Student ID: 24148501

Module Code: CMP6221

Module Title: Computing for AI

Coursework Title: Loan Approval Prediction User Guide

Acknowledgments

I would like to express my sincere gratitude to my academic advisor and faculty members for their invaluable guidance and support throughout the development of this project. A special thanks to my peers and family for their encouragement and feedback, which helped refine my ideas. Additionally, I am grateful to Kaggle for providing the dataset used in this project, which served as the foundation for building the loan approval prediction model. This project would not have been possible without the tools, libraries, and resources made freely available by the Python and open-source communities.

Executive Summary

This report presents the development of a machine learning-based loan approval prediction system aimed at streamlining the decision-making process in financial institutions. The system leverages a dataset from Kaggle, consisting of applicant demographic, financial, and loan-related features. Through data preprocessing, model selection, and evaluation, various machine learning algorithms, including Logistic Regression, Random Forest, XGBoost, and others, were implemented and assessed based on key performance metrics such as accuracy, precision, recall, and F1 score.

The Logistic Regression model was ultimately selected for deployment due to its strong validation accuracy, balanced metrics, and interpretability, aligning well with financial business needs. A user-friendly interface was developed to allow users to input loan-related data and receive real-time loan status predictions. The system provides transparent insights into the approval process, minimizing risks and enhancing customer experience.

This report also discusses the limitations of the current implementation, including dataset constraints and potential biases, along with recommendations for future improvements. The project demonstrates how machine learning can be effectively applied to automate financial decisions while maintaining accuracy and fairness.

Table of Contents

Table of Figures

Table of Tables

Table of Equation

1. Introduction

1.1. Problem Statement and Objective

Loan approval is a critical process within financial institutions as it directly impacts their profitability and stability. The ability to make accurate predictions in this area reduces financial risks, increases operational efficiency, and enhances the customer experience. However, traditional manual evaluations are often prone to human error and can be time-consuming, making it challenging for banks to consistently meet both operational demands and customer expectations (Kokru et al., 2022).

In recent years, the application of machine learning has proven invaluable in automating and refining decision-making processes in finance. By leveraging algorithms to analyze extensive customer data, banks can more effectively predict loan repayment outcomes, which is instrumental in managing risk and prioritizing loan approvals (Mukri, 2023). This project aims to develop a machine learning-based loan approval system that automates decision-making, allowing users to input data and receive real-time predictions on loan eligibility, enhancing the efficiency and reliability of loan processing.

1.2. Dataset Overview and Significance

The dataset, sourced from Kaggle, includes 13 critical features that capture a variety of information about loan applicants, including demographics, financial metrics, and specific loan details (Kaggle, 2023). The primary target variable, **Loan_Status**, indicates whether a loan is approved (1) or not (0), enabling the training of models focused on loan eligibility predictions.
Key Features:

- **Gender**: Male/Female.
- **Married**: Yes/No.
- **ApplicantIncome**: Monthly income of the applicant.
- **LoanAmount**: Loan requested (in thousands).
- **Credit_History**: Indicates credit history (1 = Yes, 0 = No).

This dataset is essential for developing a predictive model that evaluates loan eligibility based on applicant features.

1.3. Key Features and Target Variable

The dataset includes demographic, financial, and loan-related features:

- **Demographic**: Gender, Marital Status, Dependents, Education, Employment Type.
- **Financial**: Applicant Income, Coapplicant Income, Credit History.
- **Loan Details**: Loan Amount, Loan Term, Property Area.

The target variable, Loan_Status, predicts loan eligibility (1 = Approved, 0 = Not Approved). These features help models identify patterns to make accurate predictions aligned with financial requirements.

1.4. Scope and Limitations of the Project

The scope includes:

- **Data Preprocessing**: Handling missing values, encoding features, and balancing classes.
- **EDA**: Visualizing feature distributions and correlations.
- **Model Training and Evaluation**: Implementing algorithms like Logistic Regression, Random Forest, and XGBoost.
- **User Interaction**: Accepting input from users and generating predictions.

Limitations:

1. **Dataset Size and Quality**: May not cover all real-world scenarios.
2. **Feature Simplicity**: Lacks complex financial data like credit scores or interest rates.
3. **Generalization Issues**: Models may struggle with unseen data.
4. **Bias in Data**: Inherent biases (e.g., gender bias) may influence predictions.

2. System Design Overview

2.1. Project Architecture and Workflow Overview

The project follows a modular approach, with each component responsible for distinct tasks to maintain scalability and ease of use.

Workflow:

1. **Data Preprocessing**: Load the dataset using DataHandler, handle missing values, encode categorical features, and balance classes.
2. **EDA**: Visualize feature distributions and relationships.
3. **Model Training and Evaluation**: Train models with ModelManager and evaluate them using metrics like accuracy and precision.
4. **User Interaction**: Collect user input via UserInteractionManager and pass it to the trained model for prediction.
5. **Prediction Output**: Display whether the loan is approved or not.

This structure ensures clear separation of tasks, making each step manageable and modular.

2.2. UML Diagrams

The UML diagrams provide a clear representation of the system's structure and workflow.

2.2.1. UML Class Diagram:

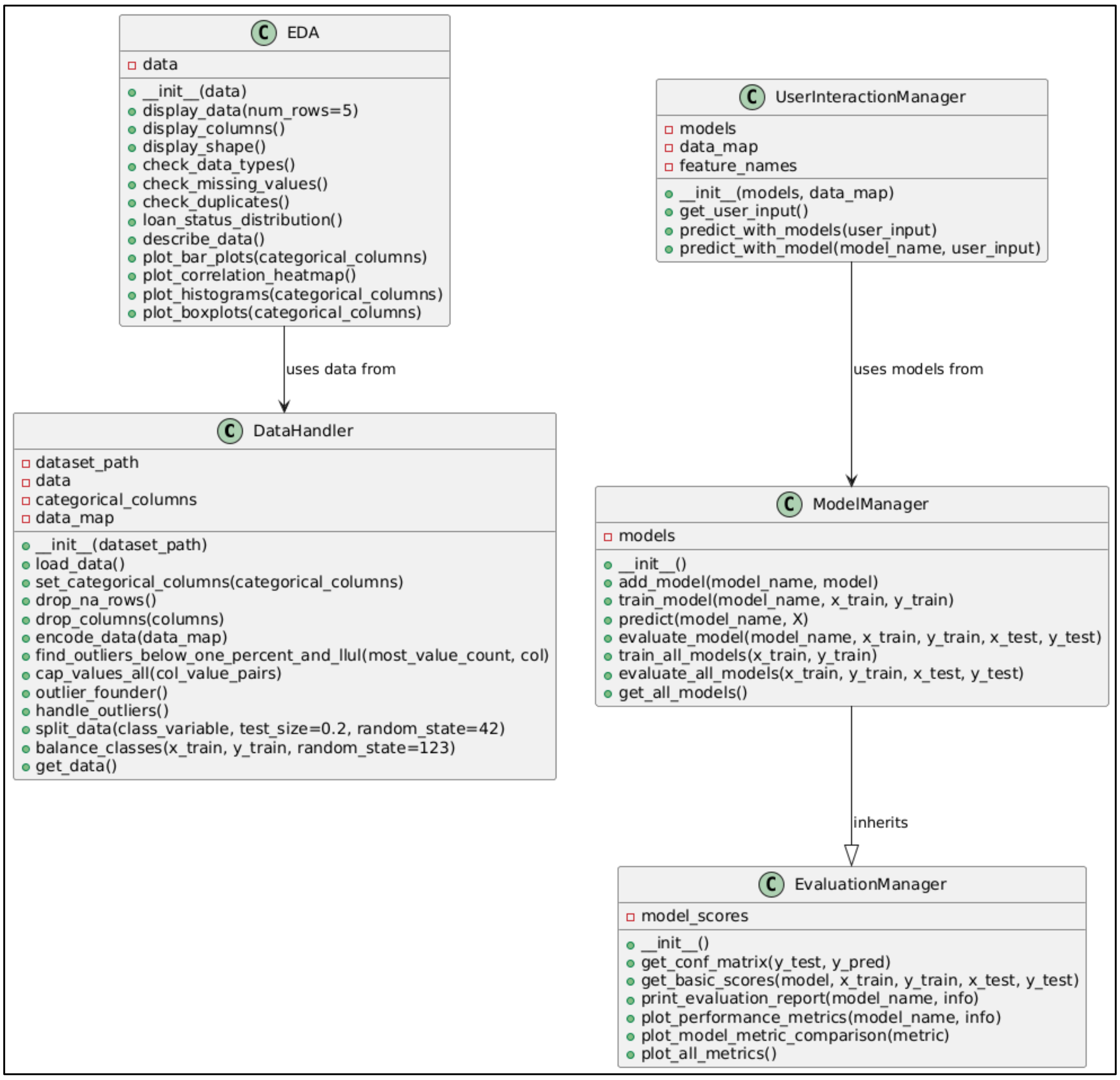

EDA
data
__init__(data)
display_data(num_rows=5)
display_columns()
display_shape()
check_data_types()
check_missing_values()
check_duplicates()
loan_status_distribution()
describe_data()
plot_bar_plots(categorical_columns)
plot_correlation_heatmap()
plot_histograms(categorical_columns)
plot_boxplots(categorical_columns)
uses data from
DataHandler
dataset_path
data
categorical_columns
data_map
__init__(dataset_path)
load_data()
set_categorical_columns(categorical_columns)
drop_na_rows()
drop_columns(columns)
encode_data(data_map)
find_outliers_below_one_percent_and_llul(most_value_count, col)
cap_values_all(col_value_pairs)
outlier_founder()
handle_outliers()
split_data(class_variable, test_size=0.2, random_state=42)
balance_classes(x_train, y_train, random_state=123)
get_data()
UserInteractionManager
models
data_map
feature_names
__init__(models, data_map)
get_user_input()
predict_with_models(user_input)
predict_with_model(model_name, user_input)
uses models from
ModelManager
models
__init__()
add_model(model_name, model)
train_model(model_name, x_train, y_train)
predict(model_name, X)
evaluate_model(model_name, x_train, y_train, x_test, y_test)
train_all_models(x_train, y_train)
evaluate_all_models(x_train, y_train, x_test, y_test)
get_all_models()
inherits
EvaluationManager
model_scores
__init__()
get_conf_matrix(y_test, y_pred)
get_basic_scores(model, x_train, y_train, x_test, y_test)
print_evaluation_report(model_name, info)
plot_performance_metrics(model_name, info)
plot_model_metric_comparison(metric)
plot_all_metrics()

Figure 1 Class Diagram

The UML class diagram visually represents the key components of the project and their relationships. It demonstrates how various modules interact to facilitate loan prediction:

- **EDA Class**: Handles data exploration by generating plots and providing data insights, such as summary statistics, correlations, and data types.
- **DataHandler Class**: Manages data preprocessing tasks, including loading data, handling missing values, encoding categorical features, and managing outliers.
- **ModelManager Class**: Responsible for managing machine learning models, adding models, training them, making predictions, and evaluating their performance.
- **EvaluationManager Class**: Inherited by the ModelManager class to provide evaluation metrics and confusion matrix functionalities.

- **UserInteractionManager Class**: Collects user input, encodes the input, and uses trained models to predict loan status. It interacts with the ModelManager and DataHandler for seamless user interaction.

This modular structure ensures scalability, maintainability, and clear separation of concerns, making it easy to enhance individual components.

2.2.2. UML Sequence Diagram:

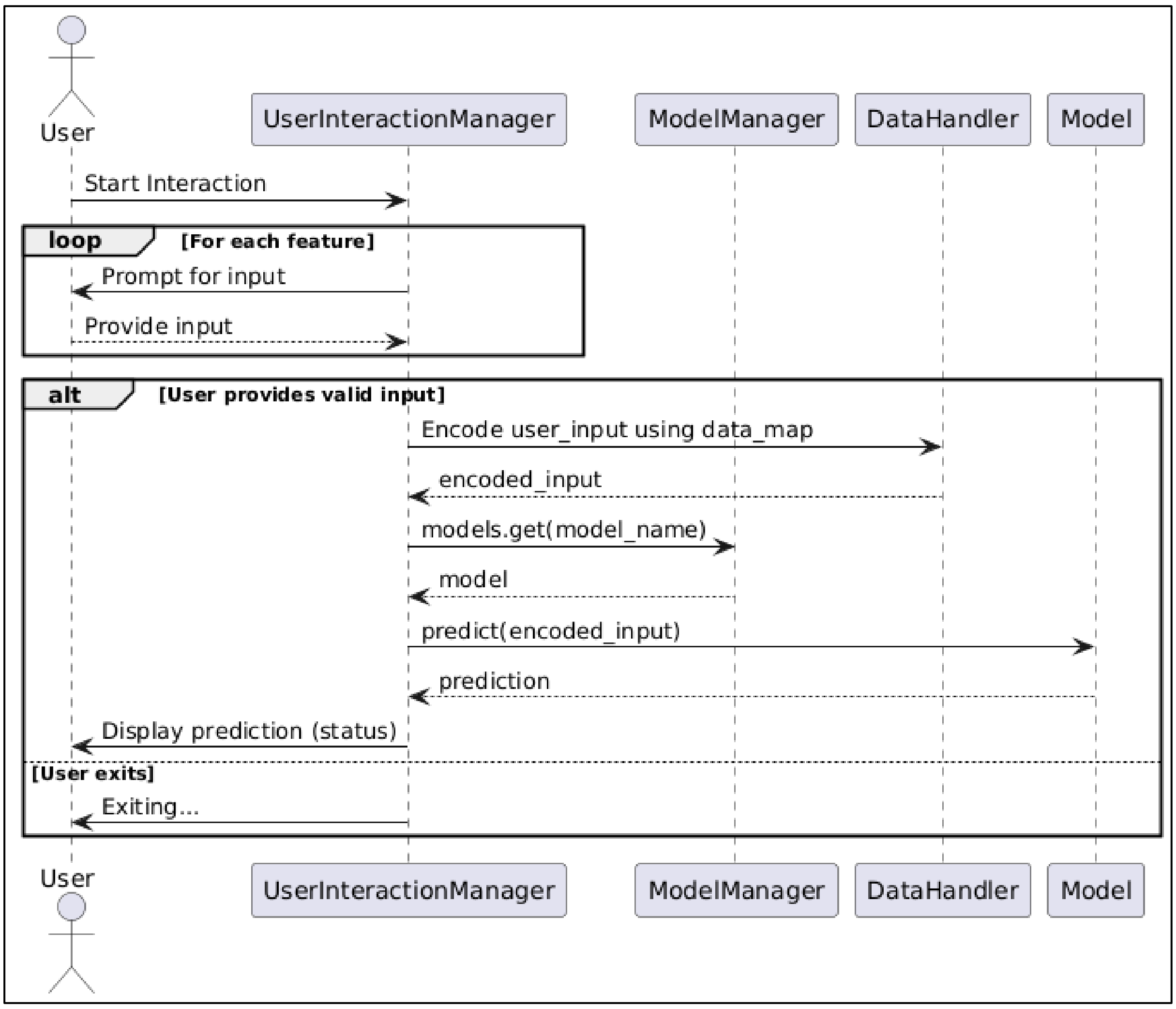


Figure 2 Sequence Diagram

The UML sequence diagram illustrates the step-by-step flow of user interaction within the loan prediction system:

1. **User Interaction Begins**: The user initiates the interaction by starting the loan application prediction.
2. **Feature Input Loop**: For each feature, the system prompts the user for input, and the user provides responses.

3. **Input Validation and Encoding**: Once all inputs are collected, they are encoded using the data_map to ensure compatibility with the models.
4. **Model Prediction**: The encoded input is passed to the ModelManager, which retrieves the selected model and makes a prediction.
5. **Display Prediction**: The system displays the predicted loan status (approved or not approved) based on the model's output.
6. **Exit Process**: The user exits the system, concluding the interaction.

This diagram highlights the real-time interaction between the user and various modules, emphasizing how input flows through the system to generate predictions efficiently.

2.2.3. Flowchart Diagram:

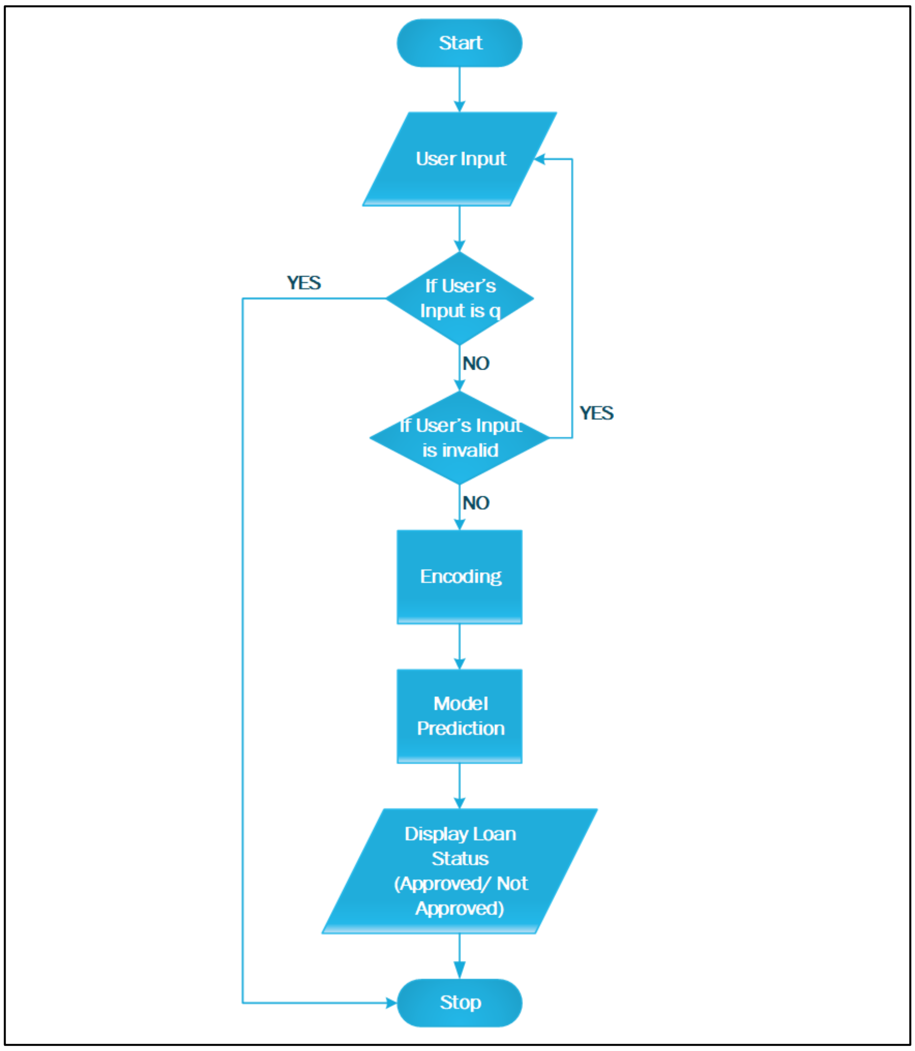


Figure 3 Flowchart

The flowchart outlines the process for a loan approval prediction system, starting with user input. If the user inputs **'q'**, the process terminates immediately. Otherwise, the system checks if the input is valid;

if invalid, it prompts the user to re-enter the data. Once valid input is provided, it is **encoded** into a numerical format suitable for the machine learning model. The model then predicts the loan status, displaying the result as either **“Approved”** or **“Not Approved.”** The process concludes at the **stop** point, ensuring a streamlined workflow with input validation and clear termination handling.

2.3. Technologies and Libraries Used

This project leverages the following technologies and tools:

Development Tools:

- **Python**: Programming language.
- **Google Colab / Jupyter Notebook**: Development environment.
- **Kaggle**: Dataset source.

Python Libraries:

1. **pandas**: Data manipulation.
2. **numpy**: Numerical computations.
3. **matplotlib** and **seaborn**: Data visualization.
4. **scikit-learn**: Model implementation and evaluation.
5. **imblearn**: Handles class imbalance with RandomOverSampler.
6. **XGBoost, LightGBM, CatBoost**: Advanced boosting algorithms.

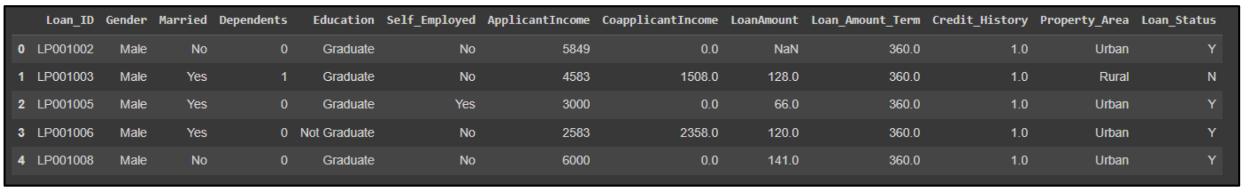
3. Exploratory Data Analysis (EDA)

3.1 Initial Dataset Overview

Exploratory Data Analysis (EDA) begins by exploring the dataset's structure, size, and completeness to understand the initial data landscape. This process helps identify potential issues such as missing values or inconsistencies that need to be addressed during preprocessing.

3.1.1 Displaying the First Few Rows of the Dataset

The initial rows of the dataset provide a preview of the data entries and allow us to verify the data loading process.



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

Table 1 Dataset Preview

This preview confirms that the data contains key features such as Gender, Married, ApplicantIncome, and Loan_Status.

3.1.2 Column Names and Dataset Shape

The dataset consists of 614 rows and 13 columns before any preprocessing. These columns capture both categorical data (like Gender and Married) and numerical data (like ApplicantIncome and LoanAmount). The presence of both types of data highlights the need for encoding and handling different formats during preprocessing.

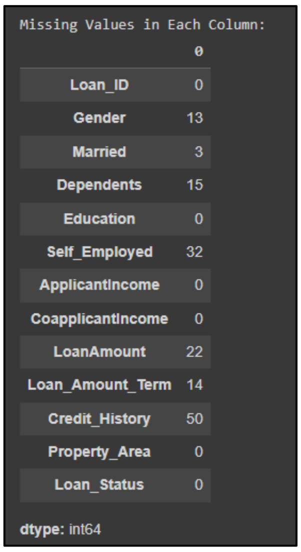
3.1.3 Data Types and Missing Values

Feature	Data Type
Loan_ID	object
Gender	object
Married	object
Dependents	object
Education	object
Self_Employed	object
ApplicantIncome	int64
CoapplicantIncome	float64
LoanAmount	float64
Loan_Amount_Term	float64
Credit_History	float64
Property_Area	object
Loan_Status	object

Table 2 EDA Feature Initial Data Type

The dataset contains both categorical and numerical features:

- **Categorical Data**: Loan_ID, Gender, Married, Dependents, Education, Self_Employed, Property_Area, Loan_Status, Credit_History
- **Numerical Data**: ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term,



```
Missing Values in Each Column:
                    0
Loan_ID             0
Gender             13
Married             3
Dependents         15
Education           0
Self_Employed      32
ApplicantIncome     0
CoapplicantIncome   0
LoanAmount         22
Loan_Amount_Term   14
Credit_History     50
Property_Area       0
Loan_Status         0

dtype: int64
```

Figure 4 Missing Values

A detailed analysis revealed missing values in the following columns:

- Gender: 13
- Married: 3
- Dependents: 15
- Self_Employed: 32
- LoanAmount: 22
- Loan_Amount_Term: 14
- Credit_History: 50

A detailed analysis revealed missing values in several columns, which is crucial to address to prevent biases in machine learning models. Approaches for handling missing data, such as k-nearest neighbor and missForest, are effective in minimizing the negative impact on model performance (Emmanuel et al., 2021). Methods like these ensure that the dataset remains suitable for training and yields accurate predictions (Gond et al., 2021).

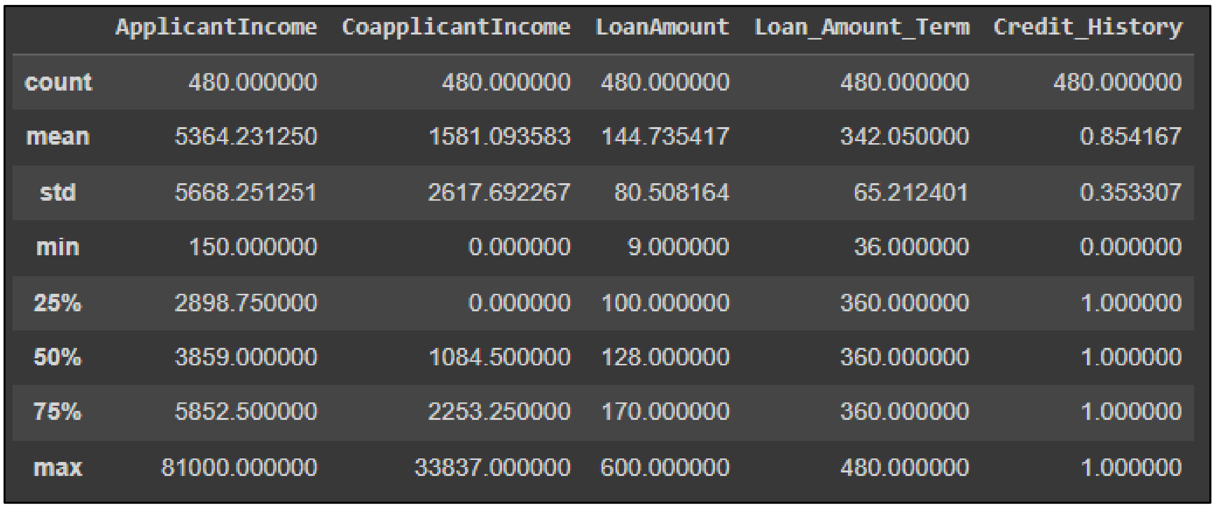
3.1.4 Identifying Duplicate Entries

The dataset was checked for duplicate rows by dropping the Loan_ID column which always unique. No duplicate entries were found, ensuring the uniqueness of each loan application. This step is essential to prevent redundant data from affecting model performance.

3.2. Statistical Summary

3.2.1 Generating Summary Statistics for Numerical Features

The summary statistics for numerical variables offer insights into the distribution and scale of the data.



	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	480.000000	480.000000	480.000000	480.000000	480.000000
mean	5364.231250	1581.093583	144.735417	342.050000	0.854167
std	5668.251251	2617.692267	80.508164	65.212401	0.353307
min	150.000000	0.000000	9.000000	36.000000	0.000000
25%	2898.750000	0.000000	100.000000	360.000000	1.000000
50%	3859.000000	1084.500000	128.000000	360.000000	1.000000
75%	5852.500000	2253.250000	170.000000	360.000000	1.000000
max	81000.000000	33837.000000	600.000000	480.000000	1.000000

Table 3 Statistical Summary of Numerical Features

This statistical overview indicates wide variations in incomes and loan amounts, with some outliers present, especially in ApplicantIncome. Additionally, a significant number of applicants have zero coapplicant income, suggesting that they applied alone.

3.2.2 Loan Status Distribution

The target variable Loan_Status is binary, indicating whether the loan was approved (1) or not (0). A quick analysis of the distribution is as follows:

- **Approved Loans (1)**: 332 (54%)
- **Rejected Loans (0)**: 148 (24%)

This initial analysis reveals a class imbalance, where approved loans significantly outnumber rejected ones. If left unaddressed, the imbalance could lead to biased model predictions, favoring the majority class. Therefore, techniques such as oversampling are essential to ensure the model learns effectively from both classes.

3.3. Visual Analysis

3.3.1 Bar Plots of Categorical Variables

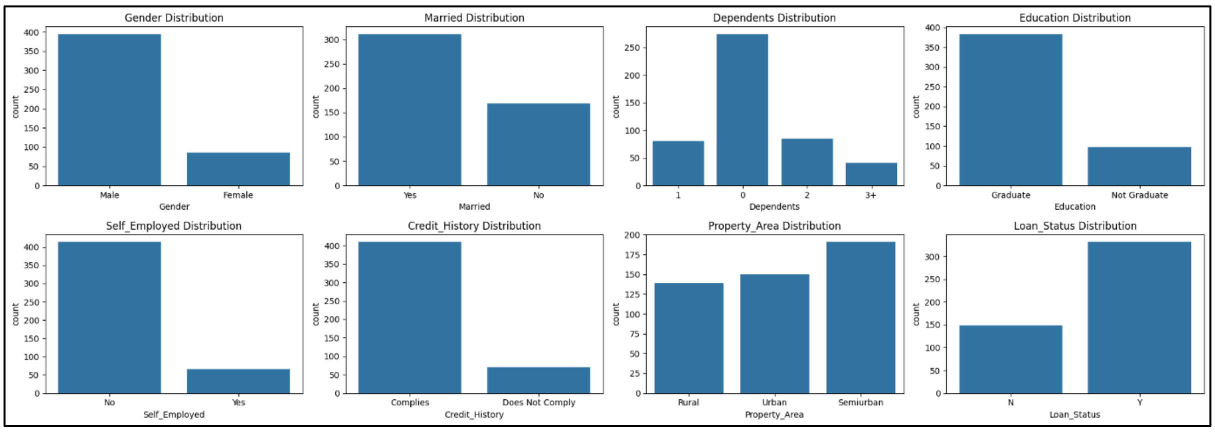


Figure 5 Bar Plot Categorical Variables

The bar plots highlight that most applicants are male and married, with a large share having no dependents. The majority are graduates, and a significant portion is not self-employed. Around 85% of applicants have a positive credit history, which aligns with its importance for loan approval. Property areas are distributed evenly, with a slightly higher share from semiurban regions. Regarding loan status, approximately 69% of loans are approved and 31% rejected, reflecting a potential class imbalance that requires attention during modeling.

3.3.2 Correlation Heatmap of Numerical Features

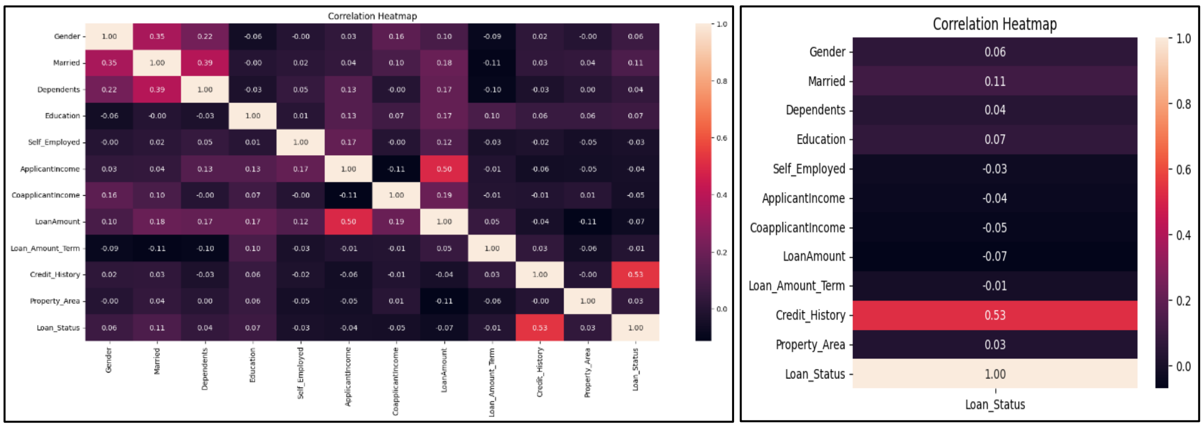


Figure 6 Correlation Heat Map

The correlation heatmaps provide insights into the relationships between numerical features. Key observations include:

- **Credit History** shows the strongest positive correlation with the target variable **Loan_Status** (0.53). This indicates that having a positive credit history significantly influences loan approval decisions.
- **LoanAmount** and **ApplicantIncome** exhibit moderate correlation (0.50), suggesting that applicants with higher income tend to request larger loans.
- Most other features show weak correlations with **Loan_Status**, indicating that no single demographic or financial feature alone strongly determines loan approval, aside from credit history.
- **Loan_Amount_Term** and **Applicant/Coapplicant Income** have minimal correlations with loan status, reflecting their limited predictive power.

These insights guide feature selection and suggest that **Credit History** will play a critical role in the model, while other weakly correlated features may require further engineering or may be less impactful for predictions.

3.3.3 Histograms of Numerical Variables

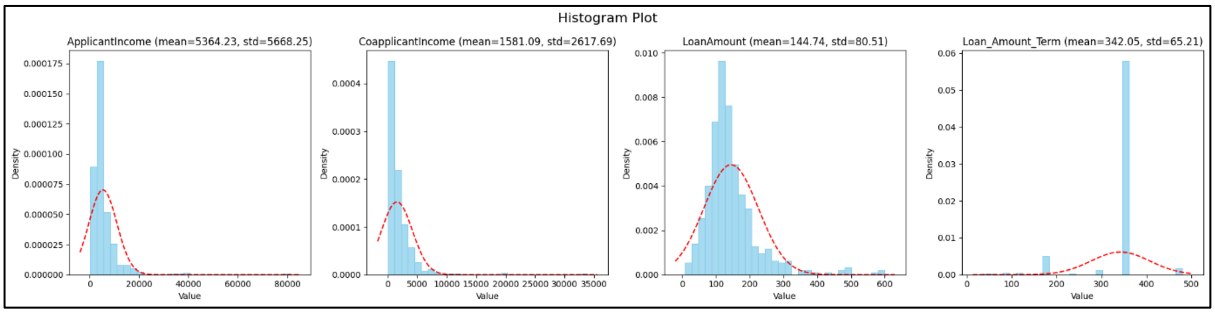


Figure 7 Histogram of Numerical Variables

The histograms highlight the distribution of key numerical variables in the dataset:

- **ApplicantIncome**:

  Displays a right-skewed distribution with a mean of 5364.23 and a few extremely high-income outliers, indicating that most applicants earn below 20,000, with a small number earning much higher.
- **CoapplicantIncome**:

  Similarly skewed, with most coapplicants earning low or no income (mean 1581.09), confirming that many applicants apply without a coapplicant. A few outliers with very high incomes are present.
- **LoanAmount**:

  Approximates a normal distribution but with a slight skew and outliers. The mean loan amount is 144.74, with most requests ranging between 100 to 250 thousand.
- **Loan_Amount_Term**:

  The term distribution is highly concentrated around 360 months (30 years), indicating that the majority of applicants seek long-term loans.

These histograms reveal skewness and outliers, particularly in income-related features, highlighting the need for outlier treatment to ensure stable model performance.

3.3.4 Boxplots for Outlier Detection

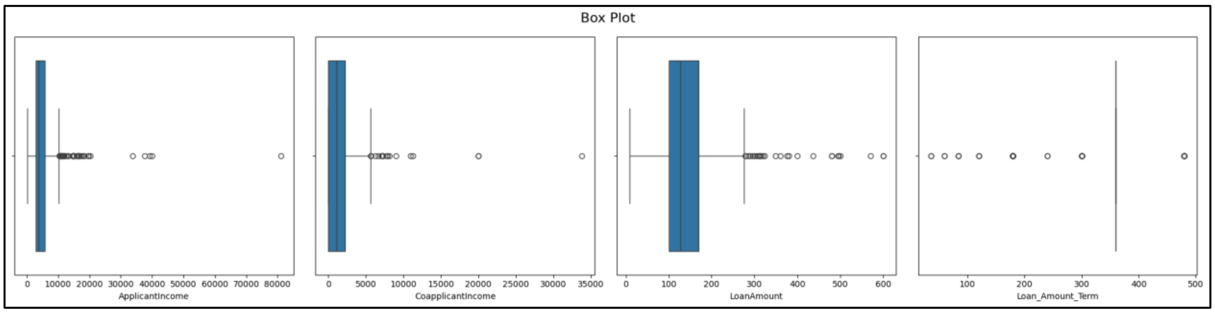


Figure 8 Boxplots

The boxplots help identify **outliers** across the key numerical variables:

- **ApplicantIncome**:
  Several extreme outliers are present, with values exceeding 50,000, reflecting a small number of very high-income applicants.
- **CoapplicantIncome**:
  Most coapplicant incomes are clustered below 10,000, but there are significant outliers extending beyond 25,000, likely from joint applications with large combined incomes.
- **LoanAmount**:
  Outliers are observed with loan amounts exceeding 300, indicating a few very large loan requests, which could skew model predictions.
- **Loan_Amount_Term**:
  Outliers are minimal, but a few terms shorter than 100 months are present, indicating some shorter-term loans.

These outliers, particularly in income and loan amounts, need to be handled carefully to avoid skewing the model. Strategies such as capping extreme values or using robust models can mitigate their impact.

4. Data Handling and Preprocessing

4.1. DataHandler Class: Overview and Responsibilities

The DataHandler class ensures that the dataset is properly cleaned and preprocessed for analysis. Key tasks include removing missing values, encoding categorical data, handling outliers, and balancing the classes to prepare the dataset for training machine learning models.

4.1.1. Data Loading and Cleaning

- Data Loading:
  The load_data() method loads the dataset from a CSV file into a pandas DataFrame.
- Handling Missing Values:
  Dropped rows with missing values in Gender, Married, LoanAmount, and Credit_History.
  Ensures the dataset contains only complete entries, avoiding errors during model training.

4.1.2. Dropping Unnecessary Columns

- Loan_ID was dropped using the drop_columns() method.
  This column serves only as an identifier and provides no predictive value for the loan approval outcome.

4.1.3 Encoding Categorical Variables

The following categorical features were encoded into numerical values using the encode_data() method:

- **Gender**: Male = 1, Female = 0

- **Married**: Yes = 1, No = 0
- **Dependents**: '0' = 0, '1' = 1, '2' = 2, '3+' = 3
- **Education**: Graduate = 1, Not Graduate = 0
- **Self_Employed**: Yes = 1, No = 0
- **Property_Area**: Semiurban = 1, Urban = 2, Rural = 0
- **Loan_Status**: Y = 1, N = 0

Converting categorical data into numerical form ensures compatibility with machine learning models.

4.1.4. Handling Outliers

The boxplots reveal the presence of outliers across several numerical features. Key insights include:

- **ApplicantIncome**:
  Although a few extremely high values were observed, they were capped at 10,283.12, with the lower bound set to -1531.87 to limit the impact of outliers.
- **CoapplicantIncome**:
  Several extreme values, such as 20,000, 33,837, and 10,968, were detected. These outliers were capped within the range of 0 to 5633.12 to prevent them from skewing the model's predictions.
- **LoanAmount**:
  The boxplot reveals large loan requests beyond 300, which were capped at 275, ensuring stability in the dataset without distorting model training.
- **Loan_Amount_Term**:
  A few shorter loan terms—such as 120, 84, 60, and 36 months—were capped to 360 months, the standard loan term, to align with the majority of the data.

Capping these outliers ensures that the extreme values do not disproportionately influence the model's learning process, leading to more stable and reliable predictions.

4.1.5. Data Splitting and Class Balancing

- **Data Splitting**:
  - The dataset was split into 80% training and 20% testing sets using the split_data() method.
  - Ensures models are evaluated on unseen data, preventing overfitting.

- **Class Balancing**:
  - The balance_classes() method used RandomOverSampler to address the imbalance in the target variable Loan_Status.
  - Both the approved and rejected loan classes now have 264 instances each.
  - Balanced data ensures fair learning for both outcomes, reducing bias.

6. Machine Learning Models and Implementation

5.1 Overview of Models Used and Justification

A variety of models were selected to explore different classification algorithms. These models span across linear models, tree-based methods, probabilistic classifiers, and ensemble techniques, each with distinct advantages.

- **Logistic Regression**
  Logistic Regression, a linear classifier, is straightforward and interpretable, making it a good baseline for binary classification tasks. It offers quick predictions and works well when there is a linear relationship between the input features and the target variable.
- **K-Nearest Neighbors (KNN)**
  KNN predicts the class of a sample based on the majority class among its nearest neighbors. It is useful for non-linear data but can struggle with large datasets, as it requires computing the distance between all points during prediction.
- **Support Vector Machine (SVM)**
  SVM finds the optimal hyperplane to separate classes in a high-dimensional space. Although computationally intensive, it works well for complex datasets. The probability=True parameter allows for probabilistic predictions, improving interpretability.
- **Naive Bayes**
  This probabilistic model assumes feature independence, which simplifies computation. While it is fast and performs well with small datasets, its performance may be limited by the strong independence assumption.
- **Decision Tree Classifier**
  The Decision Tree splits data into branches based on feature conditions, making it intuitive and interpretable. However, it is prone to overfitting, especially with small datasets.

5.2 Ensemble Models and Their Advantages

Ensemble learning combines multiple models to improve performance by reducing bias and variance. In this project, several ensemble methods were implemented.

- **Random Forest Classifier**
  Random Forest aggregates multiple decision trees to reduce overfitting and variance. It performs well on both balanced and imbalanced datasets by assigning weights to minority classes. Additionally, it handles missing values and complex feature interactions.
- **Gradient Boosting Models**
  Gradient Boosting builds models sequentially, correcting errors made by previous models. Advanced implementations used in this project include:

    - **XGBoost**: Known for speed and performance, XGBoost uses gradient boosting with regularization, making it less prone to overfitting.
    - **LightGBM**: Efficient with large datasets and sparse data, LightGBM offers faster training compared to traditional boosting models.
    - **CatBoost**: CatBoost is particularly advantageous when working with categorical data, as it reduces preprocessing requirements and delivers high accuracy.
- **GradientBoostingClassifier**
  This general gradient boosting implementation builds weak learners in sequence, focusing on correcting previous errors. It is versatile and effective for predictive tasks where boosting improves the model's learning.

5.3 Handling Class Imbalance in Models

Addressing class imbalance is essential in loan approval prediction, as the majority of loans may be approved, leaving rejected loans underrepresented. Imbalanced data can bias predictions, making models less effective. Each model has different capabilities for managing such imbalance:

- **Random Forest and Decision Tree:**
  These models naturally handle imbalance by adjusting class weights. Random Forest, through bootstrap sampling, ensures that both classes are well-represented during training.
- **Gradient Boosting Models (XGBoost, LightGBM, CatBoost):**
  These models support hyperparameters like scale_pos_weight to assign more weight to the minority class, improving their predictive power for rejected loans.
- **SVM**:
  The performance of SVM improves with class weighting techniques or oversampling the minority class. These adjustments ensure better balance in predictions.

- **Naive Bayes:**
  This model calculates the probability of each class, ensuring fair treatment even with imbalanced datasets. However, its simplifying assumptions limit its predictive capacity with complex data.
- **KNN:**
  KNN struggles with class imbalance, as the majority class neighbors tend to dominate predictions. Addressing this issue requires balancing the dataset using oversampling techniques like RandomOverSampler before training.

5.4 ModelManager Class: Managing and Training Models

The ModelManager class plays a crucial role in managing multiple models throughout the training and evaluation phases. It allows for systematic addition, training, and evaluation of models.

- **Adding and Training Models**
  Each model is registered using the add_model() method. Training is conducted via the train_model() method, ensuring consistent use of the same training data across models.
- **Training All Models Simultaneously**
  To streamline the training process, the train_all_models() method automates the training of all registered models, providing consistency in experimentation.
- **Evaluating Models and Generating Reports**
  The evaluate_model() method evaluates individual models using performance metrics such as accuracy, precision, recall, and F1-score. The evaluate_all_models() method generates comparative reports, helping to identify the most effective model.

6. Performance Evaluation and Results

6.1. Metrics Used

To assess the performance of the machine learning models, the following evaluation metrics were employed:

1. **Training Accuracy**
   - Measures the percentage of correctly predicted instances on the training dataset.
   - High training accuracy indicates that the model has learned the training data well, but excessively high values may indicate overfitting.
2. **Validation Accuracy**
   - Measures the percentage of correctly predicted instances on the unseen test dataset.
   - Validation accuracy provides insight into how well the model generalizes to new, unseen data.
3. **Accuracy Difference**
   - Calculated as the difference between training and validation accuracy.
   - This metric helps identify overfitting. A large accuracy difference indicates that the model performs well on training data but poorly on unseen data.
4. **Precision**
   - Precision is defined as the ratio of true positive predictions to the total number of positive predictions made by the model.
   - It measures the model's ability to avoid false positives, which is important in financial contexts like loan approval, where incorrectly approving a loan can be costly.

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

Equation 1 Precision

5. **Recall**
   - Recall measures the ratio of true positive predictions to the total number of actual positive cases in the dataset.
   - In this project, recall ensures that the model does not overlook eligible loans, minimizing false rejections.

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

Equation 2 Recall

6. **F1 Score**
   - The F1 Score is the harmonic mean of precision and recall, balancing both metrics. It is especially useful when dealing with imbalanced datasets, as it penalizes models that favor either precision or recall disproportionately.

$$\text{F1 Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Equation 3 F1 Score

7. **Confusion Matrix**
   - The confusion matrix is a fundamental tool in evaluating classification models, as it provides a detailed breakdown of the model's performance across different classes by showing counts of true positives, true negatives, false positives, and false negatives. This helps to quantify the accuracy and error rates, especially in binary and multi-class classification settings, making it essential for performance assessment in machine learning applications (Amin, 2022; Zeng, 2020).

- Confusion Matrix Layout:

		Predicted	
		0	1
Actual	0	TN	FP
	1	FN	TP

Table 4 Confusion Matrix Layout

- True Positives (TP): Correctly predicted approved loans.
- True Negatives (TN): Correctly predicted rejected loans.
- False Positives (FP): Incorrectly predicted approved loans.
- False Negatives (FN): Incorrectly predicted rejected loans.

6.2. Model Insights from Results

6.2.1 K-Nearest Neighbors (KNN)

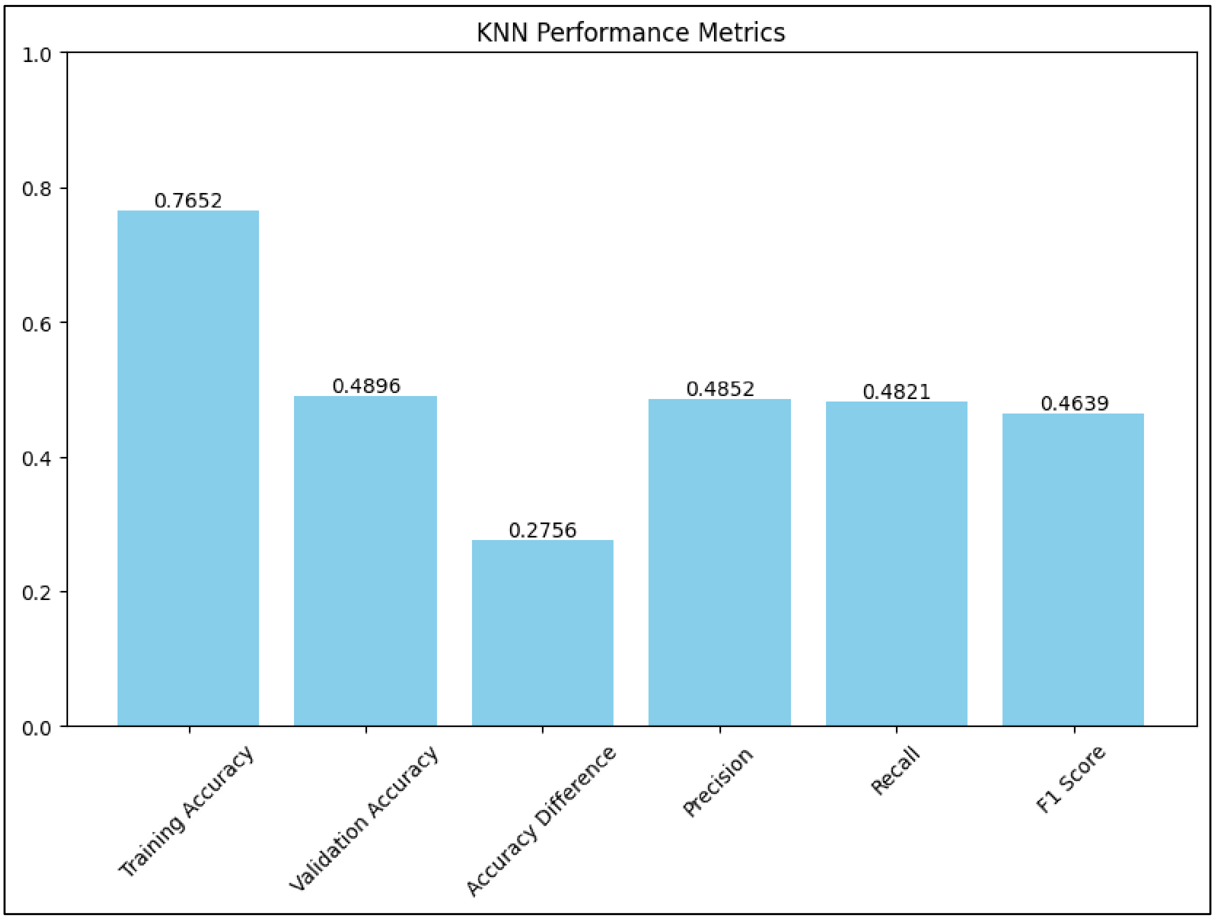
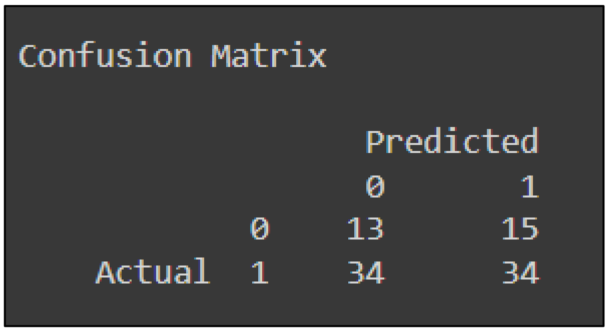


Figure 9 KNN Performance Metrics



```
Confusion Matrix

                    Predicted
                    0        1
              0    13       15
    Actual    1    34       34
```

Figure 10 KNN Confusion Matrix

The KNN model shows a significant accuracy gap of 0.2756 between training (0.7652) and validation accuracy (0.4896), indicating overfitting. Its F1 score of 0.4639 reflects poor generalization. The confusion matrix reveals that 34 false negatives and 15 false positives were recorded, suggesting the model struggles with both types of classification errors. Due to these issues, KNN may not be the optimal model for imbalanced data like this.

6.2.2 Logistic Regression

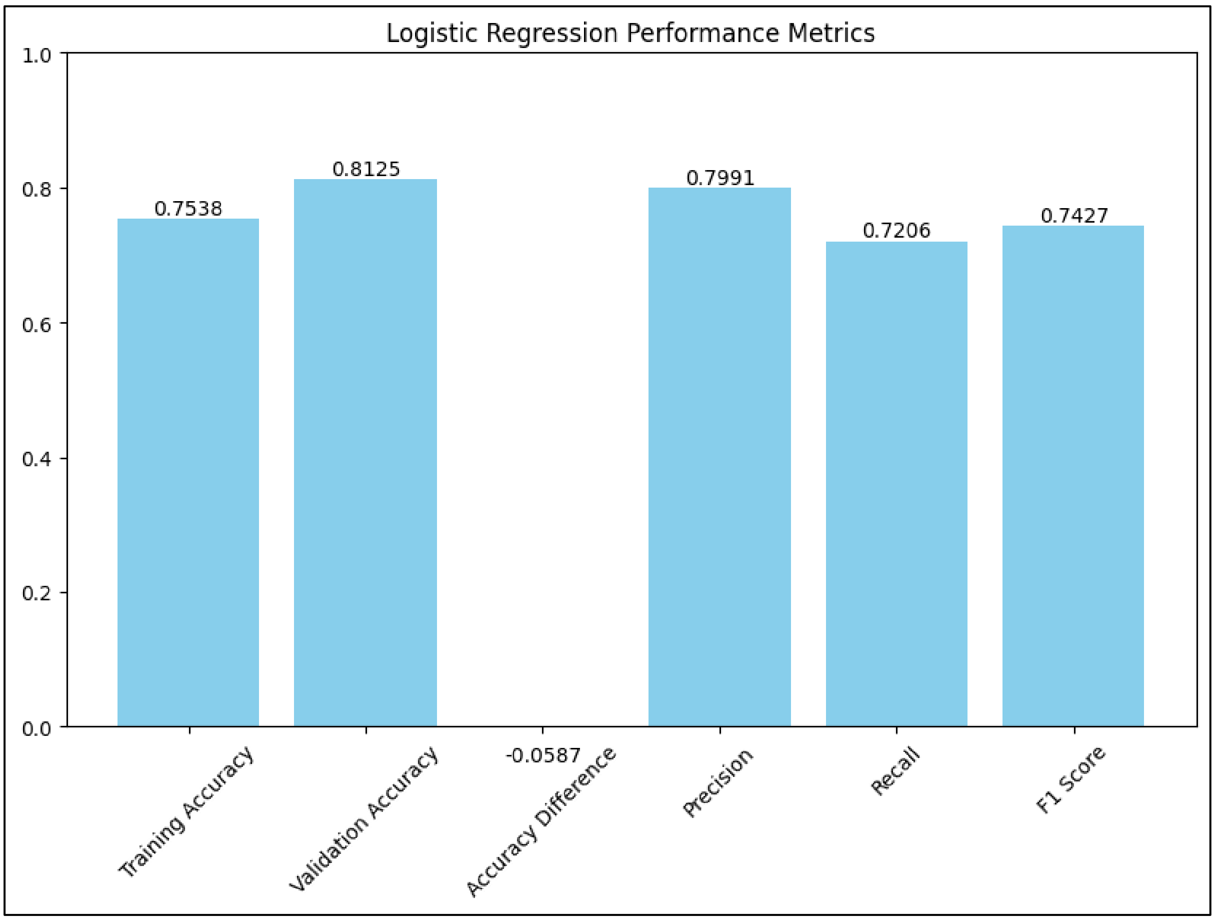
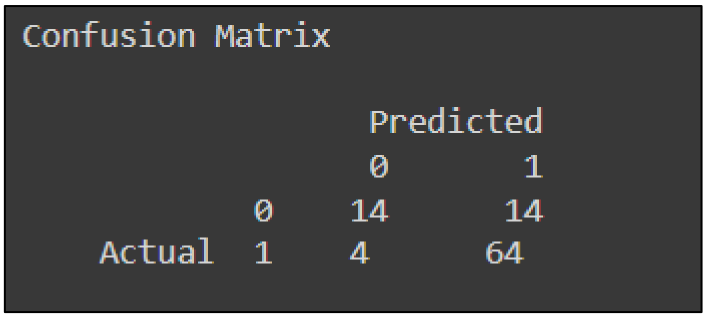


Figure 11 LR Performance Metrics



```
Confusion Matrix

                 Predicted
                 0       1
           0    14      14
    Actual 1     4      64
```

Figure 12 LR Confusion Matrix

Logistic Regression performs consistently, with 0.7538 training accuracy and 0.8125 validation accuracy, indicating minimal overfitting. Its precision (0.7991) and recall (0.7206) suggest a balanced trade-off between false positives and false negatives. The confusion matrix shows 4 false negatives, indicating good sensitivity. This model's simplicity and efficiency make it a strong candidate for deployment.

6.2.3 Random Forest

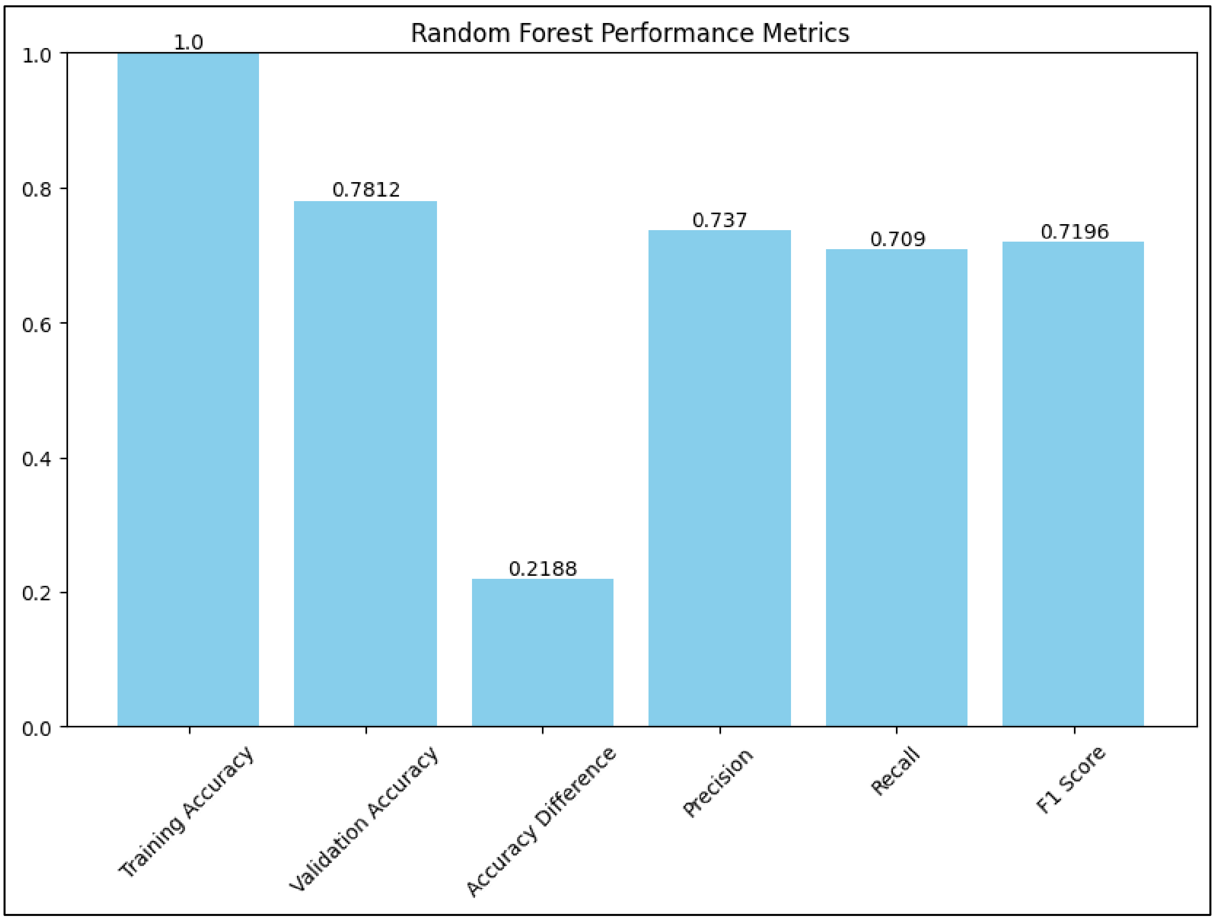
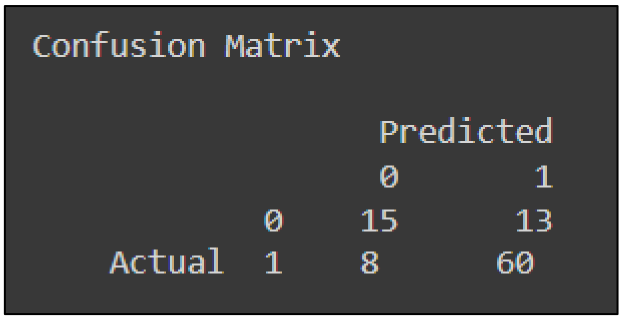


Figure 13 RF Performance Metrics



```
Confusion Matrix

                    Predicted
                    0        1
              0     15       13
      Actual  1     8        60
```

Figure 14 RF Confusion Matrix

Random Forest achieves **perfect training accuracy (1.0)** but has a validation accuracy of **0.7812**, indicating **overfitting**. With a **precision of 0.737** and recall of 0.709, it demonstrates decent predictive ability. The confusion matrix shows **8 false negatives and 13 false positives**, revealing some struggles with both false predictions. While it performs well, careful tuning is required to reduce overfitting.

6.2.4 Decision Tree

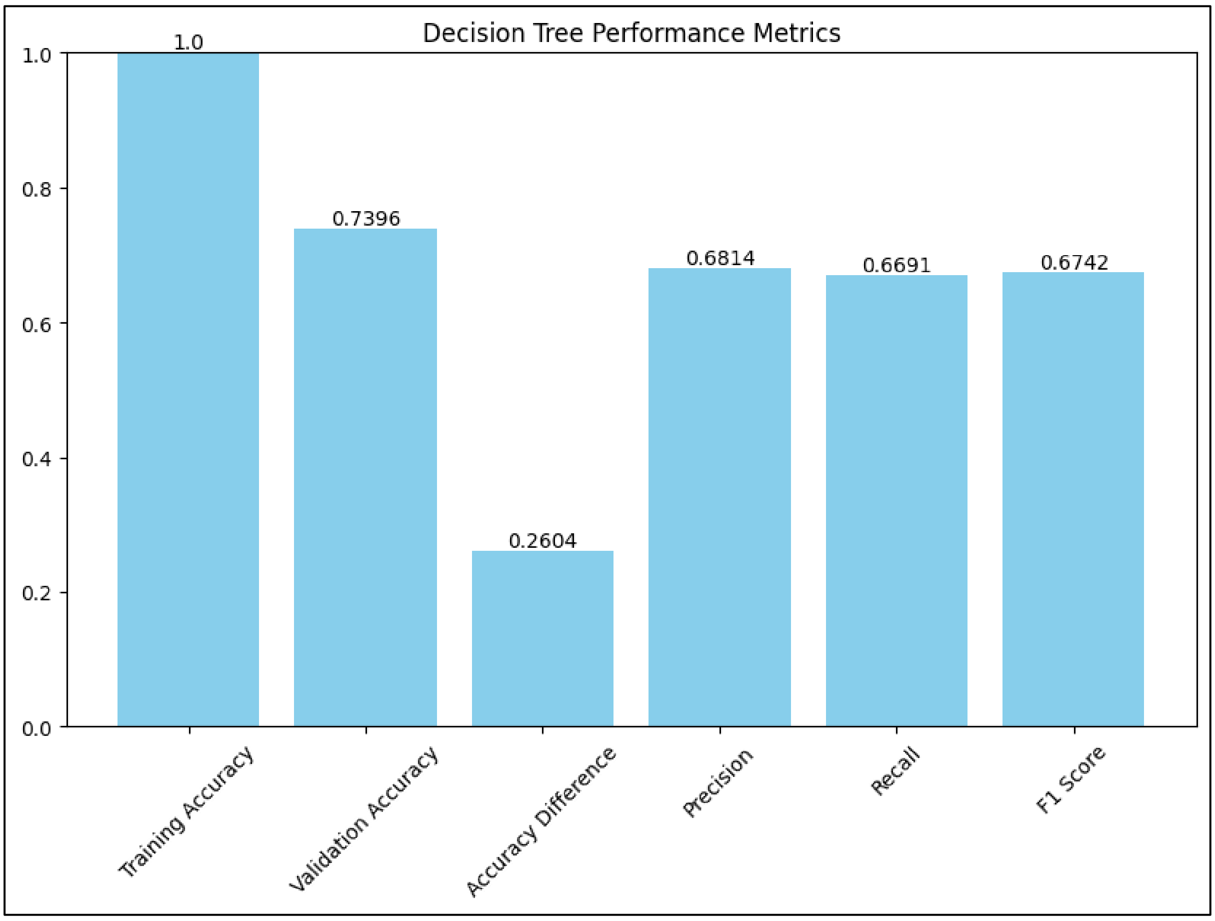
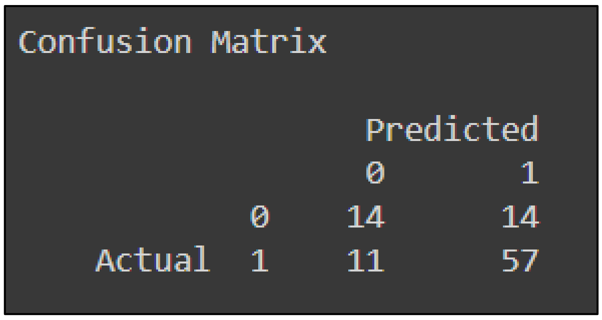


Figure 15 DT Performance Metrics



```
Confusion Matrix

                    Predicted
                    0       1
            0      14      14
    Actual  1      11      57
```

Figure 16 DT Confusion Matrix

The Decision Tree model exhibits extreme **overfitting** with **1.0 training accuracy** and **0.7396 validation accuracy**. Its **F1 score (0.6742)** suggests moderate performance, with the confusion matrix revealing **11 false negatives**. While it provides good interpretability, its overfitting makes it less reliable without further tuning.

6.2.5 Support Vector Machine (SVM)

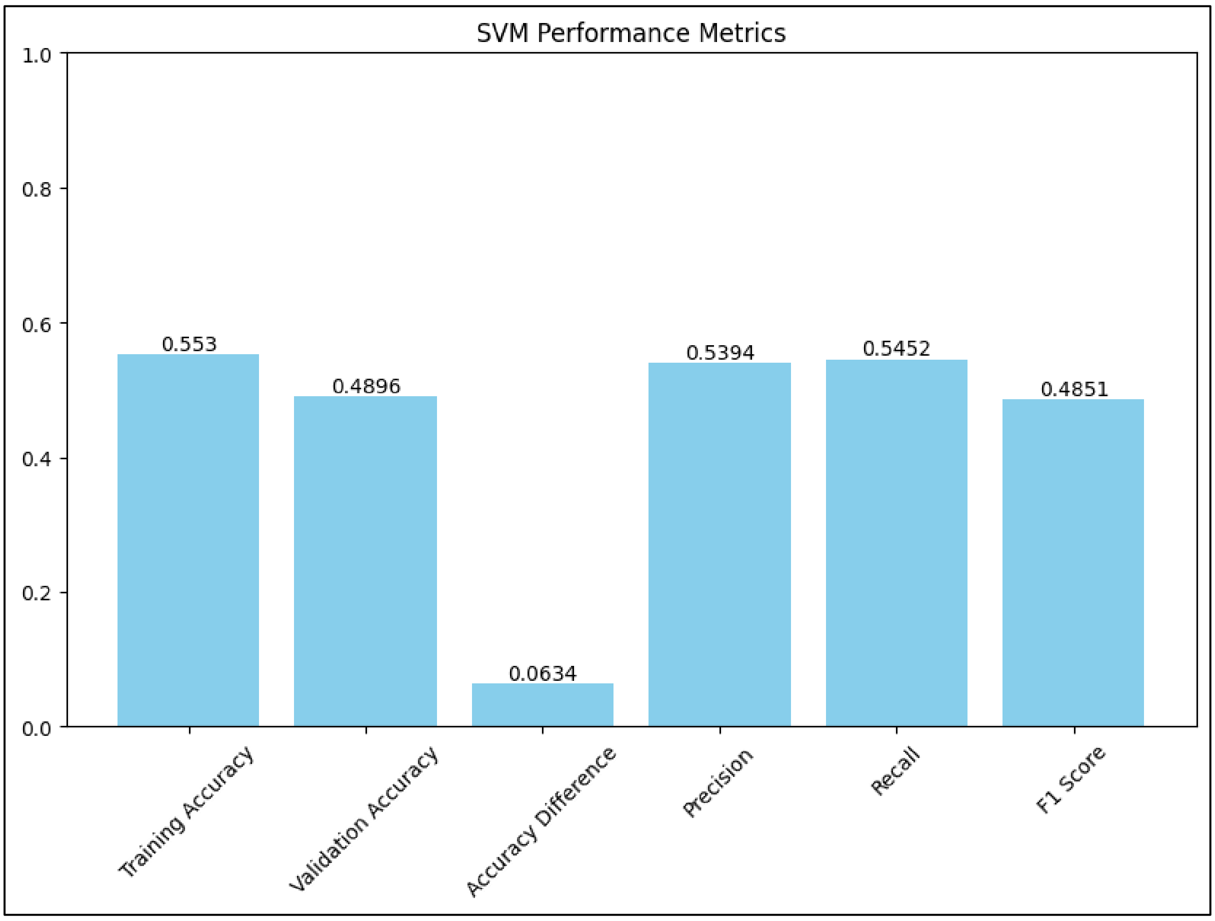
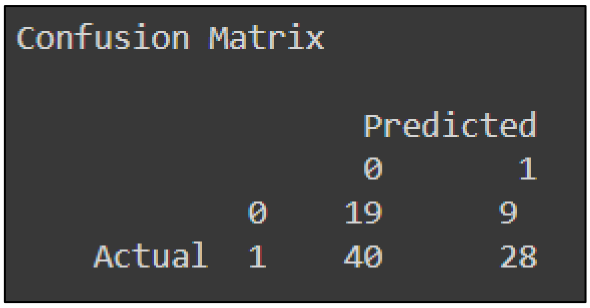


Figure 17 SVM Performance Metrics



```
Confusion Matrix

                    Predicted
                    0       1
           0       19       9
    Actual 1       40      28
```

Figure 18 SVM Confusion Matrix

SVM performs poorly, with both **training (0.553)** and **validation accuracy (0.4896)**, indicating **underfitting**. Its **F1 score of 0.4851** confirms the model's difficulty in learning the data patterns. The confusion matrix highlights **40 false negatives**, showing it struggles to identify positive cases. SVM is not well-suited for this imbalanced dataset.

6.2.6 XGBoost

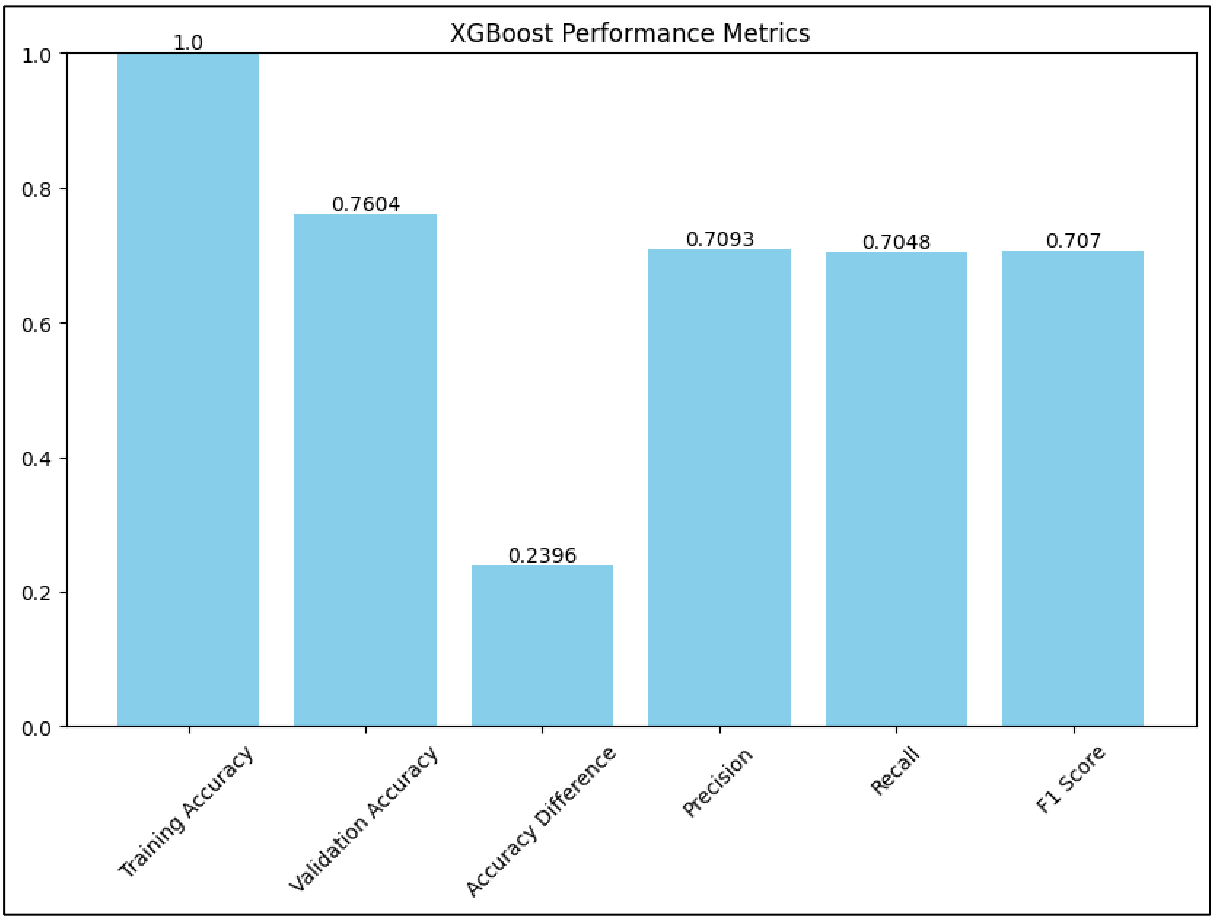
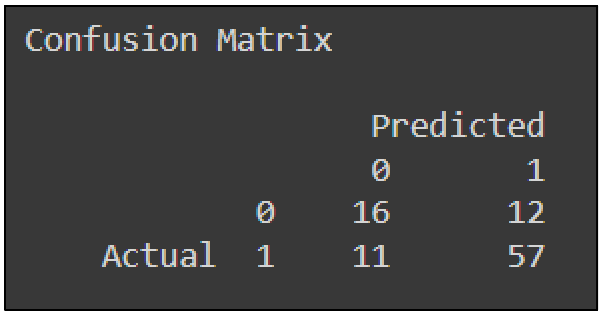


Figure 19 XGB Performance Metrics



```
Confusion Matrix

                 Predicted
                 0       1
          0     16      12
   Actual 1     11      57
```

Figure 20 XGB Confusion Matrix

XGBoost demonstrates strong performance, with **1.0 training accuracy** but **0.7604 validation accuracy**, indicating **some overfitting**. It maintains a high **precision (0.7093)**, and the confusion matrix reveals **11 false negatives**. The model performs well in complex, non-linear scenarios, making it a viable candidate with potential for tuning.

6.2.7 LightGBM

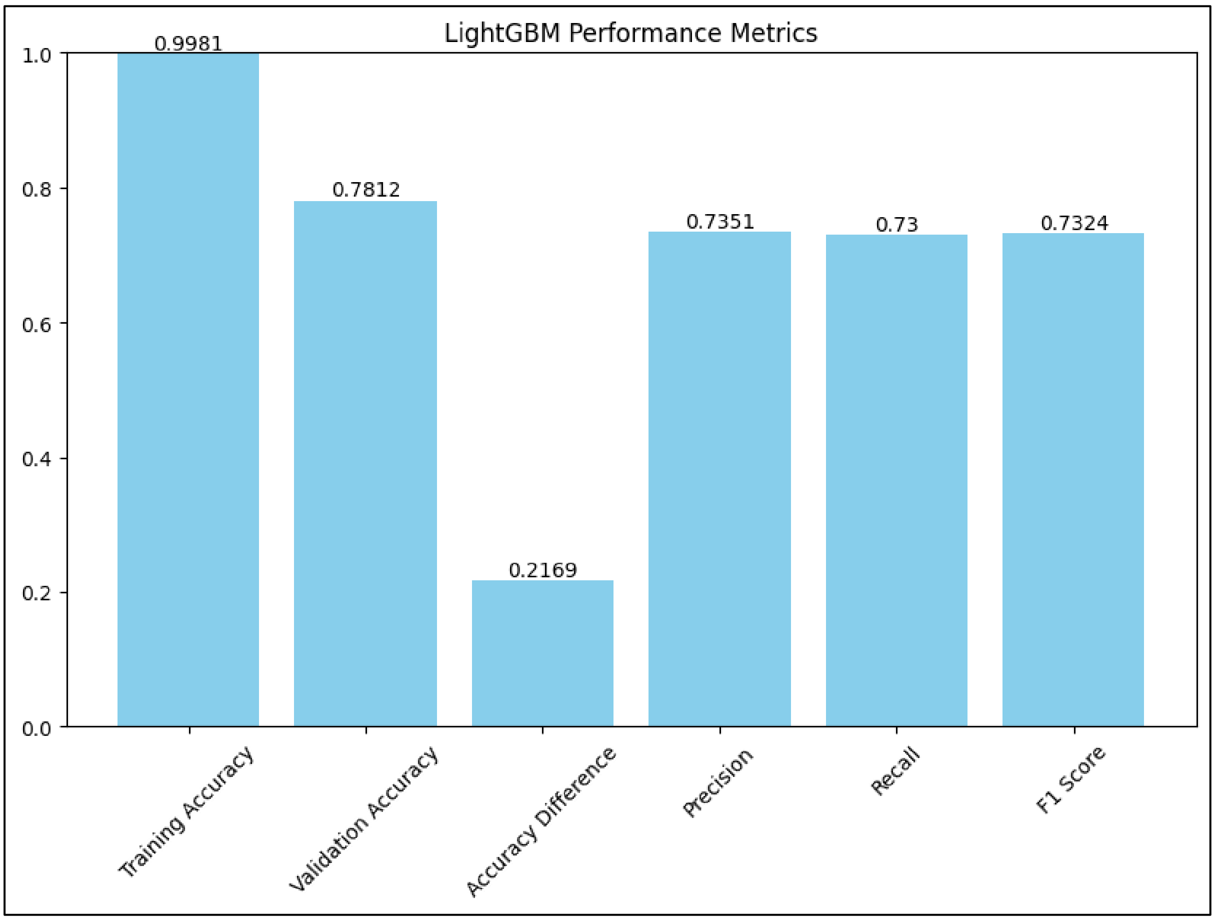
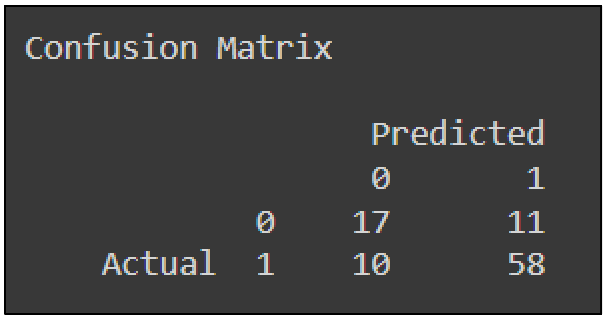


Figure 21 LGBM Performance Metrics



```
Confusion Matrix

                    Predicted
                    0        1
             0     17       11
     Actual  1     10       58
```

Figure 22 LGBM Confusion Matrix

LightGBM provides a good balance between training (**0.9981**) and validation accuracy (**0.7812**). Its **recall (0.73)** suggests it handles false negatives well, with **10 false negatives and 11 false positives** in the confusion matrix. This model is well-suited for handling imbalanced data and offers high efficiency, making it a strong contender for real-world use.

6.2.8 CatBoost

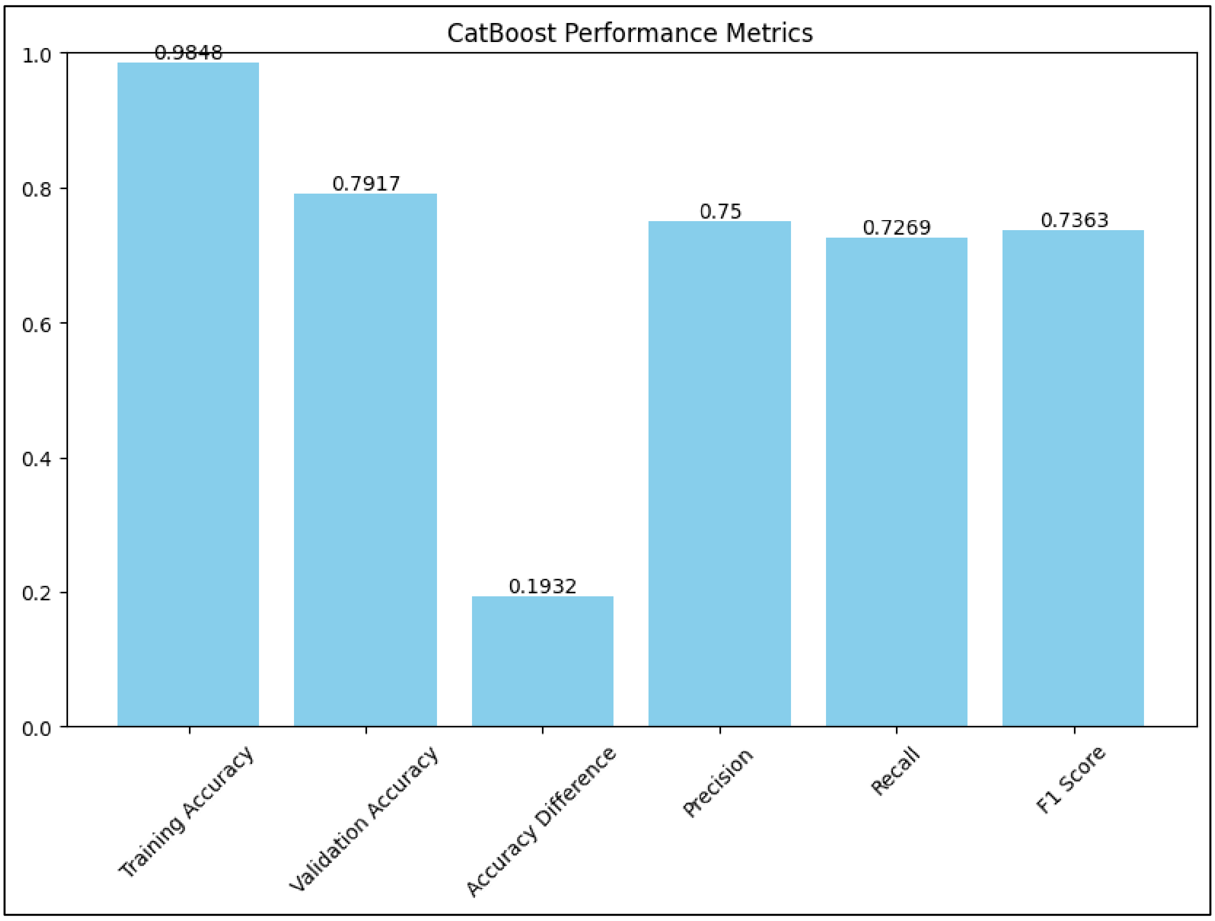
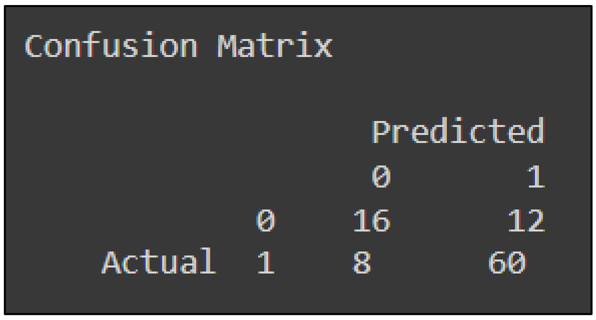


Figure 23 CB Performance Metrics



```
Confusion Matrix

                    Predicted
                     0       1
            0       16      12
    Actual  1        8      60
```

Figure 24 CB Confusion Matrix

CatBoost performs well, achieving 0.9848 training accuracy and 0.7917 validation accuracy, with minimal overfitting. Its precision (0.75) and recall (0.7269) are balanced, with only 8 false negatives recorded in the confusion matrix. This model works effectively with categorical data, making it suitable for deployment without much additional tuning

6.2.9 Naive Bayes

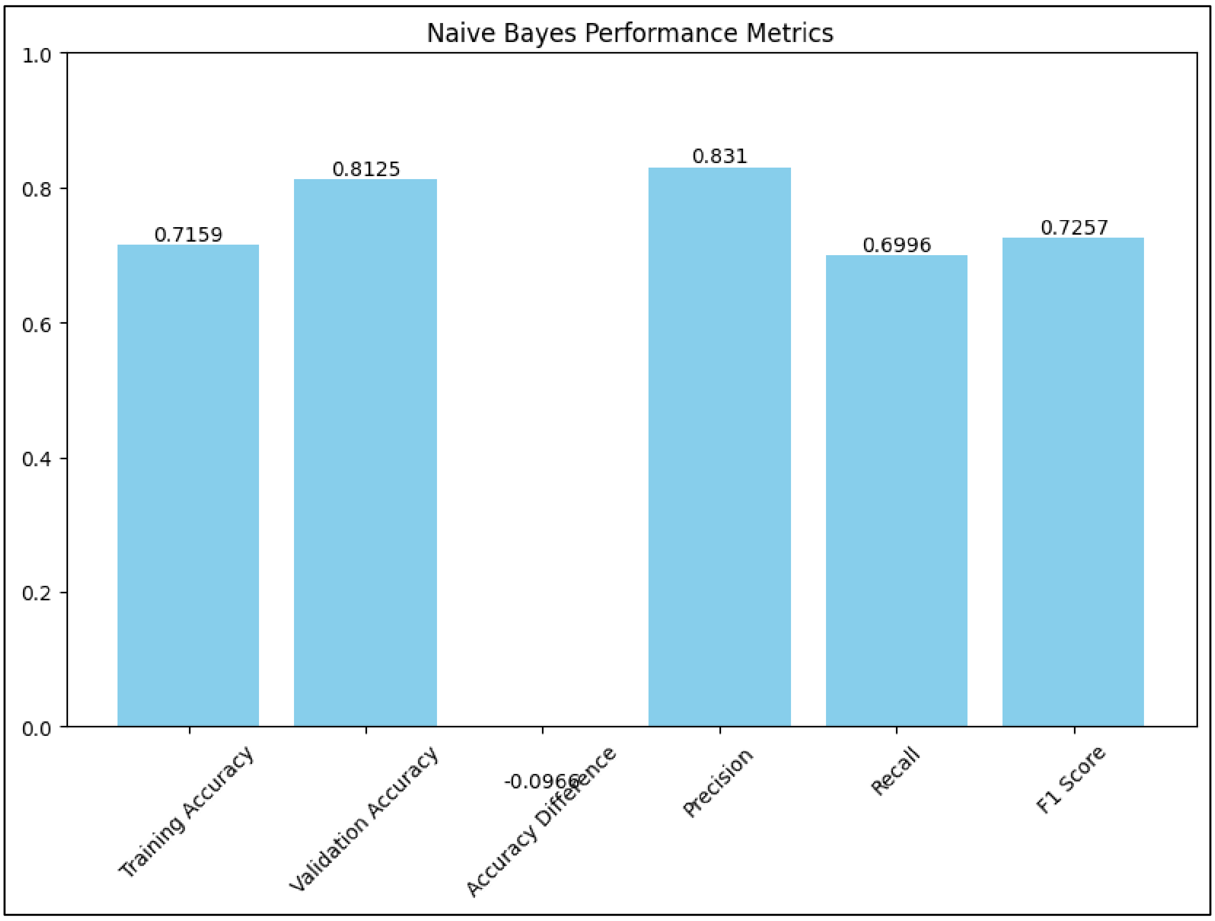
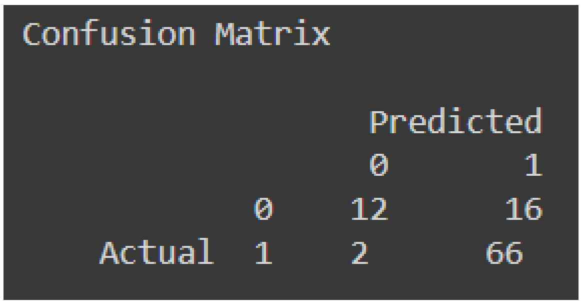


Figure 25 NB Performance Metrics



```
Confusion Matrix

                  Predicted
                  0       1
           0     12      16
   Actual  1     2      66
```

Figure 26 NB Confusion Matrix

Naive Bayes achieves better validation (**0.8125**) than training accuracy (**0.7159**), indicating it generalizes well with a **negative accuracy difference of -0.0966**. Its **F1 score of 0.7257** reflects balanced performance, though **16 false positives** and **2 false negatives** are observed. This model's simplicity and generalization make it a reliable baseline despite some misclassifications.

6.2.10 Gradient Boosting

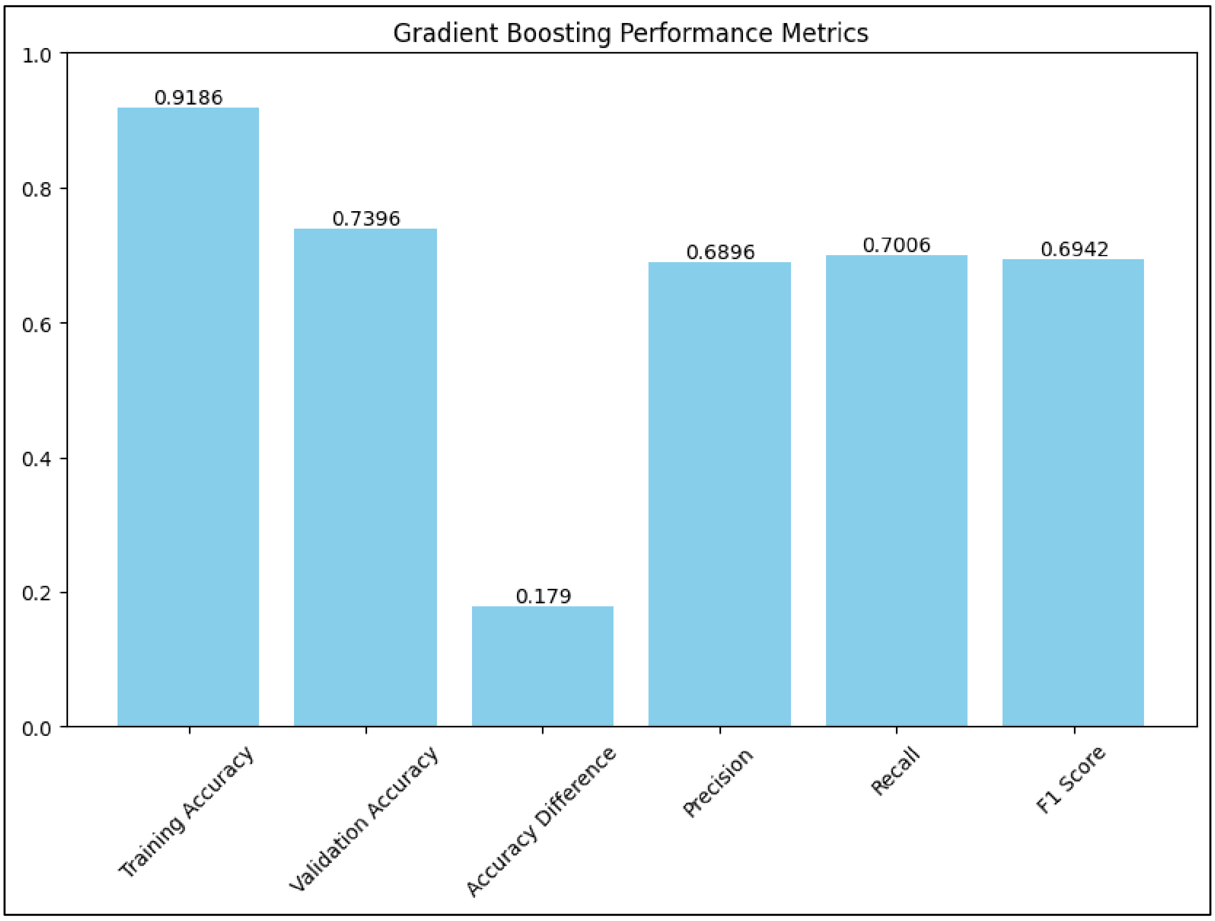
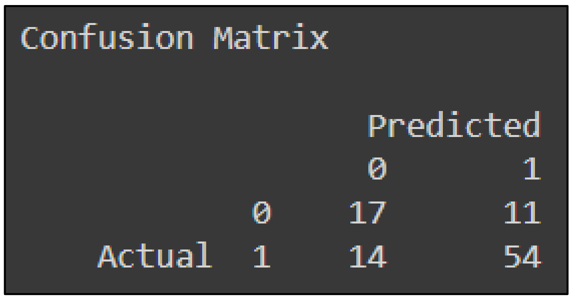


Figure 27 GB Performance Metrics



```
Confusion Matrix

                    Predicted
                    0       1
            0      17      11
    Actual  1      14      54
```

Figure 28 GB Confusion Matrix

Gradient Boosting offers robust performance, with **0.9186 training accuracy** and **0.7396 validation accuracy**, reflecting **mild overfitting**. Its **F1 score of 0.6942** and **14 false negatives** suggest it is moderately sensitive but could be improved with tuning. This model offers a good trade-off between complexity and performance, making it suitable for deployment with further refinement

6.3. Visualizing Performance: Bar Charts for Metrics Comparison

6.3.1 Training Accuracy

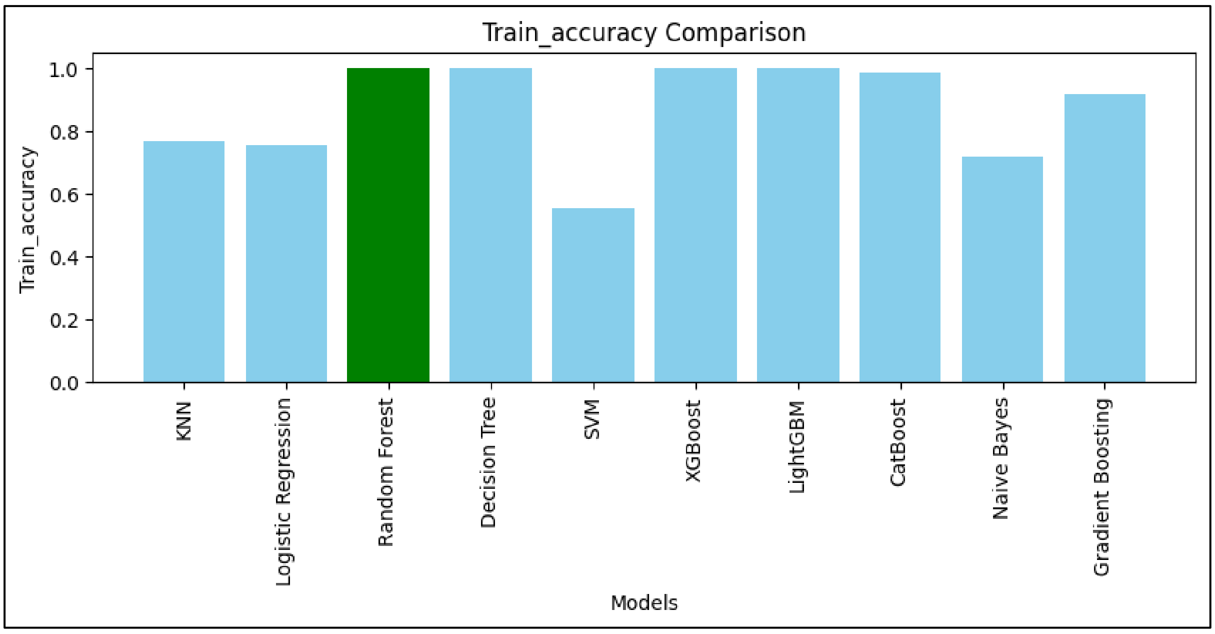


Figure 29 Bar Plot Train Accuracy Comparison

This chart compares the **training accuracy** of all models, highlighting how well each model fits the training data. Higher training accuracy generally reflects a good fit, but excessively high scores (like 1.0) may indicate overfitting, as seen with models like **Random Forest** and **Decision Tree**.

6.3.2 Validation Accuracy

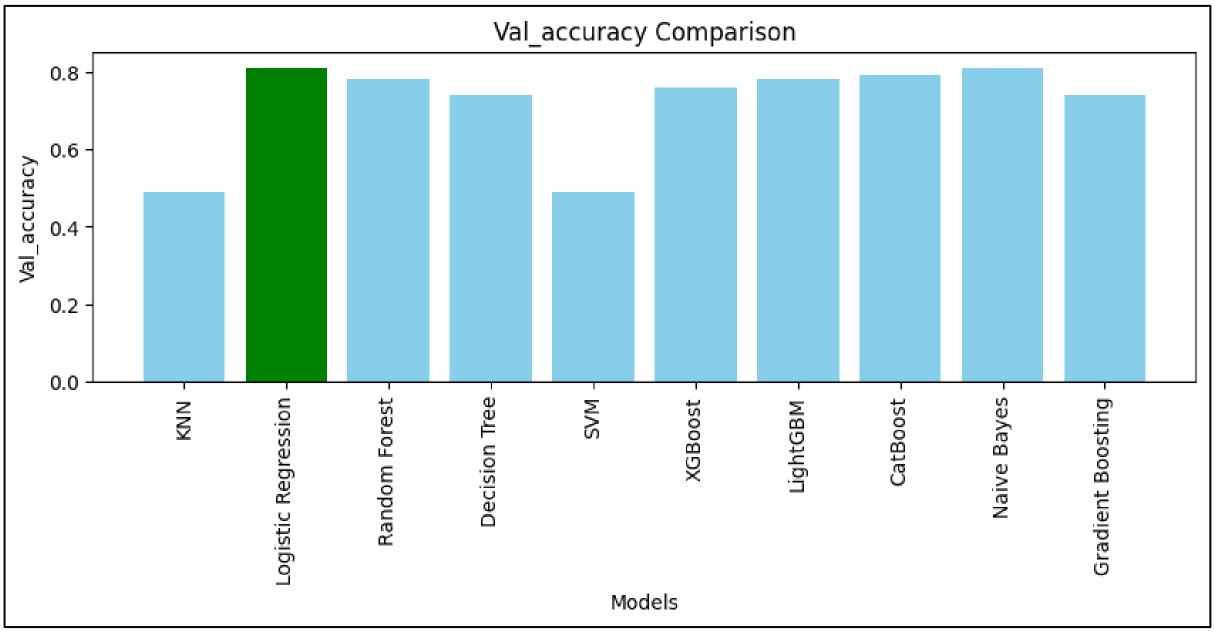


Figure 30 Bar Plot Train Validation Comparison

The validation accuracy chart shows the models' performance on unseen data. It provides insight into how well each model generalizes. Models like **Logistic Regression, CatBoost, and Naive Bayes** demonstrate strong generalization, with validation accuracy close to 0.8. Lower values (e.g., KNN and SVM) suggest potential underfitting.

6.3.3 Accuracy Difference

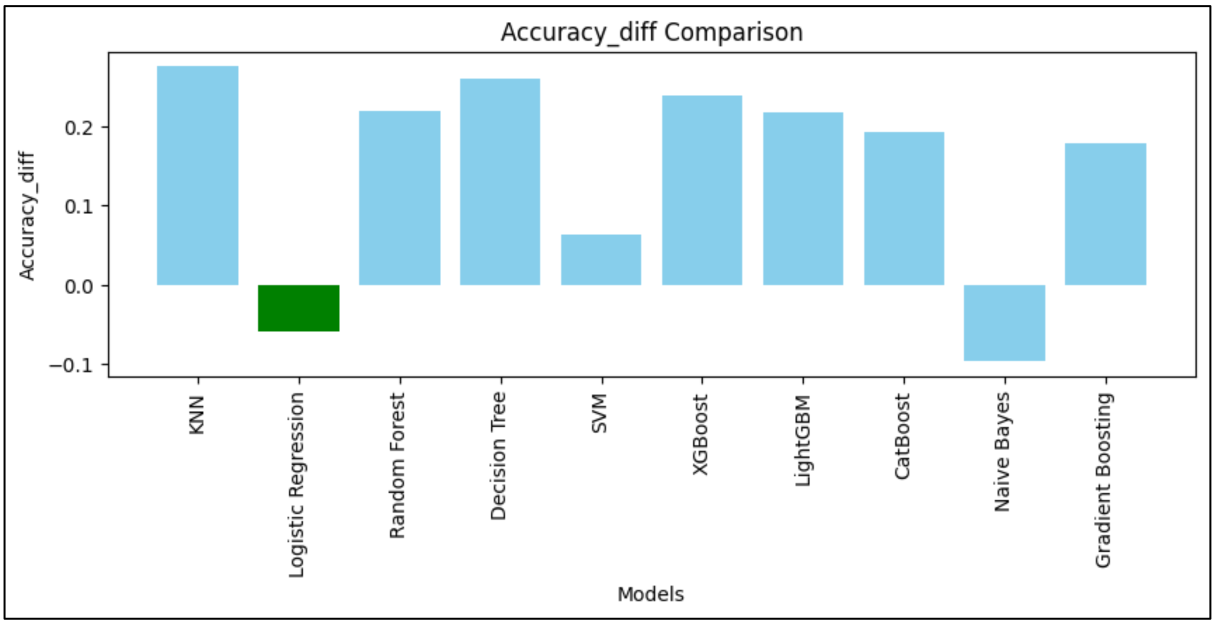


Figure 31 Bar Plot Accuracy Difference Comparison

The accuracy difference measures the **gap between training and validation accuracy**, indicating overfitting if the gap is large. For example, Random Forest exhibits overfitting with a gap of 0.2188, while Logistic Regression shows good consistency with a small negative gap.

6.3.4 Precision

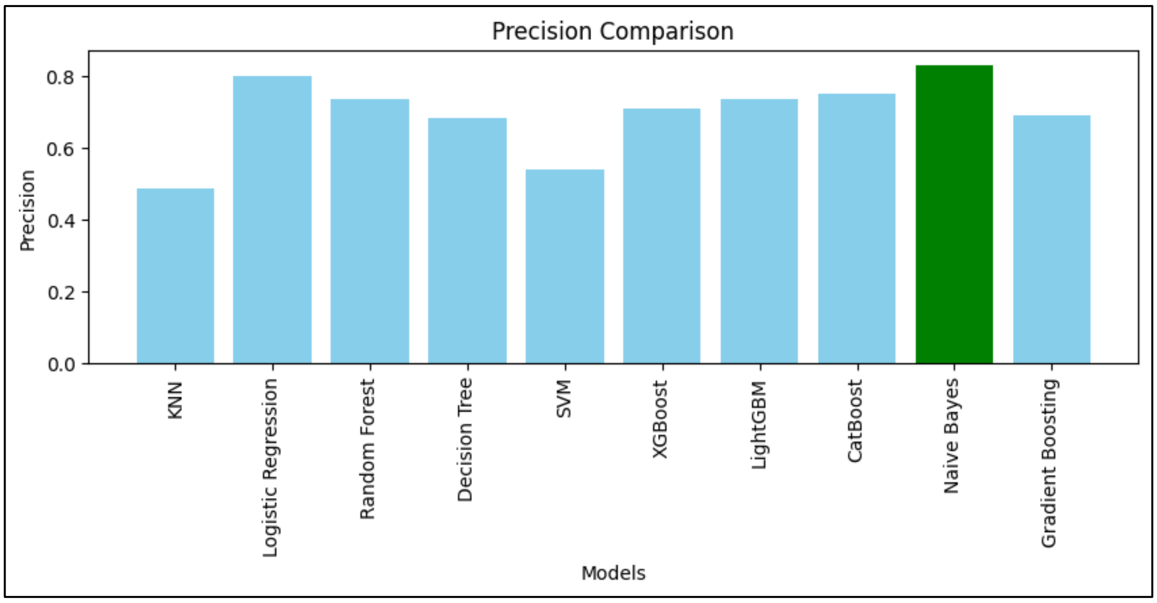


Figure 32 Bar Plot Precision Comparison

This metric highlights the precision of each model and how well it avoids false positives. **Naive Bayes** stands out with the highest precision, which suggests it performs well when it predicts a positive class. KNN and SVM show relatively lower precision, indicating a higher tendency toward false positives.

6.3.5 Recall

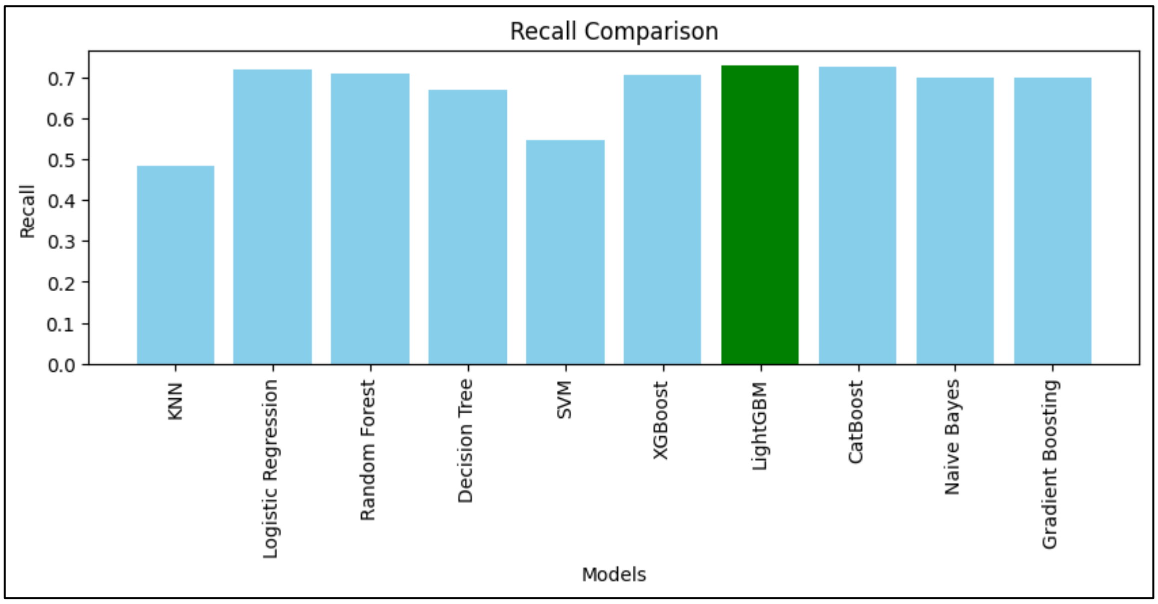


Figure 33 Bar Plot Recall Comparison

The recall comparison indicates each model's ability to correctly identify positive cases (approved loans). **LightGBM and CatBoost** perform exceptionally well in terms of recall, identifying more true positives, making them preferable in scenarios where missing positive predictions is costly.

6.3.6 F1 Score

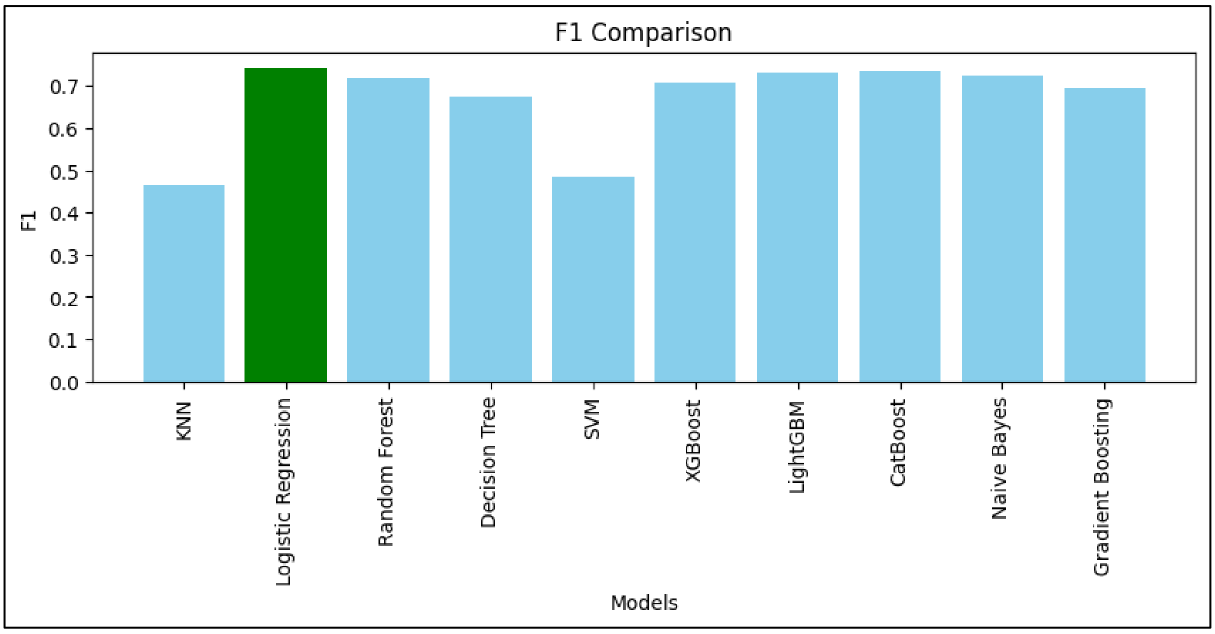


Figure 34 Bar Plot F! Score Comparison

The F1 score balances precision and recall, providing an overall measure of each model's performance. **Logistic Regression** shows a high F1 score, demonstrating strong performance across both precision and recall. KNN exhibits the lowest F1 score, indicating suboptimal trade-offs between precision and recall.

6.4. Model Selection Justification: Why Logistic Regression

After comparing all models based on multiple metrics including training accuracy, validation accuracy, accuracy difference, precision, recall, and F1 score **Logistic Regression** was selected as the optimal model for the loan approval prediction system.

1. **Balanced Performance Across Key Metrics**
   Logistic Regression achieved a high validation accuracy (81.25%) with a minimal accuracy difference (-0.0587) between training and validation phases. This indicates that the model generalizes well to unseen data, ensuring reliable predictions without overfitting.
2. **F1 Score as a Key Evaluation Metric**
   With an F1 score of 0.7427, Logistic Regression provides a good balance between precision (79.91%) and recall (72.06%).

   - Precision helps avoid financial losses by minimizing the number of unqualified loans that are incorrectly approved (false positives).
   - Recall ensures that the model minimizes missed business opportunities by reducing the number of valid loan applications that are incorrectly rejected (false negatives).

   This balance is essential in financial applications, where both types of errors carry significant risks.

3. **Reduced Risk of Overfitting**

   Compared to more complex models such as Random Forest and XGBoost, which exhibited high training accuracy but lower validation performance, Logistic Regression offers consistent performance across datasets. Its low gap between training and validation accuracy highlights its ability to generalize well, ensuring it performs reliably in real-world applications.

4. **Simplicity and Transparency for Business Use**

   Logistic Regression is a simple and interpretable model, making it ideal for financial decision-making, where transparency and accountability are essential. The model's coefficients provide clear insights into how individual features (like credit history and income) influence loan approval, ensuring decisions can be easily explained to stakeholders and regulators.

5. **Handling Imbalanced Data with Threshold Adjustment**

   Given the slight imbalance in the dataset, the probabilistic outputs from Logistic Regression offer flexibility. The model allows for the adjustment of thresholds to either increase approvals or tighten lending policies, helping align predictions with business needs and risk appetite.

6. **Efficiency for Real-Time Prediction**

   Compared to computationally expensive models like ensemble-based Random Forest or boosting algorithms, Logistic Regression is lightweight and efficient. This makes it an excellent choice for real-time loan approvals, ensuring customers receive rapid responses, enhancing the overall experience.

7. User Interaction and Program Usage

7.1. UserInteractionManager Class – Interactive User Input and Prediction Process

The UserInteractionManager class serves as the interface between the user and the machine learning models. Its primary function is to collect loan applicant data through a user-friendly interface, process this input into a format compatible with the trained model, and display predictions in real-time.

This class provides essential methods that:

1. **Collect User Input**: It prompts users to input relevant loan application features such as income, loan amount, credit history, and marital status.
2. **Data Validation**: Basic validation is performed to ensure that all required fields are filled correctly to prevent prediction errors.

3. **Model Invocation and Prediction**: Once the input is processed, the class passes the data to the selected trained model for prediction.
4. **Display Results**: The predicted loan status (approved or not approved) is displayed to the user in real-time, enhancing the decision-making process.

This interactive component ensures that users without technical knowledge can effectively engage with the prediction system, making it suitable for deployment in real-world financial applications.

7.2. StepbyStep User Guide

This section provides users with detailed instructions on how to set up, interact with, and obtain predictions from the loan approval prediction system. The guide ensures a smooth experience by outlining the necessary steps for running the program, entering data, and interpreting the predictions generated by the models.

7.2.1. Setting Up and Running the Program

To start the program and run all the cells:

1. Open the project link on Google Colab: Loan Approval Prediction Colab
2. Once the notebook loads, navigate to the Runtime menu on the top navigation bar.
3. Select Run all to execute all the code cells automatically.
4. Wait for the cells to complete their execution. You will see the prompts for user input once the UserInteractionManager class initializes.

7.2.2. Collecting User Input for Predictions

After running the program, the system will prompt you to enter the following loan-related information:

- Enter Gender (Male, Female or 'q' to quit):
- Enter Married (Yes, No or 'q' to quit):
- Enter Dependents (0, 1, 2, 3+ or 'q' to quit):
- Enter Education (Graduate, Not Graduate or 'q' to quit):
- Enter Self_Employed (Yes, No or 'q' to quit):
- Enter ApplicantIncome (numeric value or 'q' to quit):
- Enter CoapplicantIncome (numeric value or 'q' to quit):
- Enter LoanAmount (numeric value or 'q' to quit):
- Enter Loan_Amount_Term (numeric value or 'q' to quit):
- Enter Credit_History (Yes, No or 'q' to quit):
- Enter Property_Area (Semiurban, Urban, Rural or 'q' to quit):

7.2.3. Example Prediction Using Logistic Regression

Here is a sample input session and the predictions from various models:

- KNN: Loan Status Prediction - Approved
- Logistic Regression: Loan Status Prediction - Approved
- Random Forest: Loan Status Prediction - Approved
- Decision Tree: Loan Status Prediction - Approved
- SVM: Loan Status Prediction - Approved
- XGBoost: Loan Status Prediction - Approved
- LightGBM: Loan Status Prediction - Approved
- CatBoost: Loan Status Prediction - Approved
- Naive Bayes: Loan Status Prediction - Approved
- Gradient Boosting: Loan Status Prediction - Approved

7.2.4. Video Tutorial Link: Demonstrating Program Setup and Usage

A detailed video tutorial is available to demonstrate how to set up and use the loan approval prediction system. The video provides a visual guide to ensure users can easily follow along.

Video Demonstration Link

7.3. How to Interpret Model Outputs

After submitting user input, predictions are generated by multiple models. While the system provides outputs from several algorithms, **Logistic Regression** is the final selected model due to its balanced performance in accuracy, precision, recall, and F1-score.

- **Approved (1)**: The model predicts that the loan application is likely to be approved based on the provided data, indicating that the applicant meets the eligibility criteria.
- **Not Approved (0)**: The model predicts that the loan will not be approved, suggesting that the applicant's profile does not align with the requirements.

Since **Logistic Regression** is our chosen model, users should primarily rely on its prediction for decision-making. Predictions from other models are provided as a reference, offering additional insights for borderline cases. However, **Logistic Regression** offers a good balance of interpretability and performance, making it the most suitable for practical financial applications.

This interpretation process ensures that users can confidently act on predictions, knowing they are aligned with both business objectives and reliable metrics.

8. Conclusion and Future Work

8.1. Summary of Key Insights and Findings

The project successfully developed a machine learning-based system for predicting loan approvals. Through rigorous **data preprocessing**, **exploratory data analysis (EDA)**, and **model evaluation**, the system provided valuable insights into loan eligibility based on applicant profiles. Key findings include:

- **Logistic Regression** was identified as the best model, balancing accuracy, interpretability, and performance.
- The **F1 score** proved to be a crucial metric for evaluating the model, given the imbalanced dataset.
- **Data preprocessing** steps, including handling missing values, encoding categorical features, and addressing outliers, significantly improved the model's performance.
- The project demonstrated how machine learning can streamline the loan approval process, making decisions more efficient and data-driven.

8.2. Limitations of the Current Implementation

Despite the project's success, several limitations were identified:

1. **Dataset Size and Quality**: The dataset used is limited in size and may not cover all possible real-world scenarios, reducing the model's generalization ability.
2. **Limited Feature Set**: Important financial indicators like credit scores, interest rates, and debt-to-income ratios are absent, which could improve prediction accuracy.
3. **Bias in Data**: Any inherent bias in the dataset (e.g., gender or location bias) may impact model fairness.
4. **Overfitting Risk**: Some models, such as Random Forest and XGBoost, showed overfitting with high training accuracy but lower validation accuracy.
5. **Static Model Performance**: The model is trained on a single dataset, which may not reflect future data trends, requiring frequent retraining.

8.3. Future Research Directions and Improvements

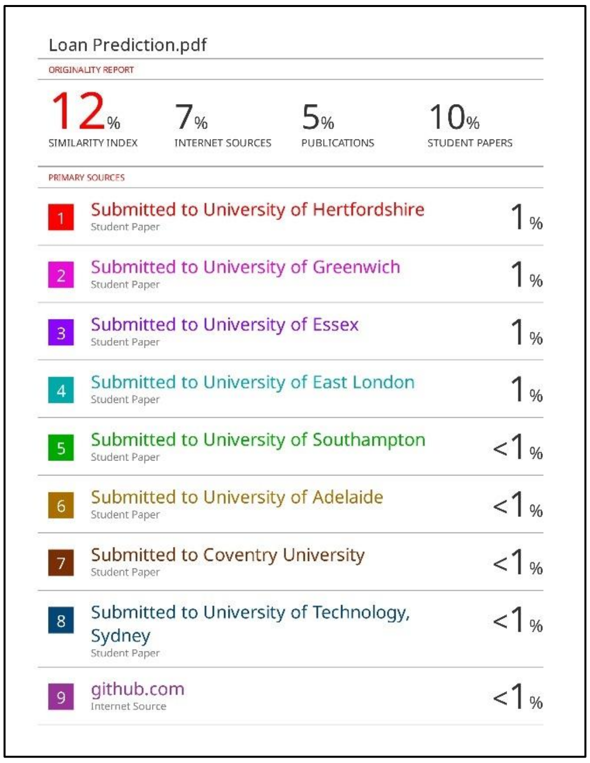
To enhance the system's performance and applicability, the following improvements are suggested:

1. **Expanding the Dataset**: Incorporating a larger and more diverse dataset would help improve model generalization and reduce bias.
2. **Feature Engineering**: Introducing additional financial indicators, such as credit scores, interest rates, and loan repayment histories, could enhance prediction accuracy.
3. **Regular Model Updates**: Implementing a framework for retraining the model periodically would ensure it stays relevant to changing financial environments.

4. **Bias Mitigation Techniques**: Bias mitigation techniques, such as fairness-aware algorithms and regular bias audits, are essential for ensuring equitable outcomes in financial modeling. These approaches not only address explicit biases in decision-making algorithms but also help secure fairer outcomes for underrepresented groups, particularly in high-stakes applications like loan approvals (Maneriker et al., 2023; Pavón Pérez, 2022).
5. **Deployment in Real-world Scenarios**: Developing a robust API or integrating the model into an organization's loan processing system would allow for seamless practical use.

These improvements would further refine the loan prediction model, ensuring higher accuracy, fairness, and scalability for real-world applications.

9. Plagiarism Report

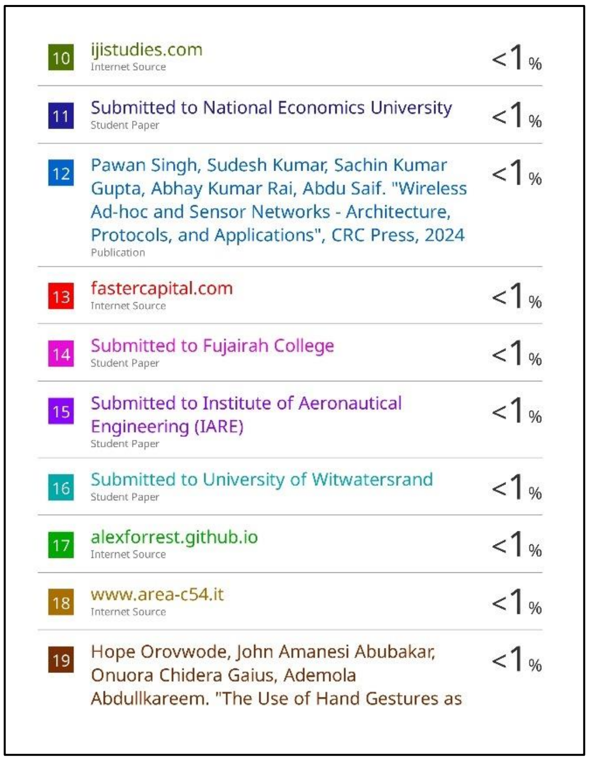

Loan Prediction.pdf

ORIGINALITY REPORT

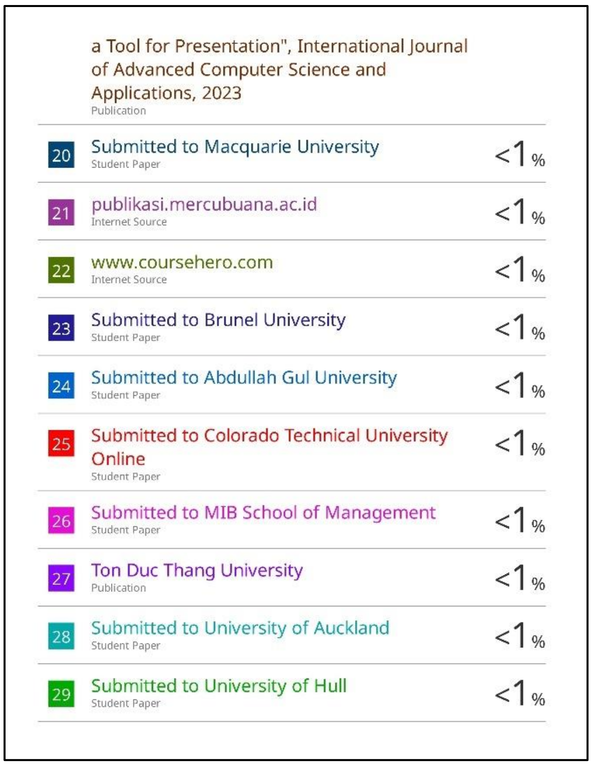
12%	7%	5%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

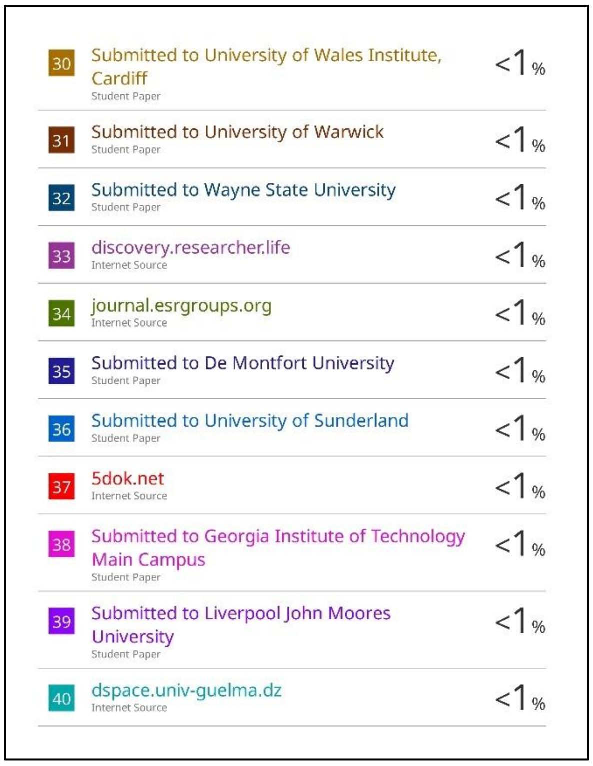
1	Submitted to University of Hertfordshire Student Paper	1%
2	Submitted to University of Greenwich Student Paper	1%
3	Submitted to University of Essex Student Paper	1%
4	Submitted to University of East London Student Paper	1%
5	Submitted to University of Southampton Student Paper	<1%
6	Submitted to University of Adelaide Student Paper	<1%
7	Submitted to Coventry University Student Paper	<1%
8	Submitted to University of Technology, Sydney Student Paper	<1%
9	github.com Internet Source	<1%



10	ijistudies.com Internet Source	<1%
11	Submitted to National Economics University Student Paper	<1%
12	Pawan Singh, Sudesh Kumar, Sachin Kumar Gupta, Abhay Kumar Rai, Abdu Saif. "Wireless Ad-hoc and Sensor Networks - Architecture, Protocols, and Applications", CRC Press, 2024 Publication	<1%
13	fastercapital.com Internet Source	<1%
14	Submitted to Fujairah College Student Paper	<1%
15	Submitted to Institute of Aeronautical Engineering (IARE) Student Paper	<1%
16	Submitted to University of Witwatersrand Student Paper	<1%
17	alexforrest.github.io Internet Source	<1%
18	www.area-c54.it Internet Source	<1%
19	Hope Orovwode, John Amanesi Abubakar, Onuora Chidera Gaius, Ademola Abdullkareem. "The Use of Hand Gestures as	<1%



	a Tool for Presentation", International Journal of Advanced Computer Science and Applications, 2023 Publication	
20	Submitted to Macquarie University Student Paper	<1 %
21	publikasi.mercubuana.ac.id Internet Source	<1 %
22	www.coursehero.com Internet Source	<1 %
23	Submitted to Brunel University Student Paper	<1 %
24	Submitted to Abdullah Gul University Student Paper	<1 %
25	Submitted to Colorado Technical University Online Student Paper	<1 %
26	Submitted to MIB School of Management Student Paper	<1 %
27	Ton Duc Thang University Publication	<1 %
28	Submitted to University of Auckland Student Paper	<1 %
29	Submitted to University of Hull Student Paper	<1 %



30	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %
31	Submitted to University of Warwick Student Paper	<1 %
32	Submitted to Wayne State University Student Paper	<1 %
33	discovery.researcher.life Internet Source	<1 %
34	journal.esrgroups.org Internet Source	<1 %
35	Submitted to De Montfort University Student Paper	<1 %
36	Submitted to University of Sunderland Student Paper	<1 %
37	5dok.net Internet Source	<1 %
38	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1 %
39	Submitted to Liverpool John Moores University Student Paper	<1 %
40	dspace.univ-guelma.dz Internet Source	<1 %

41	www.researchsquare.com Internet Source	<1 %

Exclude quotes	Off	Exclude matches	< 10 words
Exclude bibliography	Off		

10. Appendix

10.1. Google CoLab

https://colab.research.google.com/drive/1LI_N5rOq26tpBijR0bf4Q7Ng2_CXuGmH?usp=sharing

10.2. Video Demonstration

https://drive.google.com/file/d/1_h6R6HmZCK-H_mpZ-dtJlMUAuUHeb2VD/view?usp=sharing

10.3. Kaggle Dataset

https://www.kaggle.com/datasets/krishnaraj30/finance-loan-approval-prediction-data

10.4. Plagiarism

https://drive.google.com/file/d/1WL6pwVvTKinJrQEXXYeGpLqD2CeGOMg-/view?usp=sharing

11. References

- Kokru, J., Ghodke, A.S., Chavan, P., Chand, S., and Mane, S. (2022) Bank Loan Approval Prediction System Using Machine Learning Algorithms. doi:10.48175/ijarsct-2637
- Mukri, M. (2023) Predicting Loan Approval Using Machine Learning. doi:10.56726/irjmets41180
- Kaggle (2023) Finance Loan Approval Prediction Data. Available at: https://www.kaggle.com/datasets/krishnaraj30/finance-loan-approval-prediction-data
- Emmanuel, T., Maupong, T.M., Mpoeleng, D., Semong, T., Banyatsang, M., and Tabona, O. (2021) A survey on missing data in machine learning. Available at: https://dx.doi.org/10.1186/s40537-021-00516-9 (Accessed: 1 November 2024).
- Gond, V.K., Dubey, A., and Rasool, A. (2021) A Survey of Machine Learning-Based Approaches for Missing Value Imputation. Available at: https://dx.doi.org/10.1109/ICIRCA51532.2021.9544957 (Accessed: 1 November 2024).
- Emmanuel, T., Maupong, T.M., Mpoeleng, D., Semong, T., Banyatsang, M., and Tabona, O. (2021) A survey on missing data in machine learning. Available at: https://dx.doi.org/10.1186/s40537-021-00516-9
- Gond, V.K., Dubey, A., and Rasool, A. (2021) A Survey of Machine Learning-Based Approaches for Missing Value Imputation. Available at: https://dx.doi.org/10.1109/ICIRCA51532.2021.9544957
- Amin, M.F. (2022) Confusion Matrix in Binary Classification Problems: A Step-by-Step Tutorial. Available at: https://dx.doi.org/10.21608/erjeng.2022.274526
- Zeng, G. (2020) On the confusion matrix in credit scoring and its analytical properties. Available at: https://dx.doi.org/10.1080/03610926.2019.1568485
- Maneriker, P., Burley, C., and Parthasarathy, S. (2023) Online Fairness Auditing through Iterative Refinement. Available at: https://dx.doi.org/10.1145/3580305.3599454
- Pavón Pérez, Á. (2022) Bias in Artificial Intelligence Models in Financial Services. Available at: https://dx.doi.org/10.1145/3514094.3539561