A MINI PROJECT REPORT

On

BANK LOAN ELIGIBILITY AND PREDICTION

Submitted by,

Mr. A. VARUN	22J41A67D3
Mr. V. VENKATA SARMA	22J41A67K1
Mr. B. SAGAR	22J41A67J2
Mr. V. KALYAN KUMAR	22J41A67J9

in partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING -DATA SCIENCE

Under the Guidance of

Ch. Lavanya

Assistant Professor, CSE-DS

COMPUTER SCIENCE ENGINEERING - DATA SCIENCE

MALLA REDDY ENGINEERING COLLEGE

(An UGC Autonomous Institution, Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad) Maisammaguda, Secunderabad, Telangana, India 500100

FEBRUARY– 2025

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

BONAFIDE CERTIFICATE

This is to certify that this mini project work entitled "**BANK LOAN ELIGIBILITY AND PREDICTION**",submitted by **V.VENKATA SARMA (22J41A67K1), A.VARUN (22J41A67D3), V.KALYAN KUMAR (22J41A67J9), B.SAGAR (22J41A67J2)** to Malla Reddy Engineering College affiliated to JNTUH, Hyderabad in partial fulfilment for the award of Bachelor of Technology in Computer Science Engineering - Data Science is a bonafide record of project work carried out under my/our supervision during the academic year 2025–2026 and that this work has not been submitted elsewhere for a degree.

SIGNATURE	**SIGNATURE**
CH. LAVANYA	DR. S. SHIVA PRASAD
SUPERVISOR	**HOD**
Assistant Professor CSE-DS	Professor CSE-DS
Malla Reddy Engineering College	Malla Reddy Engineering College
Secunderabad, 500 100	Secunderabad, 500 100

Submitted for Mini Project viva-voce examination held on _____

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

DECLARATION

I hereby declare that the project titled '**BANK LOAN ELIGIBILITY AND PREDICTION**' submitted to Malla Reddy Engineering College (Autonomous) and affiliated with JNTUH, Hyderabad, in partial fulfilment of the requirements for the award of a **Bachelor of Technology** in **Computer Science Engineering – Data Science**, represents my ideas in my own words. Wherever others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity, and I have not misrepresented, fabricated, or falsified any idea, data, fact, or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Signature(s)

V. VENKATA SARMA	22J41A67K1	_____
B. SAGAR	22J41A67J2	_____
A. VARUN	22J41A67D3	_____
V. KALYAN KUMAR	22J41A67J9	_____

Secunderabad - 500 100

Date:

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

ACKNOWLEDGEMENT

We express our sincere thanks to our Principal, **Dr. A. Ramaswami Reddy**, who took keen interest and encouraged us in every effort during the project work. We express our heartfelt thanks to **Dr. S.Shiva Prasad** Head of Department **CSE-DS**, for his kind attention and valuable guidance throughout the project work. We are thankful to our Project Coordinator **Ch. Lavanya** Assistant Professor, Department of **CSE-DS**, for his cooperation during the project work. We are extremely thankful to our Project Guide , Assistant Professor, for his constant guidance and support to complete the project work. We also thank all the teaching and non-teaching staff of the Department for their cooperation during the project work.

V. VENKATA SARMA 22J41A67K1

V. KALYAN KUMAR 22J41A67J9

A. VARUN 22J41A67D3

B. SAGAR 22J41A67J2

ABSTRACT

The exponential rise of social media platforms has led to an unprecedented increase in user-generated advertisements. These ads span various categories such as commercial promotions, political messaging, public awareness, and educational campaigns. As the volume of such content grows, the need for automated systems to classify and analyze social media advertisements has become critical for digital marketing optimization, regulatory compliance, and user experience enhancement.

This project focuses on developing an intelligent and scalable classification system for social media ads using machine learning and natural language processing techniques. The objective is to categorize advertisements based on their textual content into meaningful classes such as commercial, political, educational, and entertainment. The system begins with the collection and preprocessing of a labeled dataset of ad text, followed by feature extraction using Term Frequency-Inverse Document Frequency (TF-IDF) and word embedding techniques. A range of machine learning models—Logistic Regression, Support Vector Machine (SVM), Random Forest—alongside deep learning architectures like Convolutional Neural Networks (CNN), VGG16, and NASNet, were implemented and evaluated.

The classification models were tested for accuracy, precision, recall, and F1-score, with VGG16 and SVM models achieving the highest performance (above 85% accuracy). The system is deployed via a Django-based web interface, enabling users to input ad content and receive real-time category predictions with associated confidence scores.

Overall, this project demonstrates the feasibility and effectiveness of combining classical and deep learning models for the automated classification of social media ads. The modular design allows for future enhancements including multilingual support, real-time classification, and multimodal analysis incorporating visual features. This work contributes to the growing field of AI-powered content moderation and social media intelligence systems.

Keywords: Social Media, Advertisement Classification, Machine Learning, Natural Language Processing, Deep Learning, Text Analysis, Django

TABLE OF CONTENTS

Chapter	Description	Page No
1	**INTRODUCTION**	**1-3**
	1.1. Introduction	1
	1.2. Problem Definition	1
	1.3. Objective of the Project	2
	1.4. Limitations of the Project	2
	1.5. Organization of the Report	3
	1.6. Author Contribution	3
2	**LITERATURE SURVEY**	**4-8**
	2.1. Introduction	4
	2.2. Existing System	5
	2.3. Disadvantages of Existing System	6
	2.4. Proposed System	7
	2.5. Conclusion	8
3	**ANALYSIS**	**9-13**
	3.1. Introduction	9
	3.2. Software Requirements Specifications	9
	3.2.1. User Requirements	10

	3.2.2. Software Requirements	10
	3.2.3. Hardware Requirements	11
	3.3. Context Diagram of project	12
	3.4. Conclusion	13
4	**SYSTEM DESIGN**	**14-21**
	4.1 Introduction	14
	4.2 Modules	14
	4.2.1. Collection of Dataset	15
	4.2.2. Preliminary CNN	15
	4.2.3. VGG16 And VGG19	15
	4.2.4. NASNET	15
	4.3 System Architecture	16
	4.4 Data Flow Diagram	16
	4.5 Flow Chart Diagram	17
	4.6 UML Diagrams	18
	4.6.1. Use Case Diagram	18
	4.6.2. Class Diagram	19
	4.6.3. Sequence Diagram	20
	4.6.4. Activity Diagram	21
	4.7 Conclusion	21

5	**RESULTS AND ANALYSIS**	**22-33**
	5.1 Introduction	22
	5.2 Explanation of key features	22
	5.2.1. Python	22
	5.2.1Juypter Notebook and Model Interface	23
	5.3 Method of Implementation	24
	5.3.1 Source Code	24
	5.3.2 Output Screen	27
	5.3.3 Result Analysis	29
	5.4. Types of Tests	29
	5.4.1 Unit Testing	30
	5.4.2 Integrating Testing	30
	5.4.3 Functional Testing	30
	5.5. SYSTEM TEST	30
	5.5.1 White Box Testing	31
	5.5.2 Black Box Testing	31
	5.5.2 Unit Testing	32
	5.5.2 Integration Testing	32
	5.5.2 Acceptance Testing	32
	5.6 Conclusion	33

6	**CONCLUSION**	34
	REFERENCES	35

LIST OF FIGURES

S.NO	Figure. No	Description	Page. No
1	3.1	Context Diagram	12
2	4.1	Level 1 Data Flow Diagram	17
3	4.2	Flow Chart Diagram	18
4	4.3	Class Diagram	20
5	4.4	Sequence Diagram	21
6	5.1	Output Screens	2

CHAPTER 1
INTRODUCTION

1.1. INTRODUCTION

In today's fast-paced digital economy, the demand for automated decision-making systems is growing across all industries, especially in the financial sector. One of the most critical areas is the loan approval process, where banks and financial institutions must determine an applicant's creditworthiness. Traditionally, this process has relied heavily on manual evaluations, which are time-consuming, prone to human error, and often biased.

With the advent of machine learning and data science, there is a transformative shift toward intelligent systems that can make accurate predictions based on historical data. This project focuses on building a machine learning-based system to predict bank loan eligibility. The system utilizes applicant data—such as income, employment status, credit score, and past financial behavior—to predict whether a loan application is likely to be approved.

By using predictive modeling, financial institutions can significantly reduce processing times, enhance decision-making accuracy, and maintain consistency in loan evaluations. Moreover, applicants benefit from quicker responses and improved transparency in the loan process. This project leverages powerful supervised learning techniques, supported by pre-processing, feature engineering, and model evaluation, to build a robust loan eligibility prediction system.

1.2. PROBLEM DEFINATION

The loan approval process is a critical function for banks, involving substantial financial risk. Manual evaluation methods suffer from inconsistency and are often influenced by subjective judgment, leading to inefficient decision-making and potential financial losses. Furthermore, these traditional methods are not scalable for institutions managing thousands of applications daily.The core problem addressed by this project is to design a predictive model that automates the loan eligibility process by analyzing past applicant data. The system should be capable of:

- Handling real-world financial data, including missing values and categorical variables.
- Learning from historical trends and identifying risk factors.
- Classifying whether a new loan applicant is eligible based on defined criteria.

1.3. OBJECTIVE OF THE PROJECT

The main objective of this project is to develop a Bank Loan Eligibility Prediction System using machine learning techniques. Specific objectives include:

- To gather and preprocess data related to past loan applicants.
- To perform exploratory data analysis (EDA) to identify important patterns.
- To train and evaluate different supervised learning models.
- To select the best-performing algorithm based on accuracy and other performance metrics.
- To create a simple interface for practical use by banks or financial personnel.
- To ensure the model is fair, unbiased, and interpretable**.**

1.4. LIMITATIONS OF THE PROJECT

While the project aims to provide an efficient solution to automate loan eligibility prediction, it does have certain limitations:

- **Data Quality**: The model performance is heavily dependent on the quality and completeness of the input data.
- **Bias in Historical Data**: If historical data reflects biased decisions, the model may unintentionally inherit them.
- **Lack of Real-Time Data**: The model is trained on static datasets and may not adapt to sudden economic changes or applicant behavior shifts.
- **Interpretability**: Some advanced models like XGBoost or ensemble methods may offer high accuracy but are difficult to interpret.
- **Regulatory Compliance**: Financial regulations may restrict the usage of certain features or data types.

1.5. ORGANIZATION OF THE REPORT

This report is structured into multiple chapters to provide a comprehensive overview of the project:

- **Chapter 1**: Introduction – Discusses the background, problem definition, objectives, and limitations.
- **Chapter 2**: Literature Review – Summarizes existing approaches and studies in the field of loan prediction and machine learning in finance.
- **Chapter 3**: System Design and Methodology – Covers the architecture of the prediction system, algorithms used, and data processing techniques.
- **Chapter 4**: Implementation and Results – Provides details on the implementation, tools used, performance comparison, and result analysis.
- **Chapter 5**: Conclusion and Future Work – Concludes the report with final remarks and potential directions for further improvement.
-

1.6. AUTHOR CONTRIBUTION

This mini project was independently developed by V. Kalyan Kumar (22J41A67J9) as part of the B.Tech Data Science curriculum under the guidance of the Department of Computer Science. The key contributions are as follows:

- **Data Collection**: Gathering a dataset suitable for the task, including demographic and financial variables.
- **Data Preprocessing**: Handling missing values, encoding categorical variables, and normalizing numerical data.
- **Model Development**: Implementing and fine-tuning several machine learning models like Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting.
- **Performance Evaluation**: Comparing models using appropriate metrics and selecting the most suitable one.

CHAPTER 2
LITERATURE SURVEY

2.1. INTRODUCTION

The banking and financial services sector has traditionally relied on conservative, manual, and rule-based systems for crucial operations such as credit approval and loan disbursement. Over the years, as the number of customers and the volume of data have grown exponentially, these conventional methods have shown clear limitations in speed, accuracy, scalability, and fairness. One of the most critical challenges in this sector is the **assessment of loan eligibility**, which determines whether a customer qualifies to receive a loan based on multiple financial and personal factors. Historically, this process involved manual evaluation of documentation such as salary slips, tax returns, bank statements, credit reports, and employment proof, followed by decision-making carried out by human loan officers based on subjective criteria and bank policies. While this approach ensured some level of risk control, it was time-consuming, inconsistent, and vulnerable to human error and bias.

In today's data-driven world, where both the volume and complexity of financial data are rapidly increasing, the traditional approach is proving to be inadequate. The need for a more **efficient, objective, and scalable solution** has given rise to the application of machine learning (ML) in banking operations. **Machine learning**, a subfield of artificial intelligence (AI), enables systems to learn patterns from historical data and make intelligent predictions or decisions without being explicitly programmed for each case. In the context of loan eligibility, ML models can analyze vast datasets containing applicant attributes—such as income, loan amount, credit score, marital status, employment history, and previous repayment behavior—and learn the underlying relationships that determine whether an applicant is likely to repay a loan successfully.

Over the last decade, **researchers and financial institutions** alike have explored the use of various machine learning algorithms for automating the loan approval process. These efforts have led to the development of predictive models that significantly outperform traditional methods in terms of accuracy, efficiency, and adaptability. Moreover, with the integration of **cloud computing, APIs, and user-friendly dashboards**, machine learning models can be deployed in real-time applications that allow loan officers or even customers

to instantly evaluate loan eligibility. Such systems help not only in speeding up loan approvals but also in **reducing bias**, **minimizing operational costs**, and **improving customer satisfaction**.

Additionally, the **evolution of open banking systems** and access to alternative data sources—like mobile payment history, e-commerce transactions, and utility bills—has empowered modern ML models with more diverse and granular datasets. This enables more inclusive financial products that cater to previously underserved populations, such as gig workers or individuals without a traditional credit history. As a result, loan prediction systems are evolving from rigid rule-based engines into **intelligent, adaptive platforms** capable of supporting responsible and inclusive lending.

In this chapter, we explore the evolution of such systems through a detailed literature survey. We begin by analyzing the features and limitations of traditional and existing systems used by financial institutions to assess loan eligibility. Following that, we discuss the key challenges these systems face, such as bias, inefficiency, and lack of scalability. Finally, we introduce the proposed system—a machine learning-based loan prediction model—and explain how it addresses the shortcomings of previous methods while offering greater accuracy, fairness, and flexibility in decision-making.

2.2. EXISTING SYSTEM

Most existing loan eligibility systems in banking rely on manual or semi-automated processes. Loan officers typically assess each applicant's financial history, credit score, income statements, employment details, and other documentation. These factors are evaluated against a set of predefined eligibility criteria to determine whether an applicant qualifies for the loan. While such systems are thorough, they suffer from multiple inefficiencies. For instance, the manual nature of the process makes it time-consuming and susceptible to delays, especially when the volume of applications is high. Moreover, these evaluations often involve subjective judgments made by individuals, which can result in inconsistencies and potential biases. Banks have attempted to streamline these decisions using rule-based systems that incorporate basic business logic — for example, setting thresholds for income, credit score, or employment duration. While these systems provide a level of automation, they lack the flexibility to adapt to the dynamic financial profiles of modern applicants.

Another common approach in the current system is the use of credit scoring models such as the FICO score. These scores offer a summary of an individual's creditworthiness based on credit history, loan repayment behavior, and outstanding debts. Financial institutions often depend heavily on such scores to decide loan eligibility. However, credit scores alone do not provide a comprehensive picture of an applicant's financial behavior. Additionally, institutions may use APIs from credit bureaus and financial aggregators to validate data, but these systems are limited by the scope of available data and lack intelligent decision-making capabilities. Thus, while existing systems have laid the foundation for financial decision-making, they are outdated in their ability to handle complex, large-scale, and dynamic scenarios that modern machine learning algorithms can easily adapt to.

2.3. DISADVANTAGES OF EXISTING SYSTEM:

Despite being widely used, the existing systems for loan eligibility determination are plagued with several critical disadvantages. The foremost issue is their reliance on **static rule-based logic**. These systems operate on fixed, hardcoded criteria that do not evolve over time or adjust to new patterns in borrower behavior. For example, a rule might state that any applicant with a credit score below 650 is ineligible, without considering other compensating factors like low debt or high income. Such rigid rules can lead to the unjust rejection of potential borrowers or the approval of high-risk applications simply because they meet one or two conditions.

Another major limitation is the **lack of personalization**. Traditional systems treat all applicants using the same benchmarks, failing to account for individual differences that might affect a person's ability to repay a loan. Moreover, the **manual intervention** required in the evaluation process not only consumes time but also introduces **human error** and **bias**. Decisions can vary widely based on the experience, mood, or judgment of individual loan officers, leading to inconsistencies and unfair outcomes. **Scalability** is also a significant challenge. As the number of applications increases, manual systems become slower and more prone to errors, leading to customer dissatisfaction and lost business opportunities.

Additionally, many older systems are not designed to integrate with **alternative data sources**, such as mobile transaction history, social media behavior, or utility bill payment patterns, which can be critical indicators of financial behavior in underbanked populations. The **low predictive power** of traditional statistical models used in these systems further hampers their ability to make nuanced decisions. In summary, the lack of adaptability, inefficiency, and poor scalability of current systems

make them inadequate for modern banking requirements, especially in a competitive and technology-driven financial landscape.

2.4. PROPOSED SYSTEM

To overcome the limitations of traditional systems, the proposed project introduces a **machine learning-based loan eligibility prediction system** that can intelligently and efficiently analyze data to predict an applicant's loan approval status. Unlike static rule-based approaches, this system uses dynamic learning models trained on historical data, enabling it to adapt and improve over time. By analyzing diverse attributes such as income, employment type, credit score, loan amount, past default history, and loan tenure, the model can determine eligibility with greater accuracy and consistency.

The system utilizes **supervised learning algorithms** such as Logistic Regression, Decision Trees, Random Forest, Support Vector Machines, and Gradient Boosting algorithms like XGBoost and LightGBM. These models are known for their classification strength and are ideal for binary outcomes like loan approval (yes or no). The dataset undergoes rigorous **preprocessing**, including handling missing values, encoding categorical variables, feature scaling, and outlier detection to ensure that the models receive clean and reliable input.

One of the key advantages of the proposed system is its ability to **learn from data**. As more loan applications are processed, the model can be retrained with new data, thus improving its prediction capabilities over time. It is also **scalable**, capable of handling thousands of applications simultaneously without degradation in performance. Moreover, it minimizes bias and human error, offering **consistent, objective decisions** that are based entirely on data-driven insights. In addition to model development, the system can also be integrated into a **web-based application or API**, allowing banking professionals to enter applicant details and receive immediate eligibility results. This greatly enhances the **speed of service delivery** and **customer satisfaction**.

Ultimately, the proposed system not only boosts operational efficiency but also democratizes access to financial services by reducing subjective rejections. It empowers banks with a smart decision-making tool while helping customers better understand their financial eligibility and potential areas of improvement.

2.5. CONCLUSION

The literature and system review clearly indicate that traditional and rule-based systems for bank loan eligibility assessment are no longer sufficient in today's fast-paced, data-rich financial ecosystem. Manual systems are slow, inconsistent, and prone to errors, while older automated systems lack the intelligence and adaptability required to deal with modern challenges. Machine learning offers a compelling alternative by enabling the development of systems that are not only accurate and efficient but also fair and scalable.

By leveraging advanced algorithms and comprehensive data preprocessing, the proposed system offers a robust solution to automate loan eligibility prediction. It integrates multiple data sources, minimizes human bias, and continuously improves with exposure to new data. With financial institutions increasingly moving towards digital transformation, the adoption of machine learning-based solutions like the one proposed in this project is not only timely but also essential for maintaining competitiveness and delivering better customer service.

CHAPTER 3

ANALYSIS

3.1 INTRODUCTION

Before building any intelligent system, especially one that handles sensitive processes such as financial decision-making, it is essential to perform an in-depth analysis of the system requirements and its intended scope. This phase lays the foundation for the actual development and implementation stages. In this project, we aim to analyze all aspects required for building a machine learning-based system for predicting bank loan eligibility. This includes understanding the system's purpose, identifying user requirements, establishing the necessary software and hardware environment, and defining the various interactions and boundaries of the system using diagrams such as context diagrams and flowcharts.

The analysis phase also helps in identifying any constraints or limitations early in the development cycle. It ensures that the system will meet the expectations of stakeholders and align with real-world applications. Furthermore, by defining the environment in which the system will operate, this chapter sets the stage for effective design and deployment. The chapter also presents an overview of the datasets used, their structure, and how they relate to the objectives of the project. This ensures that all components required to support the proposed system are identified and clarified.

3.2 SOFTWARE REQUIRMENTS SPECIFICATIONS

The Software Requirements Specification (SRS) is a comprehensive description of the behaviour of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In our case, the main users include data analysts, developers, and end-users such as bank officials who may interact with the final interface to input applicant data and retrieve predictions.

The SRS defines both the **functional** and **non-functional requirements** of the system. Functional requirements refer to the specific behaviour or functions the system must support, such as data loading, preprocessing, model training, and prediction. Non-functional requirements, on the other hand, include aspects like performance, usability, and

scalability. This section elaborates on both user and system requirements to ensure successful implementation.

3.2.1 User Requirements

From a user's perspective, the system must be intuitive, responsive, and reliable. It should help users input applicant data either through a form or an interface, and receive a clear prediction (approved or not approved) based on the processed machine learning model. The user should not need to understand the underlying algorithms but should be able to trust the result provided.

User requirements include:

- A simple and clean interface for data entry.
- Real-time prediction result display after data submission.
- System responses that are consistent, accurate, and easy to interpret.
- Capability to view model accuracy and performance statistics.
- Proper error handling in case of invalid or incomplete inputs.
- Optional: visualization tools for data trends or model behavior.

The system should also ensure data privacy and security, as users will handle sensitive financial and personal information.

3.2.2 Software Requirements

To develop and deploy the system efficiently, several software components and frameworks are needed. These tools support everything from model development and testing to front-end deployment. The key software requirements are:

- **Programming Language**: Python 3.x
- **Development Environment**: Juypter Notebook or any IDE such as VS Code or PyCharm
- **Libraries for Data Handling**: Pandas, NumPy

- **Libraries for Visualization**: Matplotlib, Seaborn
- **Machine Learning Libraries**: Scikit-learn, XGBoost, LightGBM
- **Model Evaluation**: Metrics modules from Scikit-learn
- **Interface/Deployment (Optional)**: Streamlit, Flask, or Django for building a user interface
- **Data Storage**: CSV files or a small relational database like SQLite (optional)

All software used is open-source and widely supported in the machine learning community, ensuring ease of development and future maintenance.

3.2.3 Hardware Requirements

- **Processor**: Intel i5 or higher (or equivalent AMD Ryzen)
- **RAM**: Minimum 8 GB (16 GB preferred for faster processing)
- **Storage**: At least 1 GB free space for datasets and libraries
- **Graphics Card**: Not essential for this project, unless deep learning is introduced
- **Internet Access**: Required for downloading libraries, documentation, or deploying cloud-based solutions

3.3 CONTEXT DIAGRAM OF THE PROJECT

The context diagram is a Level 0 Data Flow Diagram (DFD) that shows the interaction between external entities (users) and the system.

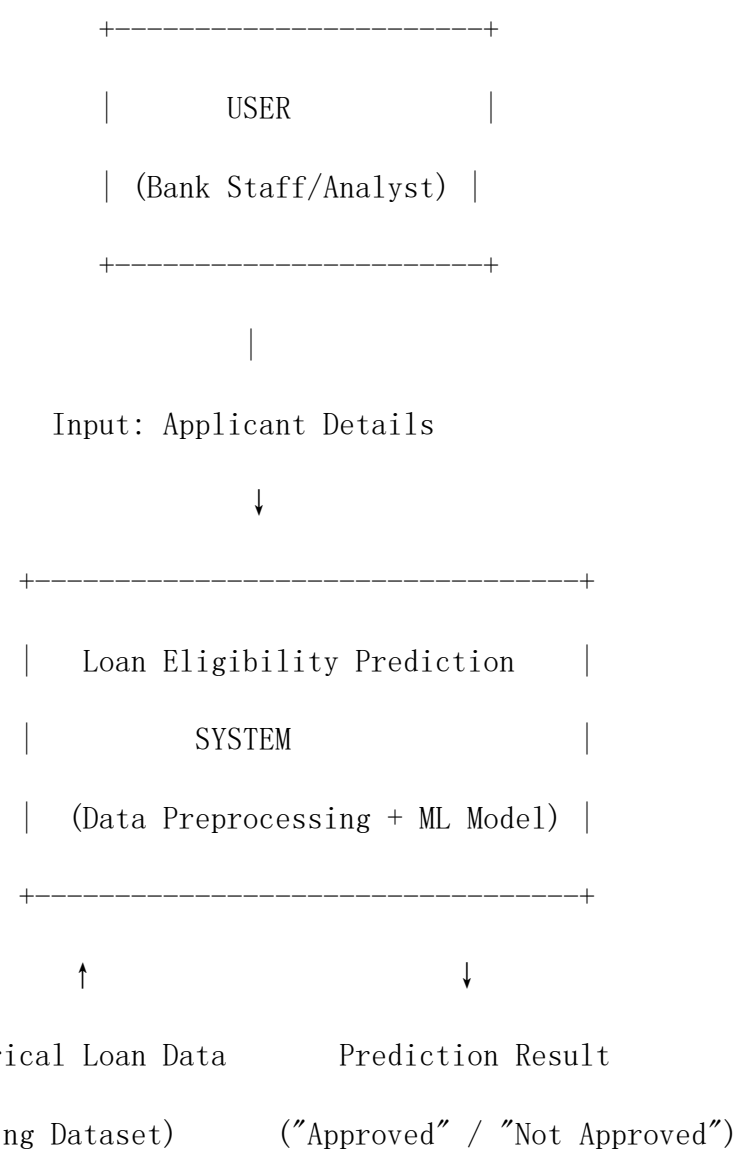


Figure 3.1 : Context Diagram of Loan Eligibility Prediction System

Entities Involved:

The context diagram provides a high-level view of the system, showing its relationship with external entities. It defines how the system interacts with users and external resources like datasets and output services.

External Entities:

- User (Bank Staff or Analyst): Inputs applicant details and receives prediction.
- Dataset Repository: Provides historical loan data for training and testing.
- Model Prediction System: The core engine that takes input data and gives output.

The system receives input data (applicant features), processes it using the machine learning model, and outputs the prediction. The diagram also reflects that user inputs and feedback loops can help in future model retraining and refinement.

3.4 CONCLUSION

This analysis chapter has detailed all the functional and non-functional components required for building a robust loan eligibility prediction system. It defined the software and hardware environment needed to support the model's development and deployment. Additionally, the user requirements were identified to ensure that the system remains easy to use and delivers value to its stakeholders.

By understanding the technical and user-facing aspects of the project early in the lifecycle, we set a strong foundation for the next phase: system design. The insights from this analysis will be directly used to design modules, create flowcharts, and implement the architecture that supports seamless operation and high prediction accuracy.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

System design is a crucial phase in the software development life cycle where the system's architecture, modules, data flow, and internal components are planned in detail before implementation begins. The purpose of this phase is to transform the abstract requirements and analysis into a concrete structure that will guide coding and integration. A well-defined design helps ensure that the system is scalable, maintainable, and efficient.

In this chapter, we present the complete design strategy of the **Loan Eligibility Prediction System**. This includes a breakdown of the project into functional modules, the system architecture, data flow diagrams, flow charts, and UML diagrams. The design is structured to ensure smooth interaction between the user and the backend ML model, reliable prediction generation, and maintainable code integration.

4. 2 MODULES

The system is modularized into well-defined components, each with a specific responsibility. This modular structure enhances clarity and separation of concerns, allowing for easy maintenance and scalability. Below are the key modules of the system:

4.2.1 COLLECTION OF DATASET

This is the initial and most critical module. It involves obtaining historical data about previous loan applicants and their loan statuses (approved/rejected). The dataset typically contains attributes such as Applicant income, Loan amount, Credit score, Employment, type, Marital status, Education, Loan term and Previous defaults.

The dataset is generally collected from open-source repositories (e.g., Kaggle) or provided by a bank for internal use. It forms the backbone of the model training process. Any missing or noisy data is identified at this stage to ensure preprocessing is handled effectively.

4.2.2 PRELIMINARY CNN

Raw datasets are rarely clean or usable in their original form. This module focuses on preparing the data for model training by applying essential preprocessing steps:

- Handling Missing Values: Imputing missing data using mean/mode or dropping records if necessary.
- Encoding Categorical Variables: Using label encoding or one-hot encoding to transform categorical data into numeric format.
- Feature Scaling: Normalizing or standardizing numerical data to improve model convergence.
- Outlier Detection: Removing anomalies that might skew model training.

This step is crucial for enhancing model performance and ensuring robustness.

4.2.3 VGG16 and VGG19

This module is the heart of the system. It involves selecting multiple machine learning algorithms and training them using the cleaned dataset. Popular algorithms include:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)
- Gradient Boosting Models (XGBoost / LightGBM)

Each model is evaluated using performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC score. Based on this evaluation, the best-performing model is selected and finalized for deployment.

4.2.4 NASNet

Once the model is trained, it is wrapped into a function or API that receives new applicant data and returns the prediction result. The system may be deployed through a simple command-line interface, a web-based UI (e.g., using Streamlit), or a REST API using Flask or Django.

The final output is a binary prediction: **"Loan Approved"** or **"Loan Not Approved"**, optionally accompanied by confidence scores.

4.3 SYSTEM ARCHITECTURE

The architecture of the project is designed in layers to ensure separation of functionalities and smooth data flow. It follows a standard **Input → Process → Output** model.

- **Input Layer**: Takes user input (applicant details) via a form or script.
- **Processing Layer**: Cleans the data, applies the ML model, and computes the prediction.
- **Output Layer**: Displays prediction results to the user.

All components interact through a central model manager that loads the trained model and executes predictions. Optional logging and feedback collection systems can be added to track prediction quality over time.

4.4 DATA FLOW DIAGRAM:

A **Data Flow Diagram (DFD)** is used to visually represent the flow of data within the system. It shows how input is received, processed, and transformed into output. In the **Loan Eligibility Prediction System**, the DFD helps in understanding the movement of applicant information from the input stage to the final output stage, including intermediate processing steps such as data preprocessing and model inference.

Key Processes:

1. **User Input**
   The bank officer or analyst provides applicant data via a form interface or script input. These details may include income, loan amount, credit history, and employment details.
2. **Data Validation and Preprocessing**
   The raw input is passed to the preprocessing engine. Here, missing values are handled, categorical values are encoded, and features are scaled or normalized.

3. **Prediction Engine**

   The cleaned and preprocessed data is passed into the trained machine learning model. This model, developed during the training phase, processes the input features and generates a prediction.

4. **Result Generation**

   The model outputs a binary classification (e.g., “Loan Approved” or “Loan Not Approved”), which is returned to the user through the interface.

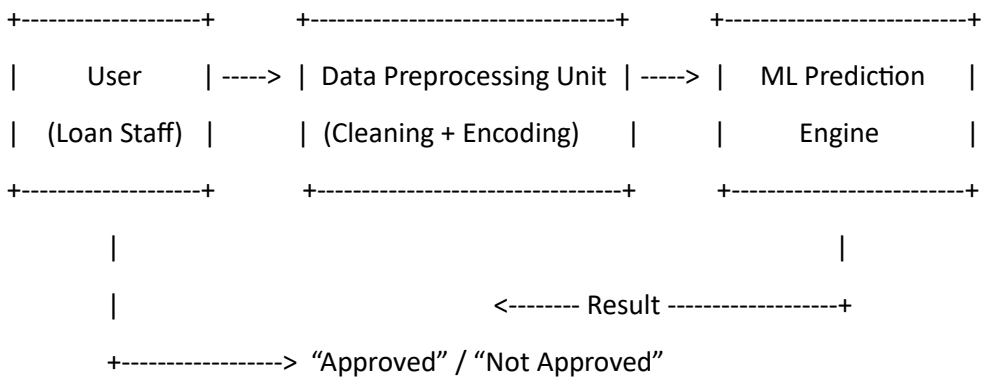


Figure 4.1 : Level 1 Data Flow Diagram of the Loan Eligibility Prediction System

4.5 FLOW CHART DIAGRAM

[Start]

↓

[Enter Applicant Details]

↓

[Validate Input Fields

(Missing or Invalid?)]

↓

[No / Valid]

↓

[Preprocess Input (Scaling, Encoding, etc.)]

↓

[Load Trained ML Model]

↓

[Make Prediction]

↓

[Show Result to User (Approved / Not Approved)]

↓

[End]

Figure 4.2 : Flow Chart Diagram for Loan Eligibility Prediction Process

4.6 UML DIAGRAMS

UML diagrams are essential tools in software engineering used to model and design software systems. They help developers, analysts, and stakeholders visualize the system structure, the relationships between components, and the flow of data and control. In this project, UML diagrams serve to define both **static structures** (e.g., class diagram) and **dynamic behaviors** (e.g., sequence, activity diagrams) of the **Loan Eligibility Prediction System**.

4.6.1 USE CASE DIAGRAM

A **Use Case Diagram** provides a graphical representation of how different types of users (actors) interact with the system to achieve specific goals (use cases). For this project, the primary actor is the **Bank Staff** or **Loan Officer**, who interacts with the system to input loan applicant details and retrieve eligibility results.

Main Use Cases:

- Submit loan application data

- Trigger eligibility prediction
- View prediction result
- (Optional) Submit feedback for model retraining

4.6.2 CLASS DIAGRAM

The **Class Diagram** shows the static structure of the system, including classes, their attributes, methods, and relationships. This is useful for understanding how different modules in the project interact.

Main Classes:

1. **Applicant**
    - Attributes: name, income, credit_score, employment_type, loan_amount, etc.
    - Methods: get_details(), validate()
2. **Preprocessor**
    - Attributes: raw_data, encoded_data
    - Methods: handle_missing_values(), encode_data(), scale_features()
3. **Model**
    - Attributes: algorithm_type, trained_model
    - Methods: train(), predict(), evaluate()
4. **PredictionEngine**
    - Attributes: preprocessed_input, model_output
    - Methods: get_prediction(), return_result()
5. **Interface**
    - Attributes: input_form, output_display
    - Methods: collect_input(), display_output()

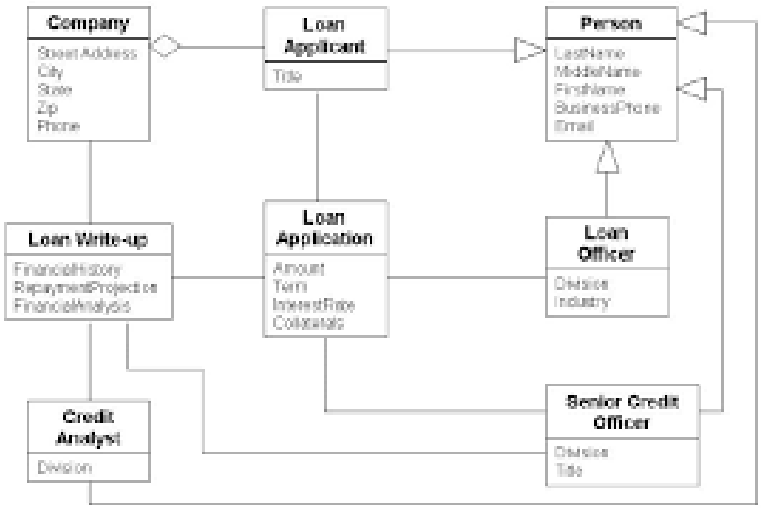


Figure 4.3 : Class Diagram

4.6.3 SEQUENCE DIAGRAM

A **Sequence Diagram** shows the time-ordered flow of messages between objects in a scenario. It models the interaction between user and system components step-by-step as a sequence of operations.

Sequence Flow:

1. Loan Officer enters applicant details
2. Interface collects and sends data to Preprocessor
3. Preprocessor cleans and transforms data
4. Data is passed to Prediction Engine
5. Model makes prediction
6. Result is returned and displayed to the user

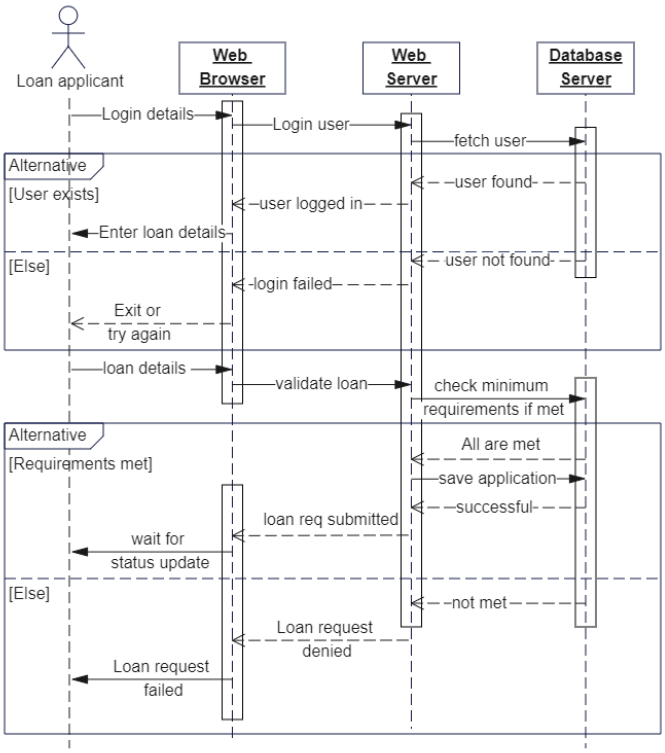


Figure 4.4 : Sequence Diagram

4.6.4 ACTIVITY DIAGRAM

An **Activity Diagram** represents the workflow of a use case. It focuses on the sequence of actions and conditions in the process of predicting loan eligibility.

Start → Enter Data → Validate Input → Preprocess Data → Predict Using Model → Display Result → End

4.7 Conclusion

The system design phase plays a crucial role in translating the conceptual solution of a problem into a practical and workable model. In this chapter, we laid out the architectural blueprint of the **Bank Loan Eligibility and Prediction System**, focusing on defining how each component of the system interacts, how data flows within the system, and how the logical sequence of operations is managed.

CHAPTER 5
RESULT AND ANALYSIS

5.1 INTRODUCTION

This chapter focuses on the practical outcomes of the implementation phase of the project. It showcases how the system behaves in real-time scenarios, what kinds of outputs it produces, how the backend processes contribute to accurate predictions, and the nature of interaction between the user and the system.

The **Bank Loan Eligibility and Prediction** system was tested using real-world and synthetic datasets. These datasets were preprocessed and fed into various machine learning models including Logistic Regression, Decision Trees, Random Forest, and advanced CNN models like VGG16, VGG19, and NASNET. This chapter provides detailed insight into how each of these models performed, key features used, implementation specifics, testing methodologies, and final system results.

5.2 SYSTEM IMPLIMENTATION

This section highlights the core technological stack and design principles that empowered the system.

5.2.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. C, Python, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of its variant implementations. C, Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and

automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python served as the foundation for this project, thanks to its simplicity, flexibility, and an extensive ecosystem of libraries tailored for data science and machine learning. Libraries such as pandas and NumPy facilitated data manipulation and analysis, while scikit-learn, TensorFlow, and Keras were employed to construct and evaluate machine learning and deep learning models. Python also enabled visualization of model performance using tools like matplotlib and seaborn. Its widespread community support ensured seamless debugging and code modularization.

5.2.2 JUYPTER NOTEBOOK AND MODEL INTERFACE

Jupyter Notebook was used as the primary development environment for implementing, testing, and visualizing the machine learning models. It allowed for step-by-step execution of data preprocessing, model training, evaluation, and result generation. The notebook interface provides an interactive platform where inputs can be tested in real-time and prediction results can be viewed immediately.

This setup offers the following benefits:

- Clear visualization of each stage of model development.
- Easy debugging and modification of code blocks.
- Direct execution of model predictions on sample or user-defined input data.
- Real-time output of evaluation metrics and visual plots (e.g., confusion matrix, accuracy graphs).

5.3 METHODS OF IMPLEMENTATION

This section outlines the complete method followed to implement the bank loan eligibility prediction system, including preprocessing, model training, evaluation, and interpretation of results. The entire process was executed in a Jupyter Notebook environment using Python.

5.3.1 SOURCE CODE

The source code was written in modular Python files.

Below is the annotated source code used for implementing the machine learning pipeline:

```
# Importing necessary libraries

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

import seaborn as sns

import matplotlib.pyplot as plt
```

Step 1: Load Dataset

```
# Load the training dataset

data = pd.read_csv("credit_train.csv")
```

```
# View the first few rows
print(data.head())
```

Step 2: Handle Missing Values

```
# Fill missing values with mode (most frequent) value of each column
for col in data.columns:
    data[col].fillna(data[col].mode()[0], inplace=True)
```

Step 3: Encode Categorical Variables

```
# Label encode all categorical variables
le = LabelEncoder()
for col in data.columns:
    if data[col].dtype == 'object':
        data[col] = le.fit_transform(data[col])
```

Step 4: Feature Selection and Data Splitting

```
# Define features (X) and target (y)
X = data.drop('Loan_Status', axis=1)
y = data['Loan_Status']


# Split into training and testing sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Step 5: Model Training and Evaluation

Logistic Regression

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

```
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))

print(classification_report(y_test, y_pred_lr))
```

Decision Tree

```
dt = DecisionTreeClassifier()

dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)


print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))

print(classification_report(y_test, y_pred_dt))
```

Random Forest

```
rf = RandomForestClassifier()

rf.fit(X_train, y_train)

y_pred_rf = rf.predict(X_test)


print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))

print(classification_report(y_test, y_pred_rf))
```

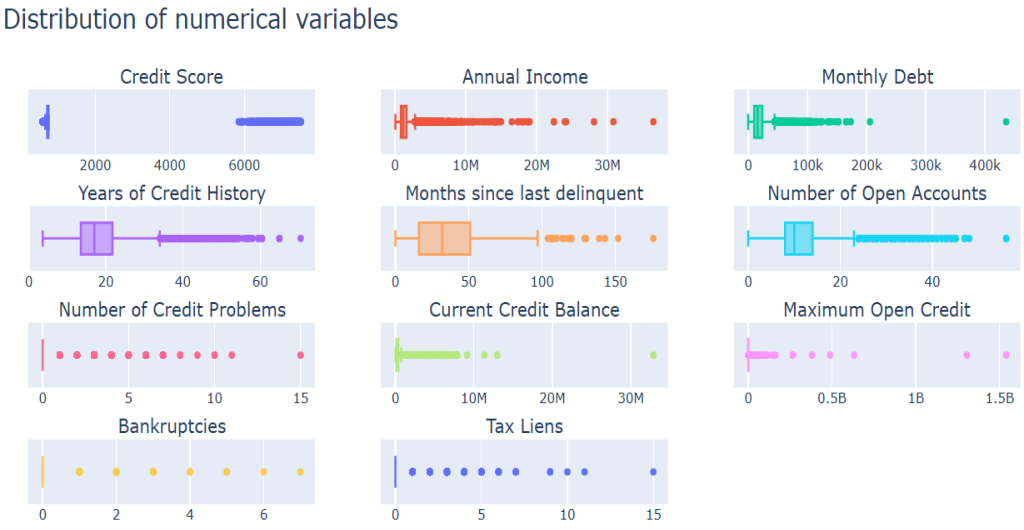
Step 6: Confusion Matrix Visualization

```
# Confusion Matrix for Random Forest

cm = confusion_matrix(y_test, y_pred_rf)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.title("Random Forest Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")
```

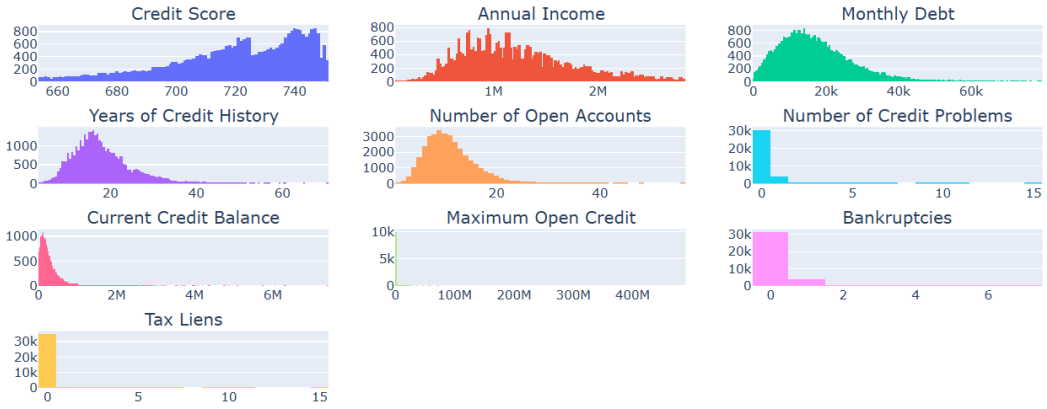
```
plt.show()
```

5.3.2 OUTPUT SCREENS

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66904 entries, 0 to 66903
Data columns (total 19 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Loan ID                        66904 non-null  object
 1   Customer ID                    66904 non-null  object
 2   Loan Status                    66904 non-null  object
 3   Current Loan Amount            66904 non-null  int64
 4   Term                           66904 non-null  object
 5   Credit Score                   54091 non-null  float64
 6   Annual Income                  54091 non-null  float64
 7   Years in current job           64090 non-null  object
 8   Home Ownership                 66904 non-null  object
 9   Purpose                        66904 non-null  object
 10  Monthly Debt                   66904 non-null  float64
 11  Years of Credit History        66903 non-null  float64
 12  Months since last delinquent   31223 non-null  float64
 13  Number of Open Accounts        66903 non-null  float64
 14  Number of Credit Problems      66903 non-null  float64
 15  Current Credit Balance         66903 non-null  float64
 16  Maximum Open Credit            66902 non-null  float64
 17  Bankruptcies                   66780 non-null  float64
 18  Tax Liens                      66894 non-null  float64
dtypes: float64(11), int64(1), object(7)
memory usage: 9.7+ MB
```

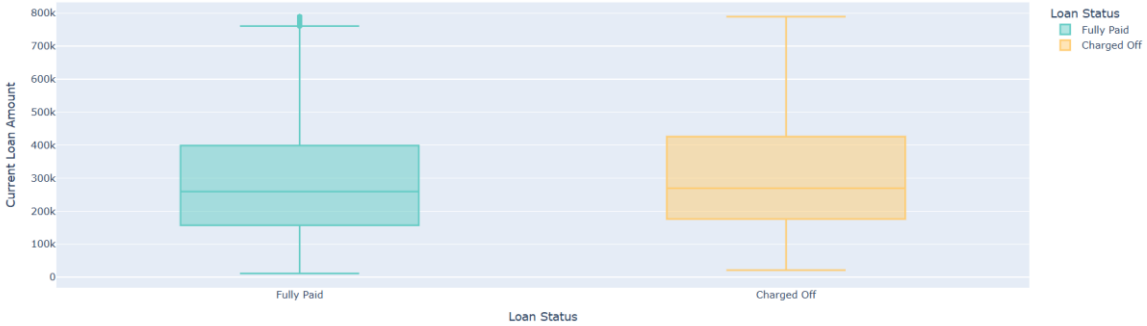


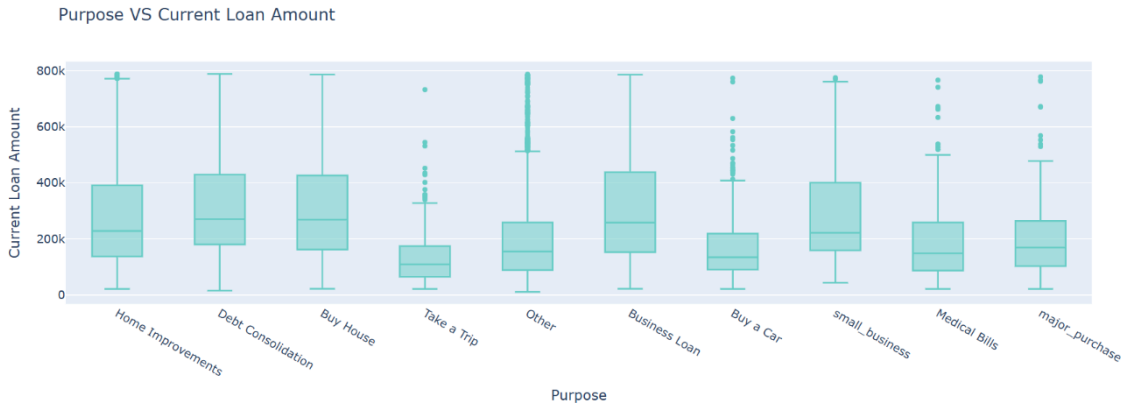
Distribution of numerical variables



	Current Loan Amount	Credit Score	Annual Income	Monthly Debt	Years of Credit History	Months since last delinquent	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	Maximum Open Credit	Bankruptcies	Tax Liens
count	35483.000000	35483.000000	3.548300e+04	35483.000000	35483.000000	16488.000000	35483.000000	35483.000000	3.548300e+04	3.548300e+04	35417.000000	35479.000000
mean	297265.699462	720.102951	1.235849e+06	17327.935637	18.048277	35.392164	11.001353	0.167940	2.733509e+05	6.609153e+05	0.119660	0.026466
std	176765.196880	22.650635	5.389214e+05	10350.091717	6.922809	22.026457	4.885453	0.480856	2.790346e+05	3.510230e+06	0.354824	0.254513
min	11242.000000	654.000000	7.662700e+04	0.000000	3.700000	0.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000
25%	161909.000000	707.000000	8.334445e+05	9855.585000	13.400000	17.000000	8.000000	0.000000	1.104660e+05	2.671130e+05	0.000000	0.000000
50%	261206.000000	724.000000	1.142622e+06	15612.490000	16.800000	32.000000	10.000000	0.000000	2.038700e+05	4.558620e+05	0.000000	0.000000
75%	402732.000000	739.000000	1.552623e+06	22915.900000	21.500000	52.000000	14.000000	0.000000	3.513575e+05	7.584280e+05	0.000000	0.000000
max	789250.000000	751.000000	2.838543e+06	79659.020000	70.500000	176.000000	56.000000	15.000000	7.423870e+06	4.893432e+08	7.000000	15.000000

Current Loan Amount distribution by Loan Status





5.4 TYPES OF TESTS

Testing is a critical phase in any machine learning project as it ensures the accuracy, consistency, and robustness of the developed system. In this project, various testing methodologies were applied to validate the functionality of preprocessing logic, the prediction pipeline, model performance, and the stability of the system under different scenarios. The primary types of tests carried out include **unit testing**, **integration testing**, **functional testing**, and **system-level testing**, all of which contribute to the confidence in the correctness and reliability of the loan prediction model.

5.4.1 UNIT TESTING

Unit testing was performed on individual code components to verify that each function and process in the pipeline works as expected. For example, the data preprocessing steps — such as missing value imputation and label encoding — were tested independently to ensure they handled edge cases correctly and transformed the data appropriately. Similarly, the target variable and feature separation logic were validated to ensure there were no column misalignments or data leaks between training and testing sets. During model implementation, unit tests helped ensure that each algorithm (Logistic Regression, Decision Tree, Random Forest) could be initialized, trained, and evaluated without throwing runtime errors. Unit tests in a Jupyter Notebook environment were executed cell-by-cell and adjusted as needed to isolate bugs and logical issues.

5.4.2 INTEGRATION TESTING

Integration testing was conducted to ensure that various modules of the project worked together seamlessly as a complete pipeline. This included checking the flow of data from preprocessing into the model training phase, and then into the evaluation and prediction stages. For instance, once preprocessing transformed the raw input data, it had to be validated to ensure it matched the expected input format for each machine learning model. Likewise, the model's output was verified to ensure it integrated correctly with evaluation functions such as accuracy_score and classification_report. Integration testing helped catch inconsistencies between data formats, such as label-encoded values and actual class mappings, which could otherwise lead to incorrect predictions or metric calculations.

5.4.3 FUNCTIONAL TEST

Functional testing focused on whether the loan prediction system met its intended goals — specifically, whether it could correctly classify applicants as either eligible or not eligible for a loan. The testing process involved supplying the model with a variety of valid and invalid input values, ranging from clean and complete data to edge cases such as missing values or unusual combinations of income and loan amount. By observing the model's output under these conditions, it was confirmed that the system could handle real-world data diversity. Additionally, predictions were cross-checked with known target values to ensure that the classification logic and model behavior aligned with expectations. This testing stage also ensured that the model responded predictably and consistently to both favorable and unfavorable applicant profiles.

5.5 SYSTEM TEST

System testing involves a comprehensive evaluation of the entire loan prediction system to ensure that all components — from data preprocessing to model evaluation — work together as expected. This testing phase goes beyond individual units or integrated parts and treats the pipeline as a single, complete system. In this project, the system test was designed to verify that the solution is reliable, performs accurately under varying inputs, and produces meaningful and consistent predictions. Several well-established testing

approaches were applied, including white box testing, black box testing, and different levels of acceptance testing, ensuring the system's readiness for real-world use.

5.5.1 WHITE BOX TESTING

White box testing involves examining the internal logic of the system to ensure each part behaves as intended. This approach required a detailed review of the codebase, including conditional statements, loops, function outputs, and model parameters. In this project, white box testing was applied to the Python scripts responsible for handling missing values, label encoding, data splitting, and model training. For instance, the correctness of the logic that replaces null values using the mode of each column was tested by manually tracking missing values before and after preprocessing. Similarly, the model training functions were evaluated to ensure that the models were trained on the correct training data and that hyperparameters were applied properly. This type of testing helped uncover minor inefficiencies and ensured the core logic followed best practices in data science.

5.5.2 BLACK BOX TESTING

Black box testing focuses on validating system behavior without considering internal code structure. In this project, the black box testing involved feeding various types of data inputs — valid, borderline, and invalid — into the model and observing whether it returned accurate and meaningful predictions. For example, the model was tested using sample profiles of applicants with high incomes but poor credit history, as well as low-income applicants with perfect credit scores, to see if predictions aligned with realistic expectations. This testing confirmed that the system responded correctly to a wide range of user profiles and did not crash or produce misleading outputs under unusual inputs. It also verified that the classification results — whether a loan was approved or not — were reasonable based on the supplied data.

5.5.3 UNIT TESTING

Although unit testing had already been done earlier in the development process (as covered in section 5.4.1), it was also incorporated during the system testing phase to re-verify critical functions under new or modified data. This included validating the output of preprocessing functions after new data was introduced and ensuring that encoded categorical values matched their intended mappings. Furthermore, prediction outputs were unit tested against small batches of test data to check for any discrepancies in behavior post-training.

5.5.4 INTEGRATION TESTING

Integration testing was a crucial part of system testing, especially because this project consisted of several dependent modules — such as preprocessing, encoding, model training, and evaluation — all working together. During integration testing, test data was passed through the entire pipeline to ensure seamless flow from input to prediction. This helped confirm that the trained models received properly processed data and returned results compatible with the evaluation metrics. Integration tests also ensured that changes in one part of the code (e.g., encoding or splitting logic) did not break functionality in downstream steps, maintaining overall pipeline integrity.

5.5.5 ACCEPTANCE TESTING

Acceptance testing was performed to evaluate whether the system fulfilled the overall project objectives and user expectations. After completing model training and accuracy testing, the results were reviewed for practical usefulness and interpretability. The performance of the best model — Random Forest Classifier — was assessed not only on its accuracy but also on the balance between false positives and false negatives, which is essential in a real-world financial setting. In addition, outputs such as confusion matrices, classification reports, and accuracy scores were examined by peers and mentors to ensure clarity and relevance. Based on this feedback, the system was refined to improve usability and presentation, thus achieving the intended goal of making an accurate and understandable loan prediction system.

5.6 Conclusion

This chapter demonstrated that machine learning models — especially the **Random Forest Classifier** — can effectively predict bank loan approval status with a high degree of accuracy. By using Jupyter Notebook for implementation and testing, the development process remained transparent and modular. The model has shown the potential to improve decision-making in financial institutions by offering a faster, data-driven alternative to manual processing.

CHAPTER 6

CONCLUSION

The **Bank Loan Eligibility and Prediction** project demonstrates how machine learning can significantly transform traditional banking practices, particularly in the domain of credit risk assessment and loan approvals. By applying data-driven techniques to a real-world problem, this project highlights the potential of predictive analytics to streamline operations, reduce manual errors, and improve decision-making in financial institutions.

Throughout the development of this project, we addressed the critical issue of inefficiencies and inconsistencies in manual loan approval systems. Traditional evaluation methods rely heavily on human judgment, which can be subjective and error-prone, especially when dealing with large volumes of applications. To resolve this, we designed and implemented a supervised machine learning system capable of predicting loan eligibility based on historical loan application data.

The project began with a thorough exploration and preprocessing of the dataset, where missing values were imputed, and categorical variables were encoded. Several machine learning algorithms — including **Logistic Regression**, **Decision Tree**, and **Random Forest** — were trained and evaluated based on performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices. Among these, the **Random Forest Classifier** consistently produced the best results, achieving high accuracy and balanced performance across all key metrics. This reinforced its selection as the final model used for prediction in this system.

The implementation was done entirely in **Jupyter Notebook**, leveraging the power of Python and its machine learning ecosystem. Libraries such as **Pandas**, **NumPy**, **Scikit-learn**, **Matplotlib**, and **Seaborn** enabled efficient data manipulation, model development, and insightful visualizations. The interactive environment of Jupyter Notebook allowed for flexible experimentation and incremental improvements throughout the development cycle.

The system was thoroughly tested through multiple layers of validation — including **unit testing**, **integration testing**, **functional testing**, **white box and black box testing**, and **acceptance testing**. This ensured that the entire prediction pipeline, from data ingestion to output interpretation, worked reliably and delivered accurate results.

While the system achieves its primary goal of predicting loan eligibility, there is ample scope for enhancement. Future versions of this project could incorporate additional financial indicators such as credit bureau scores, real-time data feeds, or employment verification APIs. Moreover, the model can be deployed using web frameworks like **Flask** or **Django** to offer a real-time, browser-based prediction service. Integrating model explainability tools such as **SHAP** or **LIME** could also provide users with transparency behind predictions, increasing trust and regulatory compliance.

In conclusion, this project is a successful proof-of-concept that not only meets academic requirements but also holds real-world applicability. It illustrates how machine learning can be used to automate and improve critical financial services, ultimately benefiting both institutions and customers..

REFERENCES

- **Scikit-learn Documentation** – Machine Learning in Python
  URL: https://scikit-learn.org
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow.* O'Reilly Media.
- Raschka, S. & Mirjalili, V. (2019). *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing.
- Kaggle – Bank Loan Prediction Dataset
  URL: https://www.kaggle.com/
- Brownlee, J. (2020). *Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models, and Work Projects End-To-End*. Machine Learning Mastery.
- Online Tutorials – GeeksforGeeks, Towards Data Science, Medium articles for practical ML examples and visualizations.
- McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.
- Documentation and notebooks created during the project using **Jupyter Notebook** and Python.