	0.886317	0.989471	[492, 21, 25, 310]
KNN	0.602124	0.590802	0.47619	0.358209	0.408859	0.107931	0.551479	[381, 132, 215, 120]
Logistic Regression	0.626623	0.604953	0.0	0.0	0.0	0.0	0.493925	[513, 0, 335, 0]
SVC	0.626623	0.604953	0.0	0.0	0.0	0.0	0.494481	[513, 0, 335, 0]
Naive Bayes	0.626623	0.604953	0.0	0.0	0.0	0.0	0.537593	[513, 0, 335, 0]
MLP	0.525589	0.604953	0.0	0.0	0.0	0.0	0.5	[513, 0, 335, 0]

Figure 13: Comparing Metrics of Different Models

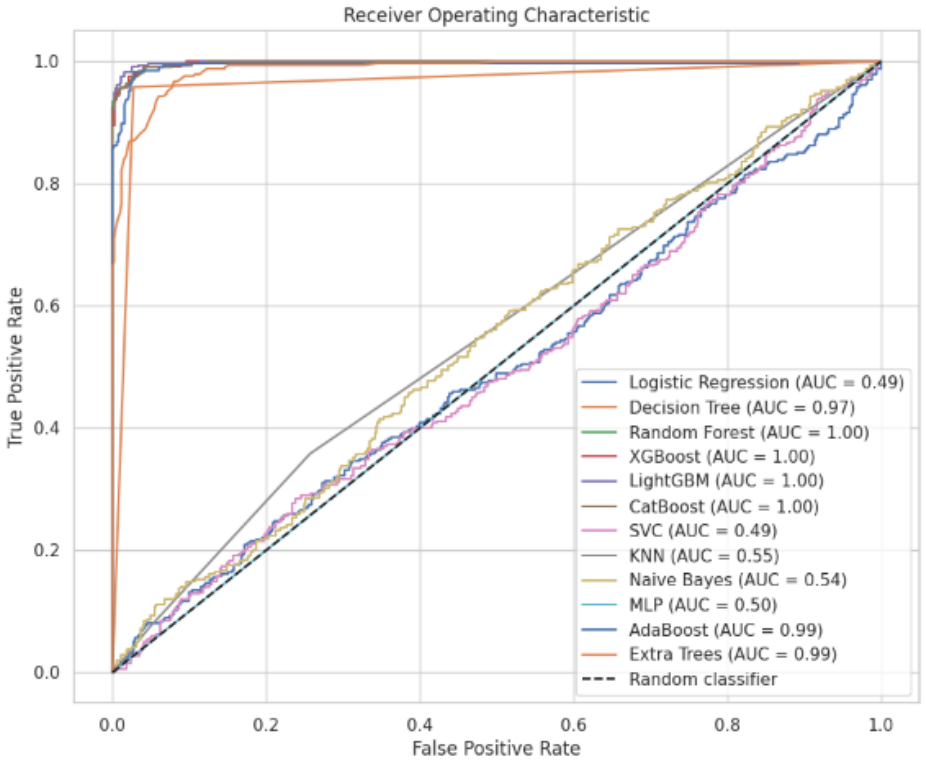


Figure 14: ROC Curve of Different Models

After splitting the dataset into training and testing sets, we evaluated the performance of various models using multiple metrics, including the ROC AUC score, Matthews correlation coefficient (MCC), precision score, accuracy, and F1 score. The analysis indicated that boosting models, particularly LightGBM (LGBM) and XGBoost (XGB), along with tree-based models such as Random Forest (RF) and Decision Tree (DT), significantly outperformed their counterparts.

As a result, we have selected these high-performing models for further refinement. Our next steps will involve fine-tuning their hyperparameters and employing a stacking approach to enhance overall predictive performance.

IX. Best Models Ensemble

Upon concluding the previous section, we observed that certain models consistently outperformed others. Consequently, these high-performing models have been selected for further refinement. To enhance their predictive capabilities, we employed Optuna for hyperparameter tuning of the chosen models. With the optimal hyperparameters identified, these models are now ready for the stacking phase, where the combination of their strengths will contribute to improved overall performance.

IX.I. Hyperparameter Tuning

To achieve optimal performance, Optuna was utilized to identify the best hyperparameters for each model. With these optimal parameters, we then fitted the training set, resulting in a significant enhancement of the model's performance metrics.

```
def objective_lightgbm(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 100, 1000),
        'max_depth': trial.suggest_int('max_depth', 3, 12),
        'num_leaves': 2 ** trial.suggest_int('max_depth', 3, 12),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.1),
        'subsample': trial.suggest_float('subsample', 0.5, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5,
            1.0)
    }
    model = LGBMClassifier(**params, device='gpu')
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    score = accuracy_score(y_test, y_pred)
    return score

study_lgbm = optuna.create_study(direction='maximize')
study_lgbm.optimize(objective_lightgbm, n_trials=5)

best_lgbm = LGBMClassifier(**study_lgbm.best_params, device='gpu')
```

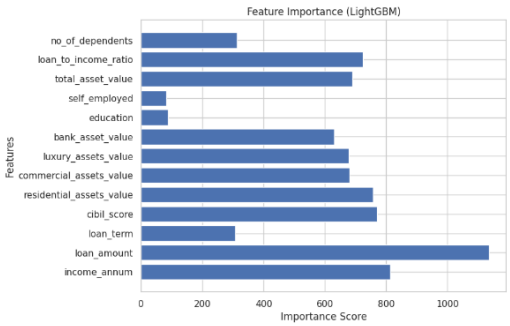
(As the example above other models were also fine tuned in similar fashion.)

IX.II. Feature Importance

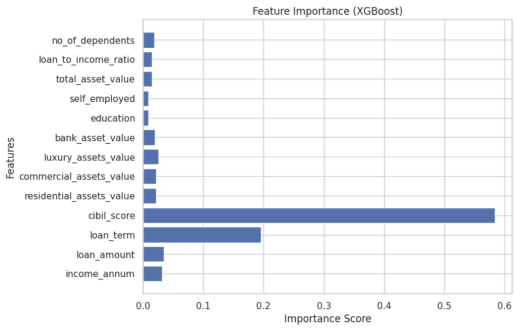
To deepen our understanding of the model's decision-making process, we also examined the feature importance of the various models employed. This analysis is crucial for several reasons. Firstly, it allows us to identify which features significantly influence loan approval or rejection, thereby enhancing our interpretability of the model's behavior. Understanding feature importance helps us to pinpoint the factors that contribute most to the predictions, enabling stakeholders to gain insights into the underlying mechanics of the model.

Upon analyzing the feature importance, several key features emerged as critical in the decision-making process: CIBIL Score, Loan Term, Loan Amount, and Annual Income. These features are vital for assessing an applicant's creditworthiness and significantly impact loan approval outcomes.

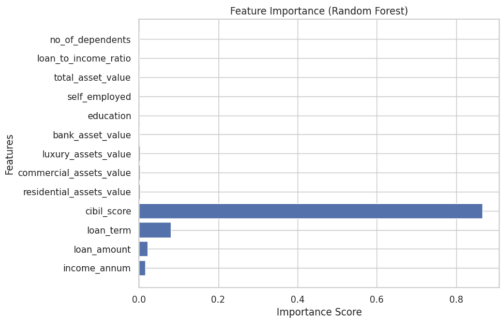
Moreover, identifying important features can guide future data collection and feature engineering efforts. By focusing on these influential variables, we can enhance model performance and ensure that our predictive tools are aligned with the factors that truly matter in the lending process. Additionally, this analysis fosters trust and transparency among stakeholders, as they can see the rationale behind the model's predictions, ultimately leading to more informed decision-making.



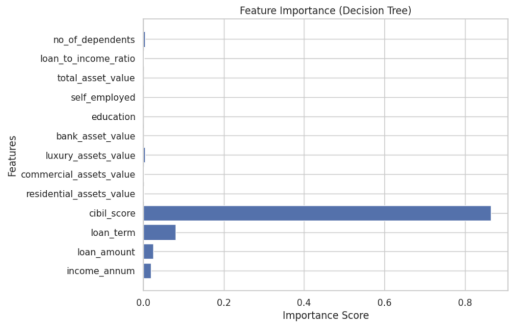
(a) LGBM



(b) XGBoost



(c) Random Forest



(d) Decision Tree

Figure 15: Overview of Feature Importance

IX.III. Stacking the Best Models

In this section, we implement a stacking model to leverage the strengths of multiple machine learning algorithms and enhance predictive performance. Stacking, also known as stacked generalization, involves training a new model (the meta-model) to combine the predictions of several base models. By using well-performing models such as LightGBM (LGBM), XGBoost (XGB), Random Forest (RF), and Decision Tree (DT) as base learners, we aim to create a more robust classifier that captures a wider array of patterns in the data. The final model employs Logistic Regression as the meta-learner, which will be trained on the predictions of the base models to yield the final output.

```
stacked_model = StackingClassifier(
    estimators=[
        ('lgbm', best_lgbm),
        ('xgb', best_xgb),
        ('rf', best_rf),
        ('dt', best_dt)
    ],
    final_estimator=LogisticRegression()
)

stacked_model.fit(X_train, y_train)

y_pred = stacked_model.predict(X_test)

y_prob = stacked_model.predict_proba(X_test)[:, 1] if hasattr(stacked_model,
    "predict_proba") else None
```

Evaluating the stacked model

After constructing and fitting the stacking classifier, we evaluate its performance on the test set using various metrics, including accuracy, precision, recall, F1 score, Matthews correlation coefficient (MCC), and ROC AUC score. These metrics provide a comprehensive view of the model's effectiveness in making predictions and its ability to generalize to unseen data. The results indicate how well the stacked model performs compared to individual models, showcasing the benefits of combining different algorithms to improve overall predictive capabilities. By utilizing stacking, we aim to achieve a more accurate and reliable model for loan approval prediction, which is crucial for financial decision-making processes.

```
Test Accuracy: 0.9775943396226415
Test Precision: 0.9776058602709546
Test Recall: 0.9775943396226415
Test F1 Score: 0.9775644377400321
Test MCC: 0.9530744727089963
Test ROC-AUC: 0.9986791190247593
```

Figure 16: Test Metrics (Stacked Model)

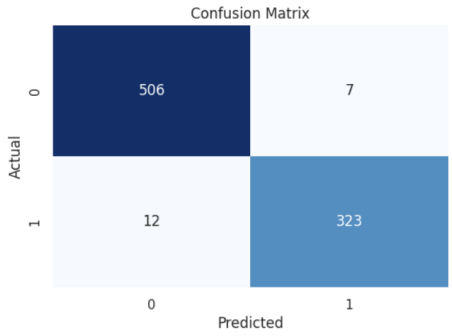
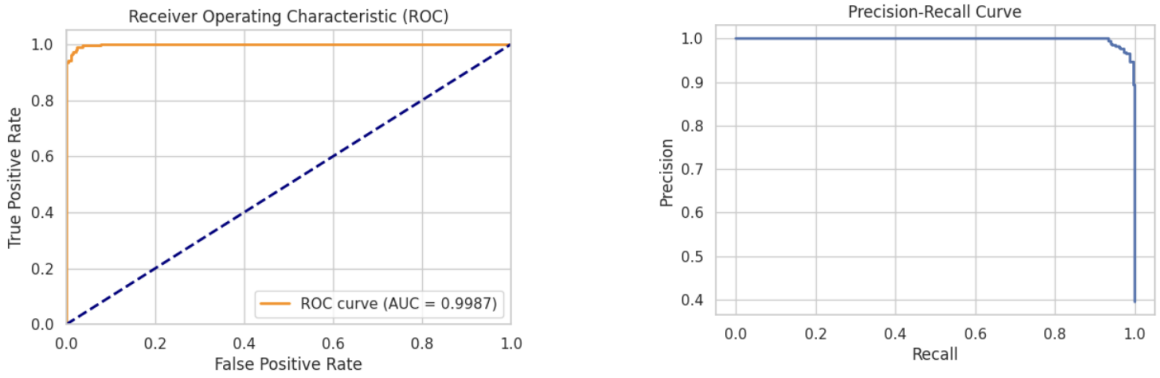


Figure 17: Confusion Matrix (Stacked Model)



(a) (b)

Figure 18: Roc and precision$_r ecall curve$

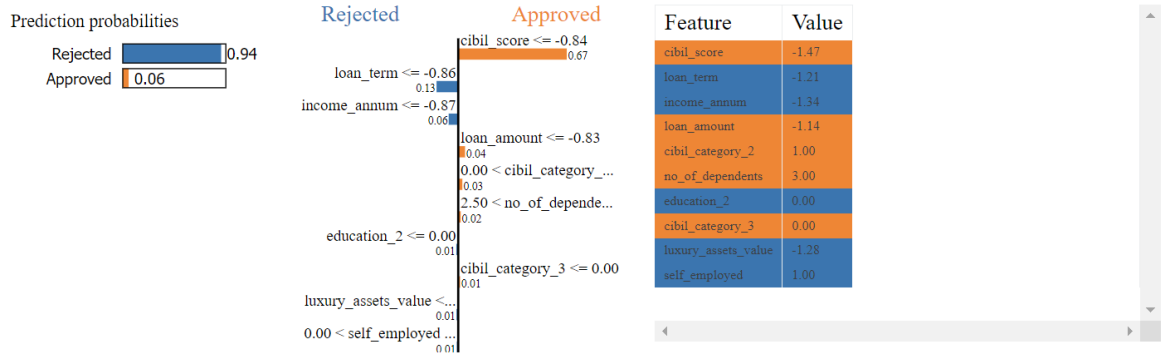
X. Model Explainability with LIME (Local Interpretable Model-agnostic Explanations)

In the context of loan approval prediction, understanding the rationale behind model decisions is crucial for both transparency and trust. To enhance interpretability, we employed LIME (Local Interpretable Model-agnostic Explanations), which provides insights into individual predictions by approximating the model locally with an interpretable model. (Ribeiro et al. 2016)

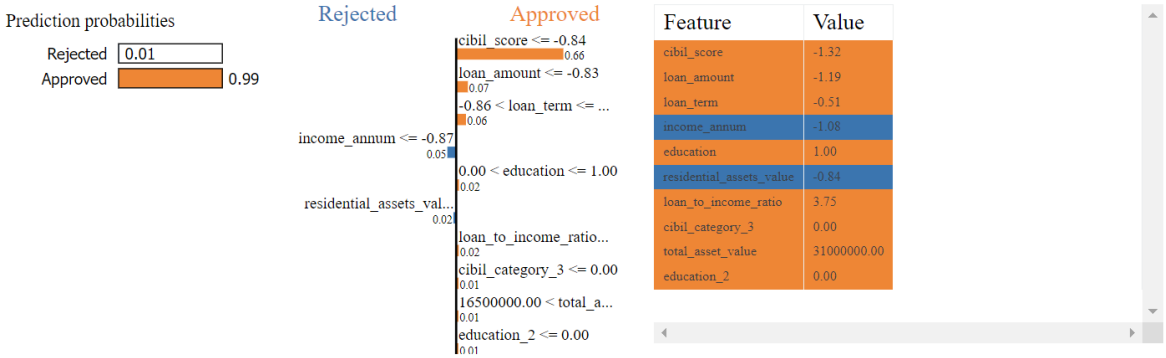
X.I. Overall Interpretability:

In addition to individual predictions, we utilized LIME to aggregate insights across multiple predictions. This overall interpretability helps to identify common patterns and feature impacts across the dataset.

By analyzing multiple loan applications, we observed that credit score, loan amount, and income consistently emerged as critical factors influencing loan approval decisions. This insight aligns with the expectations from domain knowledge and reinforces the model's reliability.



(a)



(b)

Figure 19: LIME Explanations

LIME provides visualizations that illustrate the contribution of each feature for selected predictions. These visualizations, typically in the form of bar charts, depict positive contributions (which favor approval) and negative contributions (which favor rejection). The visual representations effectively communicate the reasoning behind predictions, making it easier for stakeholders to comprehend the model's decision-making process.

Impact of Features:

The LIME explanations emphasized the importance of specific features:

- **Credit Score:** A high credit score significantly increases the likelihood of loan approval.

- **Loan Amount:** Larger loan amounts, particularly when not justified by income, negatively impact approval chances.
- **Income:** Higher income levels correlate positively with loan approvals, highlighting the importance of financial stability.

By integrating LIME into our analysis, we enhanced the interpretability of the stacked model used for loan approval prediction.

XI. Model Deployment

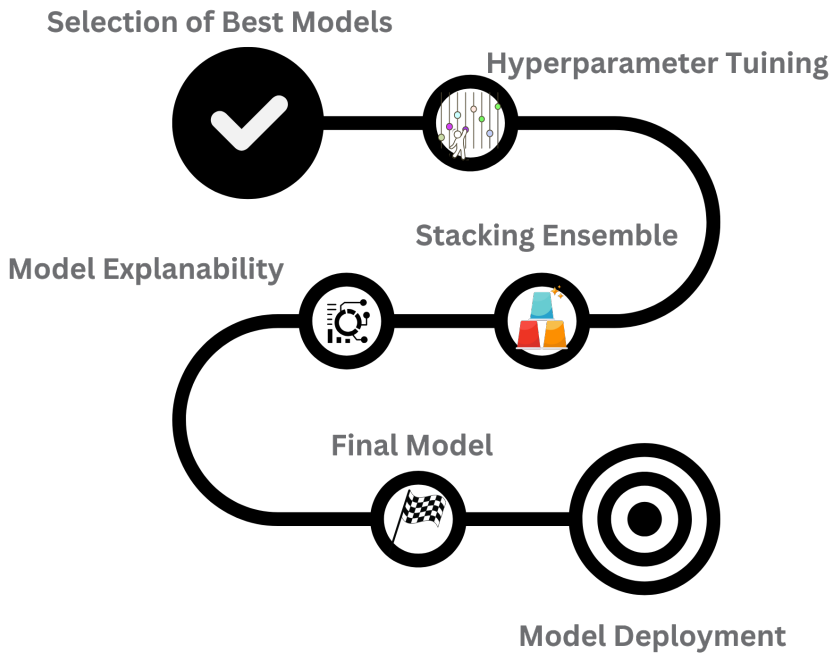


Figure 20: The proposed approach for model development

In the final step, we deployed the machine learning model to make it accessible for real-time predictions. To enable this, we extracted and saved the preprocessor, the individual best-performing models, and the stacked model using the **joblib** library. These components allow us to streamline predictions by processing user-provided data and passing it through the trained models.

```
import joblib

joblib.dump(pipeline, 'preprocessor.pkl')

joblib.dump(best_lgbm, 'best_lgbm.pkl')
joblib.dump(best_xgb, 'best_xgb.pkl')
joblib.dump(best_rf, 'best_rf.pkl')
joblib.dump(best_dt, 'best_dt.pkl')

joblib.dump(stacked_model, 'stacked_model.pkl')
```

For deployment, we used Flask as the web framework, complemented by Tailwind CSS to create a clean and responsive user interface. Flask manages the backend operations, handling incoming requests, processing data, and returning predictions, while Tailwind CSS ensures that the frontend provides a user-friendly and visually appealing experience. This setup allows users to easily input their data and receive quick, accurate predictions based on the model's insights.

Finally a GitHub[2] repository was created to host the complete application, featuring the deployed prediction model. The web service is seamlessly deployed on Render[3], making the model accessible and ready for use. This setup provides a structured, scalable, and easily maintainable deployment for the predictive application.

XII. Conclusion

This project effectively showcases the power of machine learning in transforming the loan approval process by accurately predicting the likelihood of a loan application's approval or rejection. By leveraging a structured approach that includes comprehensive data preprocessing and meticulous feature engineering, the model is prepared to handle diverse applicant profiles and complex patterns within the data. Robust machine learning models, such as XGBoost, LightGBM, and Random Forest, are applied to maximize predictive accuracy and ensure reliability in the results.

The deployment of this model through a Flask-based web application, styled with Tailwind CSS, provides a user-friendly, intuitive interface for end-users, enabling seamless access to real-time predictions. Financial institutions can integrate this application into their operations, streamlining the loan assessment process by significantly reducing the time and operational costs involved. Moreover, this predictive approach supports a more consistent and objective decision-making framework, reducing human biases and enhancing transparency.

Beyond improving the efficiency and fairness of loan approvals, this project can be scaled and adapted for other predictive purposes within the financial industry, such as credit scoring or fraud detection. By enhancing the accuracy and speed of loan approvals, the model holds potential to improve customer satisfaction and enable more robust risk management, offering a forward-thinking solution that aligns with the goals of modern financial services.

XIII. Future Work

While this project demonstrates promising results, there are several areas for further enhancement:

1. **Model Optimization:** Additional hyperparameter tuning and experimentation with advanced ensemble methods could further improve model accuracy and robustness.
2. **Data Updates:** Periodic retraining with new data would help maintain the model's accuracy over time as lending trends and financial patterns evolve.
3. **Deployment Scalability:** Future deployments could explore cloud-based solutions like AWS or GCP for improved scalability and performance in handling larger volumes of real-time requests.
4. **Security Compliance:** Enhancing data security measures and ensuring compliance with financial regulations would be critical in a production environment.

Expanding in these areas will not only improve the model's utility and reliability but also pave the way for more sophisticated, scalable loan prediction systems in the future.

[2]Link to Repo: `https://github.com/bses7/Loan-Approval-Model`
[3]Link to App: `https://loan-approval-model.onrender.com/`

Bibliography

Aphale, A. S. & Shinde, S. R. (2020), 'Predict loan approval in banking system machine learning approach for cooperative banks loan approval', *International Journal of Engineering Trends and Applications (IJETA)* **9**(8).

Arun, K., Ishan, G. & Sanmeet, K. (2016), 'Loan approval prediction based on machine learning approach', *IOSR J. Comput. Eng* **18**(3), 18–21.

Kadam, A. S., Nikam, S. R., Aher, A. A., Shelke, G. V. & Chandgude, A. S. (2021), 'Prediction for loan approval using machine learning algorithm', *International Research Journal of Engineering and Technology (IRJET)* **8**(04).

Pušnik, M., Kous, K., Godec, A. & Šumak, B. (2019), Process evaluation and improvement: A case study of the loan approval process1, *in* 'Proceedings of the SQAMIA 2019: 8th Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications', pp. 1613–0073.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016), " why should i trust you?" explaining the predictions of any classifier, *in* 'Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining', pp. 1135–1144.

Wing, J. M. (2019), 'The data life cycle', *Harvard Data Science Review* **1**(1), 6.