Loan Eligibility Prediction System

Tribhuvan University

Institute of Science and Technology

A Final Year Project Submission in

Partial Fulfilment of the Requirement for the Degree of

Bachelor of Science in Computer Science and Information Technology

UNDER THE SUPERVISION OF

Avishek Kuinkel

Submitted By

Nirjala Dahal (24288/7th Semester/2076)

Rabin Bhattarai (24294/7th Semester/2076)

Rishab Shrestha (24298/7th Semester/2076)

Submitted to

TRINITY INTERNATIONAL COLLEGE

Department of Computer Science and Information Technology

Dillibazar Height, Kathmandu, Nepal

January 2023

ABSTRACT

"LOAN ELIGIBLITY PREDICTION SYSTEM" is a web application that focuses on predicting loan eligibility efficiently based on certain user qualities. This system is empowered by the Random Forest Algorithm. Employing efficient and effective modern machine learning techniques, this system is designed to analyze user-provided data which includes cibil-score, commercial assets, annual assets and other various factors, to accurately predict loan eligibility. The User Interface is designed with Keeping Ease of Understanding at mind with open-source Python package "Gradio". The application offers a user-friendly interface for interaction with clear instruction of what to enter and where to enter the data as well as restrict user from entering wrong data. The Random Forest algorithm enhances predictive accuracy, ensuring reliable evaluation of loan eligibility. This system also highlights the current limitations and future possibilities of loan eligibility prediction.

Keywords: *Machine learning, Random Forest Algorithm, Cibil-score, Python, Gradio.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

DFD	Data Flow Diagram
ER	Entity Relationship
IG	Information Gain
KNN	K-Nearest Neighbors
S	Entropy
SVM	Support Vector Machine
UI	User Interface

CHAPTER 1
INTRODUCTION

1.1 Introduction

In today's dynamic financial landscape, lending organizations face the intricate challenge of discerning the creditworthiness of potential clients. The pivotal decision of extending a loan requires a meticulous evaluation of various factors, including income, credit history, and employment details. To streamline this process and enhance efficiency, the "Loan Eligibility Prediction" system, leverages cutting-edge technologies such as data mining and machine learning. The lending industry operates on a fundamental principle providing clients with a specified sum of capital in exchange for repayment with an added interest component. The decision-making process regarding loan approval is multifaceted, involving a comprehensive analysis of client information. Recognizing the complexities of this task, this project aims to develop a system that empowers users to determine the eligibility of a client for a loan swiftly and accurately. This system involves the utilization of diverse mining techniques and machine learning algorithms to extract knowledge from raw data. The project encompasses several key phases, including data collection, cleaning, exploratory data analysis, feature engineering, model selection, training, evaluation, and integration with an application. This comprehensive methodology ensures that the system is not only accurate but also adaptive to the intricacies of individual client profiles. The system's implementation unfolds through a series of steps, starting with the collection of relevant data about past loan applicants. Rigorous data cleaning and preprocessing techniques are applied to ensure the integrity of the dataset. Exploratory data analysis provides critical insights, guiding the feature engineering process. Model selection involves choosing the most suitable machine learning algorithm for binary classification tasks, such as random forest algorithm, decision trees.

The training phase involves optimizing the model's parameters using a carefully curated training dataset. Thorough evaluation on a separate testing set ensures the model's accuracy and reliability. The culmination of these steps results in a robust machine learning model poised to predict loan eligibility accurately.

The predictive power of the model is seamlessly integrated into an application. This user-friendly interface empowers clients and lending professionals alike to input client details and receive real-time loan eligibility predictions. The system not only provides a clear indication of eligibility but also offers insights into the reasoning behind non-approval, fostering transparency and understanding.

The "Loan Eligibility Prediction" system has the potential to revolutionize the lending process, making it more efficient, transparent, and accessible. By leveraging advanced technologies, the system aim to bridge the gap between lending organizations and clients, fostering financial inclusion. The system's predictive capabilities ensure that loan decisions are based on data-driven insights, mitigating risks and enhancing overall portfolio performance.

In conclusion, the project represents a significant step toward creating a technologically advanced solution for the lending industry.

1.2 Problem definition

People In response to the challenges faced by lending institutions and loan applicants in the traditional loan eligibility assessment process, the proposed Loan Eligibility Prediction System aims to revolutionize conventional methods. The manual, time-consuming assessments often result in delays, errors, and biases, impacting the efficiency and fairness of loan approvals. This innovative system leverages advanced data analytics and machine learning techniques to automate and optimize the assessment of loan eligibility. By harnessing the power of data-driven insights, the system strives to provide quicker, more accurate, and unbiased predictions, transforming the lending landscape for both financial institutions and individuals seeking loans. Through this technological advancement, the goal is to create a streamlined and equitable approach to loan eligibility, promoting efficiency and fairness in the ever-evolving financial ecosystem.

1.3 Objectives

- Automate the loan eligibility prediction process using machine learning to increase efficiency and provide highly accurate result.
- Implement measures to address ethical considerations, such as fairness, accountability, and responsible use of applicant data.
- Develop a user-friendly interface for borrowers and lending institution staff, focusing on enhancing the overall user experience.

- Develop the system as a decision support tool, providing valuable insights to lending institution staff.

1.4 Scope and Limitations

Scope:

- Automate the loan approval process by analyzing applicant data, including income, cibil score, and relevant features.
- Integrate the system seamlessly into the workflow of lending institutions, providing user-friendly interfaces for applicants and staff.
- Thoroughly test and evaluate the model's accuracy, addressing biases and ethical considerations in the decision-making process.

Limitation:

- The system's accuracy is sensitive to the quality and representativeness of the training data.
- Challenges may arise in handling rare or extreme cases that deviate significantly from the training data.
- The interpretability of the Random Forest model is limited compared to simpler models.
- Performance may be impacted by imbalances in the distribution of eligible and non-eligible loan cases.

1.5 Development Methodology

The development methodology for the Loan Eligibility Prediction System involves the utilization of Random Forest for feature processing and prediction. The development process begins with preprocessing the input features in the data set through techniques such as handling outliers, scaling numerical values, and mapping categorical variables. The Algorithm is then used to analyze the processed features and predict the loan eligibility status (Accept or reject). The methodology purely uses the facilities/features provide by Random Forest Classifier. As Random Forest is well-suited for classification tasks; it is suitable for predicting loan approval or rejection based on the provided features. The system focuses on efficiently processing user directed input features and making informed decisions regarding loan using our training.

1.6 Report Organization

The report is organized in the following ways:

Chapter 1: It gives the introduction which provides a comprehensive overview of the project, encompassing the problem statement, project objectives, and the defined scope and limitations.

Chapter 2: It provides a existing research identifying key finding and theories. It includes framework of our model its methodology which outline the systematic approach of the model. This chapter is like a bridge, connecting our study to what others have researched and showing the way we'll do our own research.

Chapter 3: This chapter focuses on understanding the requirement of the system It includes the collection of functional and non-functional requirements, specifying features and performance aspects. It includes a feasibility study, evaluating the technical and operational aspects to ensure the practicality and viability of implementing the system.

Chapter 4: This chapter specify the design of input and output mechanisms specifying how data get enter and processed. It gives the information about the workflow within the system. It gives the relationship between the different elements present on the system.

Chapter 5: This section marks the transition from design to actualization. This section details the execution of the system, turning conceptual designs into a functional reality. It encompasses the tools used for implementation, providing insights into the specific technologies applied. Additionally, the chapter outlines resource requirements to run the system.

CHAPTER 2

BACKGROUND WORKS AND LITERATURE REVIEW

2.1 Background Works

The Random Forest Classifier algorithm is applied in the Loan Eligibility Prediction System which is one of the most effective machine learning algorithms.

The technology uses past data that includes attributes like the number of dependents, education level, self-employment status, annual income, loan term, CIBIL score, and different asset valuations in order to estimate loan eligibility. The main algorithm used to evaluate these features and forecast whether a loan will be approved or denied is the Random Forest Algorithm.

The Random Forest's classification and prediction capabilities are utilized by the Loan Eligibility Prediction System. In order to enable the model to identify patterns and correlations between input variables and loan approval outcomes, training entails the use of datasets with labeled cases.

For the Application of such system, we need to only look where the loan is provided and as such it can be integrated in banking app, commercial websites etc.

2.2 Literature Review

In the article "A study on predicting loan default based on the random forest algorithm," that was written in 2019 where research is carried out regarding prediction of loan default by individuals outside of banks jurisdiction in a peer-to-peer network using a machine learning algorithm known as the rainforest algorithm, where they created a model based on the loan data for the first quarter of 2019 from lending Club which was stated as' 'the world largest online financial platform" (Zhua et.al,2019, pp.1-4)In the Article the performance obtained by the random forest algorithm is compared with another machine learning methods like decision tree, the logistic regression and the SVM, and as stated in the article it was done" to predict its performance (accuracy, AUC, F1-Score and recall)" (Zhua et.al, 2019, pp.9)Through the comparison it was evident that based on their needs it outperformed the other algorithms (Zhua et.al, 2019, pp.10).The paper in turn showed that the rainforest algorithm was a suitable algorithm for predicting loan default based on a set of data, an idea of which we plan to implement into our product of loan eligibility prediction system[2].

Research paper “Loan Prediction using Decision Tree and Random Forest,” here proposes a machine learning model for loan prediction using decision tree and random forest algorithms. The paper aims to classify and analyze the nature of the loan applicants based on various attributes, such as gender, income, education, credit history, etc. The paper uses a data set from the banking sector and applies feature selection and data preprocessing techniques to improve the accuracy of the model. The paper compares the performance of the decision tree and random forest algorithms and reports that the random forest algorithm achieves the best accuracy of 85.75%. The paper also discusses the 3 implications and limitations of the proposed model and suggests some future directions for improvement [4].

The paper “Loan Default Prediction on Large Imbalanced Data Using Random Forests,” addresses the problem of predicting loan default on imbalanced and large data sets using machine learning techniques. The paper argues that most existing methods assume balanced data or ignore the class distribution, which leads to poor performance in the minority class. The paper also claims that large data sets pose challenges for efficiency and scalability of learning algorithms. The paper proposes an improved random forest algorithm that assigns weights to decision trees based on their out-of-bag errors in training. The paper also employs parallel computing to speed up the learning process. The paper uses a balanced random forest approach to deal with the imbalanced data problem, either by tuning the sample size parameter or by using Smote sampling method. The paper evaluates the proposed algorithm on a data set from Kaggle.com, which contains 150,000 loan applicants with two classes: default and non-default. The paper compares the proposed algorithm with SVM, KNN, C4.5 and the original random forest algorithm in terms of overall accuracy and balanced accuracy. The paper reports that proposed algorithm outperforms the other algorithms in both metrics. The paper also shows that parallelism can reduce the running time of random forests significantly. The paper contributes to the field of loan default prediction by proposing an improved random forest algorithm that can handle imbalanced and large data sets effectively and efficiently. The paper also provides empirical evidence of the performance and efficiency of the proposed algorithm on a real-world data set [3].

CHAPTER 3

SYSTEM ANALYSIS

3.1 Requirement Analysis

Requirement Analysis is a critical phase in the software development life cycle, encompassing processes to understand, document, and manage the needs and constraints of a proposed system. The System Requirement Specification of project consists of functional and non-functional requirements.

3.1.1 Functional Requirements

Functional Requirement are the basic functionality that are a must to have in a system and can be understood as the basic requirement of a system that are needed of the system and without them the system doesn't exist and can't be design as such our loan predicting system has some functional

- Input: The user should be allowed to enter the user parameters like assets evaluation, education level to determine his eligibility.
- Calculate: A model is designed to carry out prediction based on the input using random forest algorithm.
- Output: Based on calculations user provide various output is displayed.

More process is explained by the Use Case Diagram and the following is the explanation:

- Input Data: The user can input different parameters and attributes
- Output: The user can view the result as per the prediction given by the model
- Train using Dataset: The Developer uses the Dataset to train the model
- Model: Model is created using our Random Forest Algorithm.

Use Case Diagram

The Use case diagram, the system interacts with two primary actors: the "User" and the "Developer." The "User" engages with the system by providing input data, specifying different parameters and attributes for processing. Subsequently, the system utilizes the Random Forest Algorithm to create a predictive model ("Model"). On the other hand, the "Developer" interacts with the system to train this model using a specified dataset. The system's output, representing the result of the prediction generated by the model, is then presented to the user. This diagram encapsulates the essential functionalities of the system, illustrating the flow of actions from user input, model training, to the final prediction output, thereby providing a comprehensive overview of the use cases and interactions within the system.

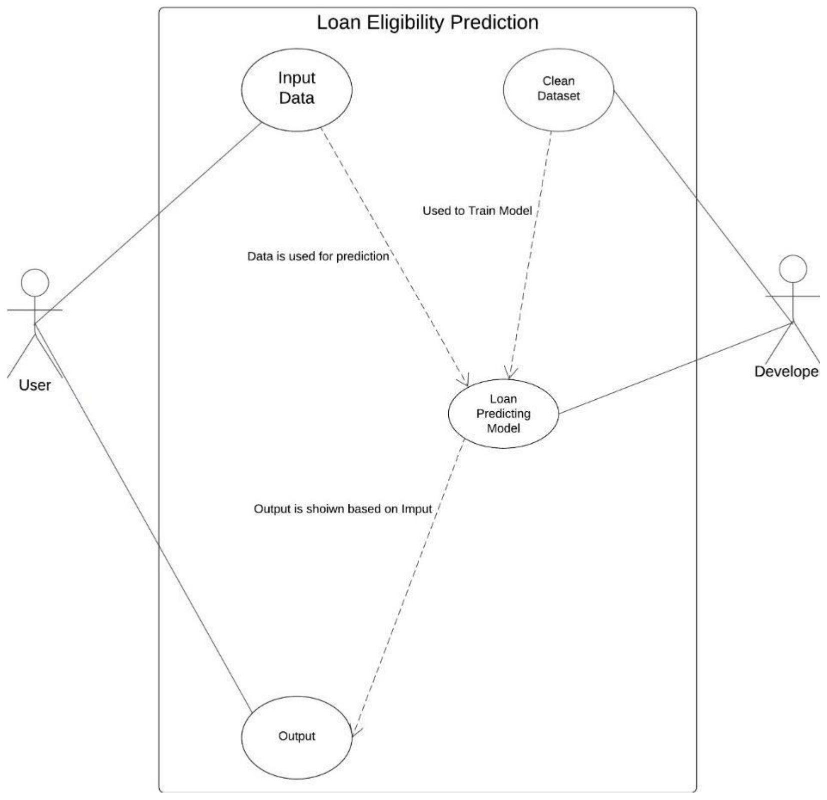

Loan Eligibility Prediction
Input Data
Clean Dataset
Used to Train Model
Data is used for prediction
User
Develope
Loan Predicting Model
Output is shoiwn based on Imput
Output

Figure 1: Use Case Diagram for Email Spam Classifier

3.1.2 Non-Functional Requirement

Some of the non-functional requirement of loan eligibility prediction system are summarized as following:

- Availability: This system can be accessed by anyone and from anywhere using PC.
- Performance: The system should provide loan eligibility predictions within a specified time frame.
- Ease of Use: The system has a user-friendly interface so that any user can use the system without facing any difficulties.
- Reliability: The system should be consistency and reliable predictions for loan eligibility based on the input data.
- Maintainability: The system's code should adhere to coding standards and be well-documented for ease of maintenance.

3.2 Feasibility Study

A feasibility study is a systematic analysis carried out to see whether a suggested project is feasible and workable. It involves a thorough analysis of the project's operational, legal, technological, and economic aspects to determine its ability to succeed. The following are the feasibility concerns in our project:

3.2.1 Technical Feasibility

In the development of Loan Eligibility Prediction System, Python is used for implementation of Classification and regression Algorithm. Gradio is used for designing and implementing the User Interface. So, this Loan eligibility prediction system is technically feasible.

3.2.2 Operational Feasibility

It's simple and fast to conduct such a system. For the same project different data mining tools have been used and it is found that Random Forest Algorithm from our literature reviews and it shown to have the highest accuracy on its performance in data mining than the SVM, KNN, C4.5 and other methods. As such Random Forest is operationally Feasible as it provides tolerable speed and high accuracy.

3.2.3 Schedule Feasibility

Our project maintained a timely schedule by adopting a web-based approach that suited the available time constraints. The execution occurred during reasonable hours, ensuring efficiency. The Gantt Chart below outlines the estimated project schedule.

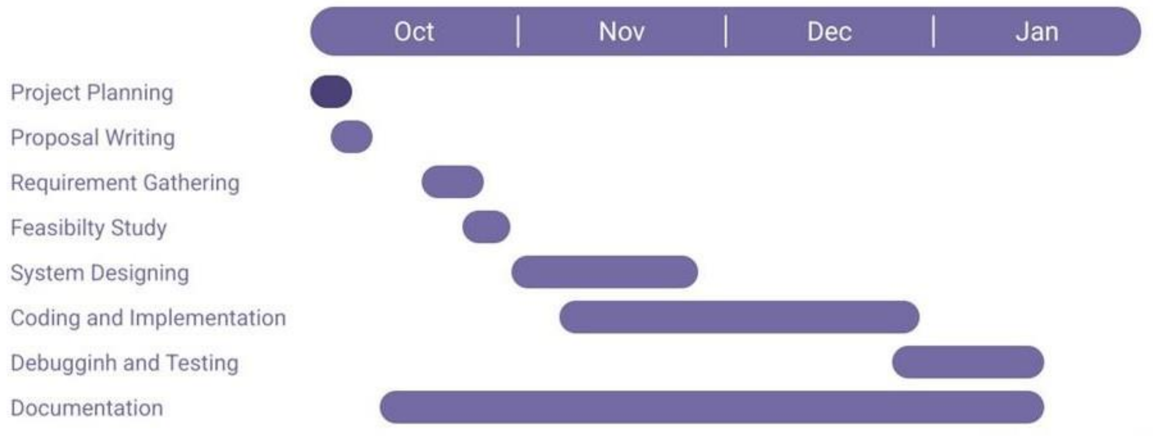


Figure 2: Gantt chart of the project

3.3 Structuring System Requirements

3.3.1 Process Modeling

A process model is a description of a process at the type of level. Since the process model is at the type of level, a process is an instantiation of it.

- **Level-0 DFD:**

  This level-0 DFD describes the system in which there is one user and one system where the user put their applicant data. Then the system predicts whether the customer is eligible for loan or not according to their input data.

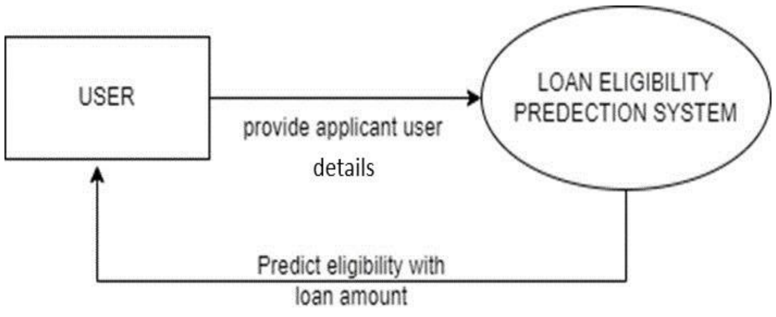

USER
provide applicant user details
LOAN ELIGIBILITY PREDECTION SYSTEM
Predict eligibility with loan amount

Figure 3: 0 level DFD

- **Level-1 DFD:**

  The Level-1 DFD builds upon the Level 0 DFD, delineating the loan prediction system into three key components: Input, Model, and Output. The Input component incorporates applicant details, financial history, and relevant data fed into the system for loan prediction. The Model component serves as the processing unit, employing machine learning algorithms (Random Forest Algorithm) to analyze input data and determine loan approval. The Output component presents the system-generated results, including predicted loan status and insightful recommendations. This structured breakdown facilitates a high-level understanding of the essential components involved in the loan prediction system.

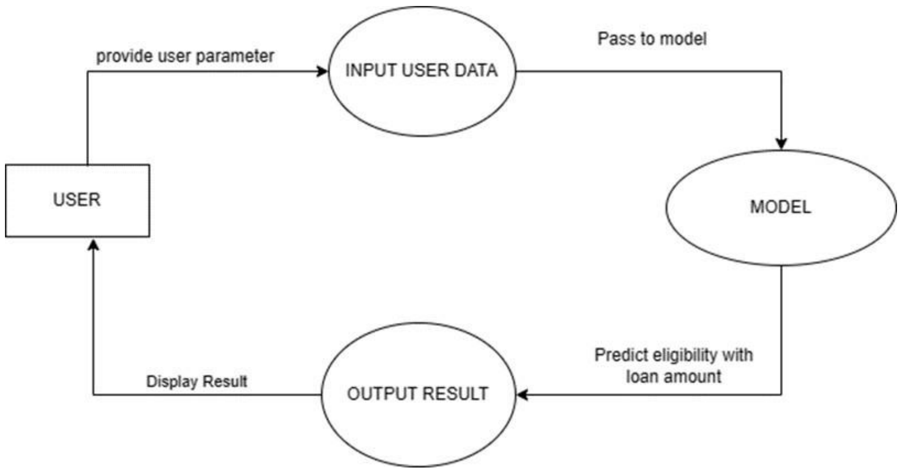


Figure 4: 1 level DFD

- **Level-2 DFD:**

  In the Level 2 Data Flow Diagram (DFD) for the Loan Prediction System employing the Random Forest algorithm without import, each primary process from the Level 1 DFD is meticulously expanded to unveil a detailed sequence of sub-processes and their interconnections. The Model Process is disintegrated into five key sub-processes: Data Preprocessing, Feature Selection, Model Training, Trained Model and Output Generation. These sub-processes encompass specific entities such as Raw Input Data, Transformed Data, Selected Features, Trained Model, Input Data for Prediction, Loan Prediction Results, and Final Output. Data stores, including modules for Data Cleaning, Feature Selection, Random Forest Training, Trained Model Handling, and User Interface, facilitate the manipulation and storage of data at various stages. The addition of the Trained Model Handling sub-process emphasizes the crucial step of managing the trained model within the system, ensuring its seamless integration into the prediction process.

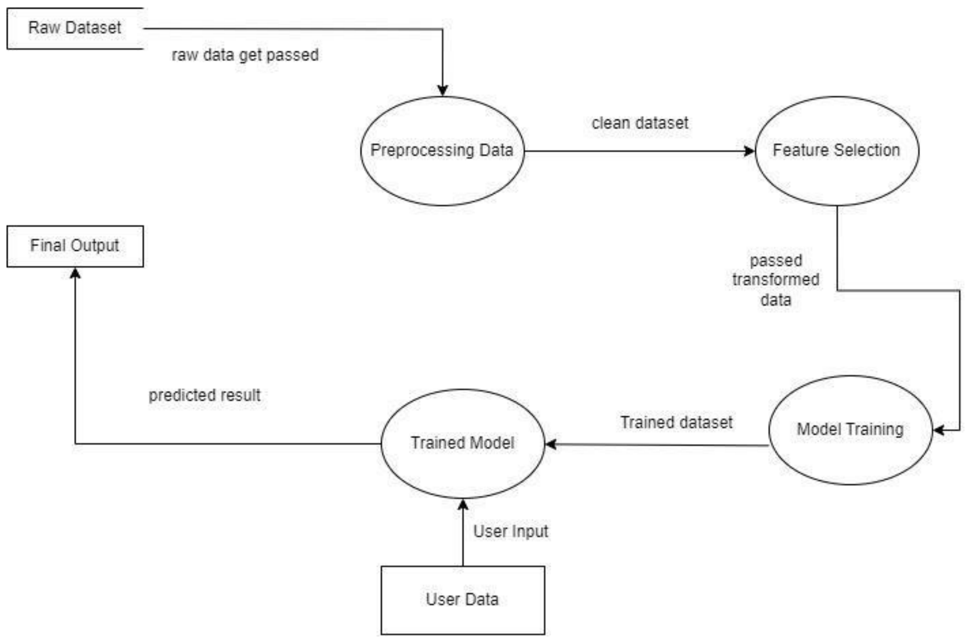


Figure 5: 2 level DFD

3.3.2 Entity Relationship Diagram

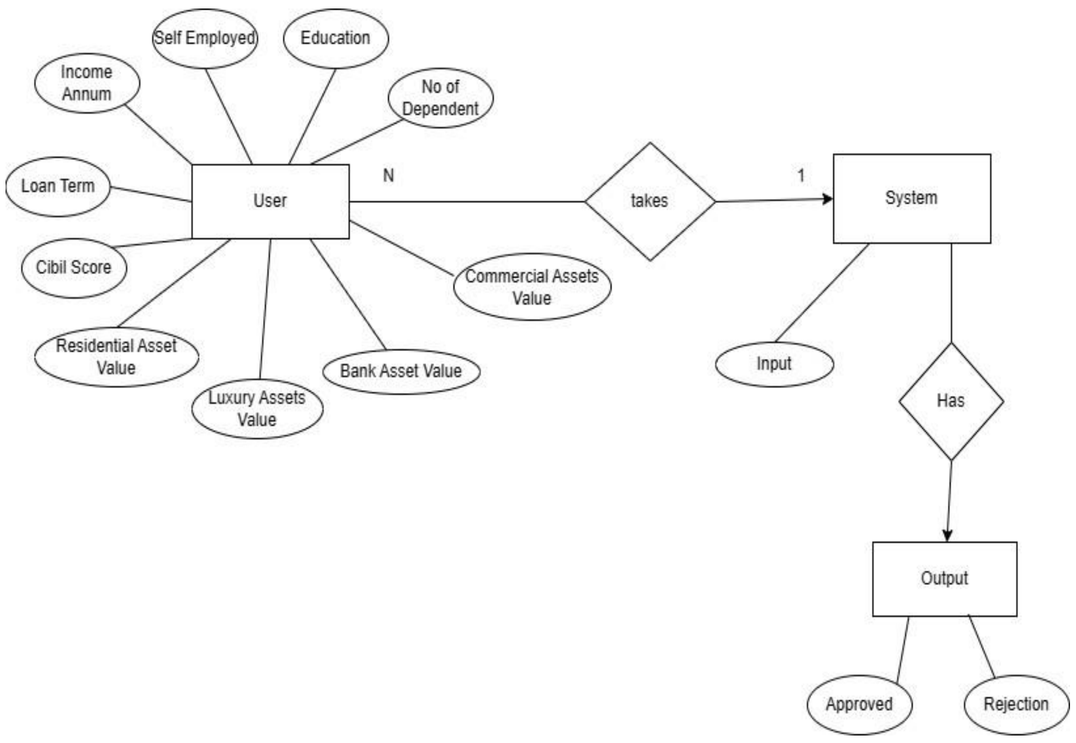


Figure 6: ER diagram

Entity-Relationship (ER) diagram for the loan prediction system, there are two main entities: User and System. The User entity has attributes representing user data crucial for loan prediction. The System entity is connected to the User entity through the "Takes" relationship, indicating that the system utilizes user data for loan assessment. Within the System entity, there is an attribute named Input, signifying the data received from users. The System entity is further linked to another entity named Output through the "Has" relationship. The Output entity encompasses two attributes: Approved and Rejection, capturing the system's final decision on loan approval or rejection. This ER diagram visually outlines how user data is processed by the system, leading to a clear output regarding the loan application status.

CHAPTER 4

SYSTEM DESIGN

4.1 SystemDesign

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

4.1.1 Process Design

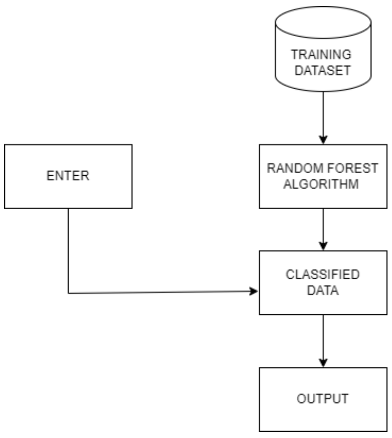


Figure 7: Process Design

As the user enters the system input interface is displayed. Here users can enter the attributes and find out if s/he is eligible for a loan or not. The classified results are displayed on the display interface. If the user is eligible, the system will display the approved loan amount that they can borrow from the bank; otherwise, the system will provide reasons for the loan rejection. Hence, this sums up the overall processing of the system.

4.1.2 Framework of the model

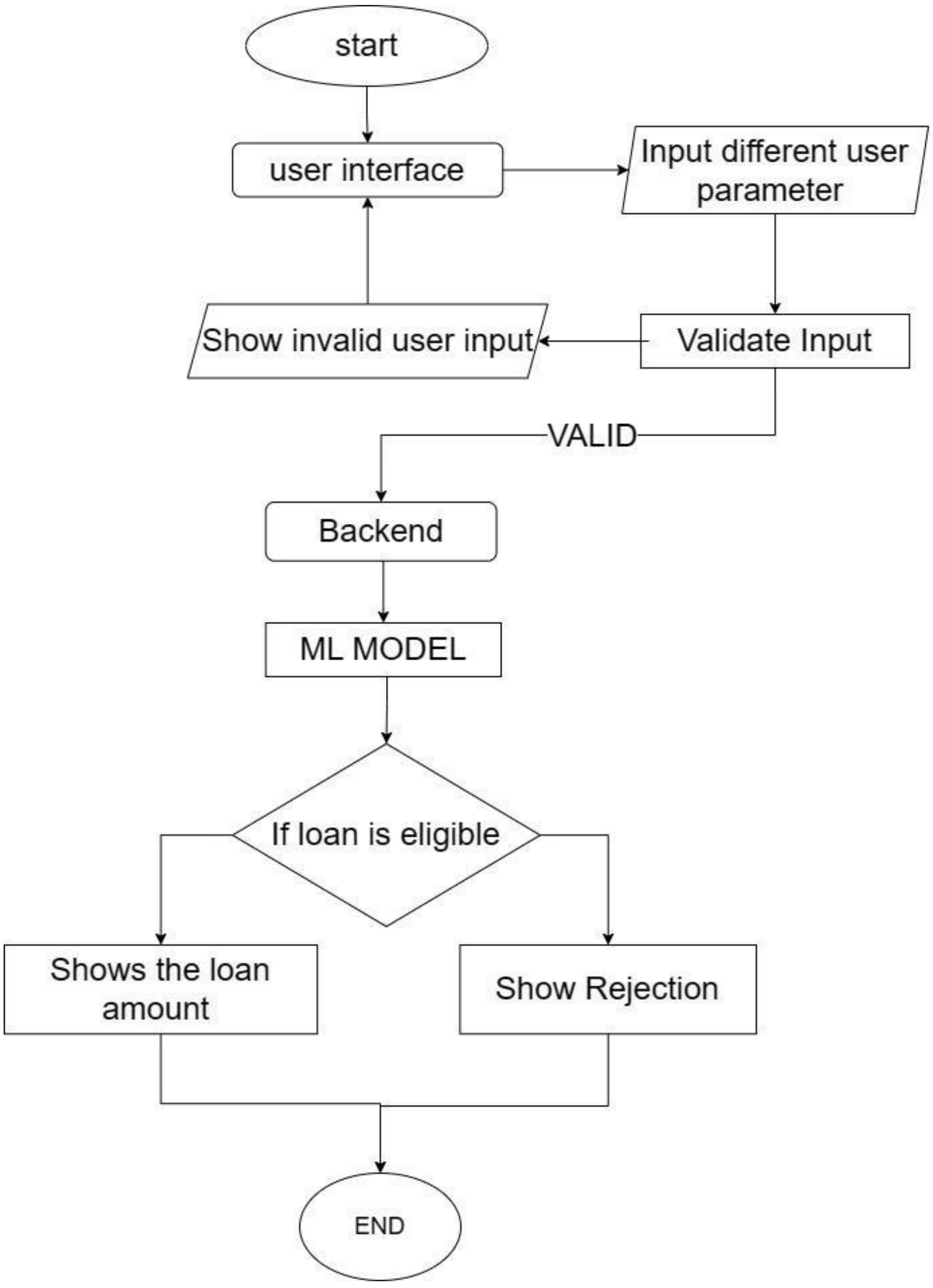


Figure 8: Framework of the system

The framework we have outlined for our loan eligibility prediction system is well-structured. Let's break down each step:

1. UI (User Interface):

The first stage of our loan eligibility prediction system centers on users inputting essential information via a user-friendly interface. In the event of any invalid input, the system promptly notifies users, alerting them to the ineligibility of their provided information. Conversely, when the input is deemed valid, the system seamlessly transitions to the subsequent phase, facilitating the smooth passage of input through the chain of processes. This careful validation and transition mechanism ensures a user-friendly and error-responsive experience as the system progresses through its workflow.

2. Backend:

In the implementation phase, our backend intricately processes the validated user inputs, utilizing the Random Forest Algorithm to assess loan eligibility. If a user is found eligible, the system seamlessly proceeds to display the corresponding loan amount. Conversely, if a user is deemed ineligible, the system tactfully communicates the specific reasons for rejection. This thoughtful approach ensures a transparent and informative user experience, providing clear insights into the loan eligibility decision-making process.

This meticulously designed framework ensures a logical progression, commencing with user input validation, navigating through the implementation of the Random Forest Algorithm.

4.2. Algorithm/Methods/Theory

I. Random Forest

Based on this application requirements and personal study carried out related to the project, it is evident that the utilization of the random forest algorithm is crucial for achieving objectives most effectively. The Random Forest algorithm serves as a versatile solution, adeptly employed for both classification and regression tasks, aligning seamlessly with the diverse needs of our application.

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputs the mode of the classes for classification tasks or the mean prediction for regression tasks. Each decision tree in the ensemble is trained on a random subset of the training data, and at each split, a random subset of features is considered. This randomness injected into the training process enhances the model's robustness, generalization, and resistance to overfitting. The final prediction is then made by aggregating the predictions of all the individual trees, resulting in a robust and accurate model that is widely used in various machine learning applications.

Working:

I. Bagging

Random forest starts by creating multiple subsets of the training data through bootstrapped sampling. It involves selecting datasets randomly from original datasets with replacements. Here it uses the sampling with replacement concepts which means each data points has equal probability of getting selected.

II. Building multiple decision tree

According to the subsets created a decision tree is built for each subset. However, at each node of these trees only a random subset of features is considered for splitting. It's process:

- Calculate the entropy for entire datasets.

  $\text{Entropy}(S) = - \sum p_i * \log_2 (p_i); i = 1 \text{ to } n]$ ....(i)

  Where, n is the total number of classes in the target column p_i is the probability of class 'i' in the dataset.
- For each attribute/feature

  Calculate entropy for all its categorical values. Calculate information gain for the feature.
  $IG(S, A) = \text{Entropy}(S) - \sum ((|S_v| / |S|) * \text{Entropy}(S_v))$ ....(ii)

  IG (S, A): Information Gain for a feature column A in a dataset S Entropy(S): Entropy of the entire dataset S. It quantifies the disorder in the dataset.

  S_v: It is a subset of the dataset S for which the feature column A has
  a specific value v.

  $|S_v|$: The number of rows or instances in the subset S_v.

  $|S|$: The total number of rows or instances in the original dataset S.

- Find the feature with maximum information gain.
- Repeat it until we get the desired tree for each decision tree.

III. Growing the trees:

It involves recursively splitting nodes based on selected features until a stopping criterion is met. It might involve specifying criteria like maximum depth, minimum number of samples required and other criteria to split a node.

IV. Predictions:

Once the stopping criteria is met, there will be no more splitting. When making the classification prediction using random forest a higher number of classes becomes the predicted class. For regression, each tree predicts the values, and the final prediction becomes the average of all tree values.

V. Ensemble of trees:

Once all the trees are trained, they are combined to form the random forest ensemble. When making the prediction on new data, each tree in the ensemble provides its own prediction, and the final prediction is determined by aggregating these individual predictions.

4.2.1Algorithm implementation (Pseudocode)

Importing the necessary libraries

```
pandas as pd
accuracy score
numpy as np
pyplot as plt
```

Load the dataset

```
loan=pd.read_csv('loan.csv')
```

Data preprocessing and feature engineering

- **Renaming the column**

```
loan=loan.rename(columns={' no_of_dependents':'no_of_dependents',
                          ' education':'education',
                    ' self_employed': 'self_employed',
                    ' income_annum':'income_annum',
                    ' loan_amount':'loan_amount',
                    ' loan_term':'loan_term',
                    ' cibil_score': 'cibil_score',
                    ' residential_assets_value': 'residential_assets_value',
                    ' commercial_assets_value':'commercial_assets_value',
                    ' luxury_assets_value':'luxury_assets_value',
                    ' bank_asset_value':'bank_asset_value',
                    ' loan_status':'loan_status'
                   })
```

- **Checking and removing the null value**

```
Loan.info()
```

- **Removing the outliers**

```
for i in range(3):
    for co in col:
        Q1 = loan[co].quantile(0.25)
        Q3 = loan[co].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        loan = loan[(loan[co] >= lower_bound) & (loan[co] <= upper_bound)]
```

- **Encoding categorical variables**

```
custom_mapping = {' Graduate': 1, ' Not Graduate': 0,'Graduate':1,'Not Graduate':0}
loan['education'] = loan['education'].map(custom_mapping)
custom_mapping1 = {' Yes': 1, ' No': 0,'Yes': 1,'No': 0}
loan['self_employed'] = loan['self_employed'].map(custom_mapping1)
custom_mapping2 = {' Approved': 1, ' Rejected': 0,'Approved': 1, 'Rejected': 0}
loan['loan_status'] = loan['loan_status'].map(custom_mapping2)
```

- **Droping and splitting the data**

```
X = loan.drop(['loan_status', 'loan_amount', 'loan_id'], axis=1).values
y = loan['loan_status']. values
```

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

Training the model

- **Creating the decision tree class:**

```
Function __init__(self, max_depth=None):
    self.max_depth = max_depth
    self.tree = None

  Function fit (self, X, y, depth=0):
    unique_classes, counts = get_unique_classes_and_counts(y)

    if len(unique_classes) == 1 or depth == self.max_depth:
        return {'value': unique_classes[0]}

    feature_index, threshold = get_best_split(X, y)
    left_indices = X[:, feature_index] <= threshold
    right_indices = ~left_indices

    left_subtree = fit(X[left_indices], y[left_indices], depth + 1)
    right_subtree = fit(X[right_indices], y[right_indices], depth + 1)

    return {'feature_index': feature_index, 'threshold': threshold,
            'left': left_subtree, 'right': right_subtree}

 Function get_best_split(self, X, y):
    best_info_gain = -1
    best_feature, best_threshold = None, None

    for feature_index in range(X.shape[1]):
        unique_values = get_unique_values(X[:, feature_index])

        for threshold in unique_values:
            info_gain = calculate_information_gain(X, y, feature_index, threshold)

            if info_gain > best_info_gain:
                best_info_gain = info_gain
                best_feature = feature_index
                best_threshold = threshold

        return best_feature, best_threshold

    Function calculate_information_gain(self, X, y, feature_index, threshold):
        left_indices = X[:, feature_index] <= threshold
        right_indices = ~left_indices

        left_entropy = calculate_entropy(y[left_indices])
        right_entropy = calculate_entropy(y[right_indices])
        parent_entropy = calculate_entropy(y)

        weight_left = len(y[left_indices]) / len(y)
        weight_right = len(y[right_indices]) / len(y)

        information_gain = parent_entropy - (weight_left * left_entropy + weight_right *
right_entropy)
        return information_gain

    Function calculate_entropy(self, y):
        _, counts = get_unique_classes_and_counts(y)
        probabilities = counts / len(y)
        entropy = -sum(probabilities * log2(probabilities + 1e-10))
        return entropy

    Function predict_single(self, node, sample):
        if 'value' in node:
            return node['value']

        if sample[node['feature_index']] <= node['threshold']:
            return predict_single(node['left'], sample)
        else:
            return predict_single(node['right'], sample)

    Function predict(self, X):
        return [predict_single(self.tree, sample) for sample in X]
```

- **Creating random forest:**

```
Function __init__(self, n_trees=100, max_depth=None, random_state=None):
        self.n_trees = n_trees
        self.max_depth = max_depth
        self.random_state = random_state
        self.trees = []

    Function fit(self, X, y):
        For _ in range(self.n_trees):
            tree = DecisionTree(max_depth=self.max_depth)
            tree.tree = tree.fit(X, y)
            self.trees.append(tree)

    Function predict(self, X):
        Predictions = np.array([tree.predict(X) for tree in self.trees])
        Return np.round(np.mean(Predictions, axis=0))

    Function predict_proba(self, X):
        Predictions = np.array([tree.predict(X) for tree in self.trees])
        Return np.mean(Predictions, axis=0)
```

For Loan Amount:

```
A = amount.drop('loan_amount', axis=1)
b = amount['loan_amount']
A_train, A_test, b_train, b_test = train_test_split(A, b, test_size=0.2, random_state=42)
model = RandomForestRegressor(n_estimators=100,max_depth=15,random_state=42)
model.fit(A_train, b_train)
predictions = model.predict(A_test)
```

Now, the model is trained and evaluated.

For a new data point (user input), follow these steps:

```
user_data = np.array([no_of_dependents, education, self_employed, income_annum,
                      loan_term, cibil_score, residential_assets_value,
                      commercial_assets_value, luxury_assets_value, bank_asset_value])
```

For Result:

```
if prediction ==1:
    print(f"Loan is Approved")
    money=model.predict(new_data_point.reshape(1,-1))[0]
    print(round(money))
else:
    print(f"loan is rejected")
```

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implemetation

In the implementation of the loan eligibility system using Random Forest, the user provides information to the system. The system captures and processes this input through a Random Forest algorithm, which evaluates the eligibility of users for loans. The algorithm considers various factors, such as income, credit score, and assets, to make predictions. Upon determining loan eligibility, users receive feedback.

5.1.1 Tools used

For the development and designing of the system following tools have been used:

CASE Tools

- `Draw.io: Draw.io was used to draw various diagram to represent the stem architecture for Ease of understanding and also planning for example DFD Diagram (DFD 0,1,2), ER diagram. These diagrams were used to represent various aspect of our system

Back End Tools

- Python: The primary programming language for development of all of our models as well all the task carried out in our system ranging from importing data to trainingthe model as well as cleaning the data as well as predicting the results.
- Notebook: Jupiter Notebook (Python Notebook) was used as our tool to write our code in python which provides an interactive environment and execute Python codein a step-by-step manner to execute and visualize our code/model. It also aids in ease in Documentation for our code as ease in commenting and managing our code.

Front End Tools

- Gradio: Gradio was used to create a UI which would run locally on our pc and interface out notebook and it allows for easy User Interaction to Enter the User Dataand also display the result as well. Gradio provided ease in development as no individual development of to the backend was needed and it also allowed controlleduser input with restriction.

5.1.2 Resource Requirement

The system was developed using the resource mentioned below:

Hardware Requirement

- Personal Computer

Software Requirement

- Windows OS
- VS CODE
- Jupyter Notebook

5.2 Testing

All functional modules of the system are tested. The results of the different phases of testing are evaluated and then compared with the expected output. Errors were uncovered, debugged and corrected. Some test cases are designed and executed during this stage.

5.2.1 Unit Testing

In unit testing, each module are separately tested in an attempt to find any possible errors. The user interface, trained model and classifiers are tested with various possible inputs to make them error free as well as more reliable.

Table 1: Unit Testing

Test Case	Description	Expected Result	Actual Result	Status
Enter attributes or data	User enters the data from GUI	The input data should be processed	The data was processed	Successful
Get Prediction	Make prediction for the attributes that is taken from the user	The function take data as input and returns the predicted result	The function returned error	Failed
Get Prediction	Make prediction for the attributes that is taken from the user	The function take data as input and returns the predicted result	The function returned the predicted output as loan accepted with loan predicted amount	Successful

Train Model	Training the model with training datasets.	The model should take dataset to train itself and predict eligibility accurately	The model object ready for prediction, along with any relevant parameters	Successful

5.2.2 Integration Testing

After the entire unit are properly tested each module are integrated systematically to form a complete system. The system was tested again to know whether it is functioning properly or not after integration.

Table 2: Integration Testing

Test case	Description	Expected Result	Actual Result	Status
Enter Data	User enter the data and run the system	User should be able to see the prediction made by the system	The prediction was displayed on the interface	Successful

5.2.3 System testing

The system was tested as whole. Different varieties of testing and training datasets were used to check the system is giving accurate prediction.

Table 3: System Testing

Test Case	Input	Expected Result	Actual Result	Status
Loan Eligibility Prediction	Number of dependents: 5 Education: 1 Self Employed: 1 Annual Income:	The system should take data as input and display the	The system as able to display the eligibility with predicted amount	Successful

	8700000 Loan Term: 4 CIBIL Score: 678 Residential Assets Value: 22500000 Commercial Assets Value: 14800000 Luxury Assets Value: 29200000 Bank Assets Value: 4300000	eligibility of loan		

5.3 Result Analysis

The overall system is trained on a dataset containing various intents and their corresponding responses. Result analysis for a loan eligibility prediction system involves assessing the system's performance through various metrics such as accuracy, confusion matrix, precision, recall, F1 score, by comparing the system's predictions with actual outcomes, we can determine the accuracy of its predictions and identify any misclassifications, false positives, or false negatives. These metrics provide valuable insights into the system's ability to correctly classify loan applicants as eligible or ineligible. Additionally, interpretation of the results helps in identifying areas for improvement, such as adjusting model parameters or decision thresholds, to enhance the system's predictive performance and reliability.

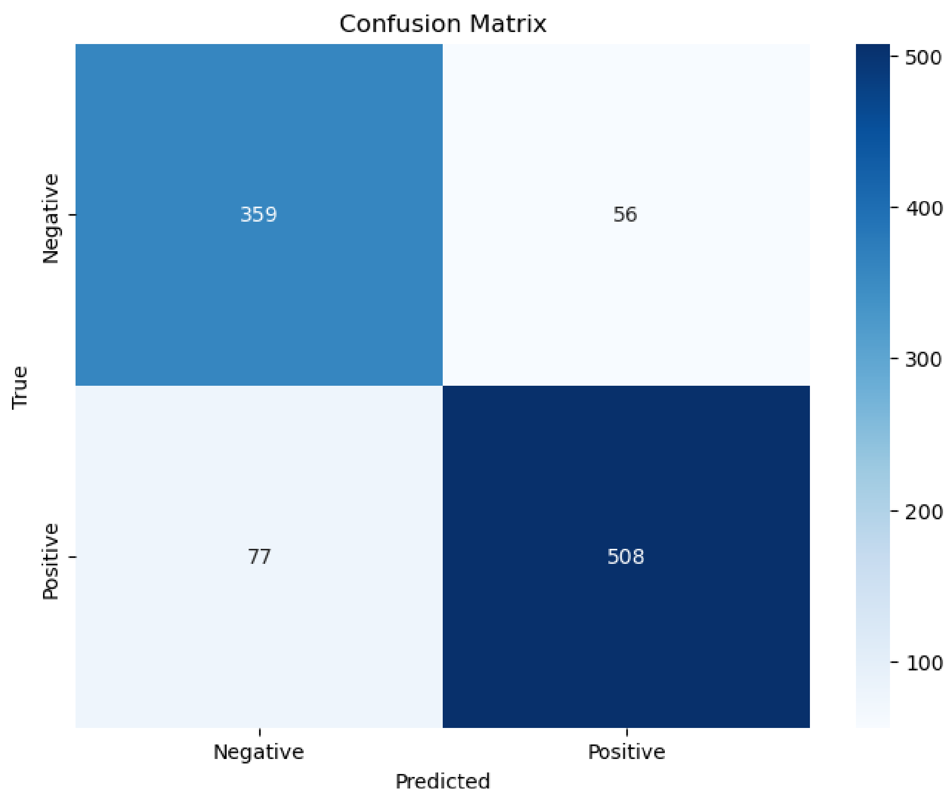


Fig 9: Confusion matrix for classification

In this confusion matrix, where negative represents rejection and positive signifies acceptance, the model correctly classified 359 instances where loan applications were rejected (True Negatives). However, it erroneously classified 56 cases of rejected applications as accepted (False Positives), potentially leading to inappropriate approvals. Additionally, the model failed to identify 77 instances where loan applications were accepted but predicted them as rejections (False Negatives), indicating missed opportunities for loan approvals. On a positive note, the model accurately predicted acceptance for 508 loan applications (True Positives), ensuring that deserving applicants were rightfully approved.

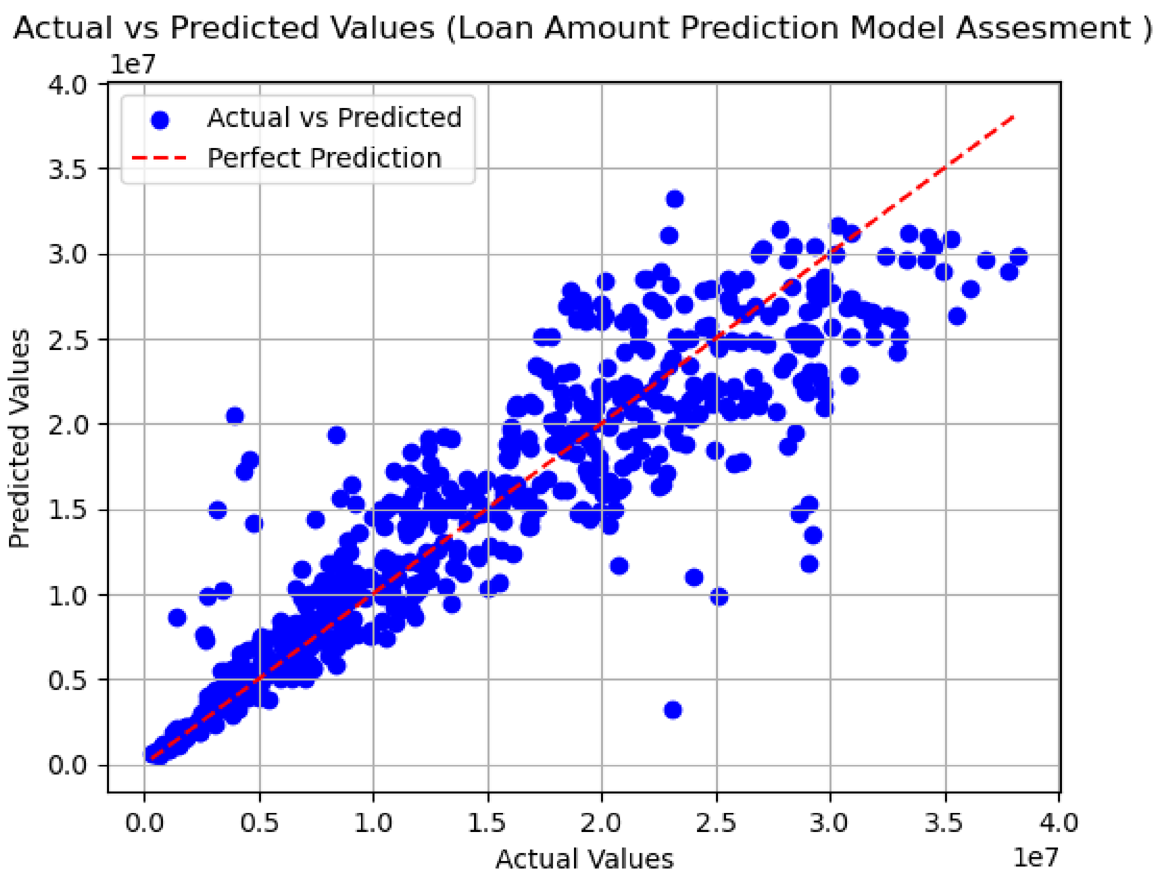


Figure 10: Actual vs perfect prediction graph for loan amount

This graph represents the relationship between the actual values and the predicted values generated by a model. Each point on the graph corresponds to an observation in the dataset, where the x-axis represents the actual values (ground truth) and the y-axis represents the predicted values generated by the model. And the red line represents the perfect predictions.

Overall, result analysis serves as a crucial step in evaluating the effectiveness of the loan eligibility prediction system and guiding future enhancements.

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

The Loan Eligibility Prediction Project is an ideal implementation of one of the most popular algorithms i.e. random forest algorithm. Through this project we demonstrate the classification and regression capabilities of random forest algorithm in identifying and predicting loan eligibility and amount. Developed using the Python programming language and leveraging the Random Forest algorithm as well as utilizing Html/CSS/JavaScript and Gradio for our frontend, our project aims to accurately predict whether an individual is eligible for a loan based on various features such as income, credit score, and employment status. By training the Random Forest algorithm on historical loan data, we enable it to make informed decisions regarding loan approval. We facilitate the input of data from our frontend which in turn utilizes the model and shows us the result in a systematic and clear manner. To improve the accuracy of our results we have made the size number of decision tree to 19 having depth 15.

This project showcases the effectiveness and versatility of the Random Forest algorithm in handling classification and regression tasks in identifying candidates suitable for loans and the amount of loans they can get, making it a valuable tool for financial institutions and loan applicants alike. The project was completed by studying what user details are the most impactful towards the person's loan taking capability and what dictates the person capacity. By engaging with the application, users can obtain information about their loan eligibility without needing to be physically present in a bank or interacting with other people as well as provide a clear information regarding how much loan they are capable of obtaining to plan their goals and takes.

6.2 Future Enhancement

Rejection Reason Explanation: When an applicant is rejected for a loan, the model can provide a clear explanation of the reason(s) behind the rejection. These reasons could include factors such as low credit score, insufficient income, providing specific feedback helps applicants understand why their application was denied and what areas they need to address to improve their chances of approval in the future.

Credit Score Improvement Guidance: In addition to explaining the rejection reasons, the model can offer personalized guidance on how the applicant can improve their credit score. This guidance could include suggestions such as paying bills on time, reducing credit card balances, disputing errors on credit reports, or diversifying credit mix.

Loan Type Selection: Applicants can choose the type of loan they want, like a business loan, home loan, education loan, or personal loan. Each type has its own requirements, such as how much money you make or your credit history. This helps make sure you get the right loan for what you need.

APPENDIX I

The code for loan_system.py

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score, r2_score
from random_forest import RandomForest
import joblib
loan=pd.read_csv('./data/loan.csv')
loan = loan.rename(columns={
    ' no_of_dependents':'no_of_dependents',
    ' education':'education',
    ' self_employed': 'self_employed',
    ' income_annum':'income_annum',
    ' loan_amount':'loan_amount',
    ' loan_term':'loan_term',
    ' cibil_score': 'cibil_score',
    ' residential_assets_value': 'residential_assets_value',
    ' commercial_assets_value':'commercial_assets_value',
    ' luxury_assets_value':'luxury_assets_value',
    ' bank_asset_value':'bank_asset_value',
    ' loan_status':'loan_status'
})
col=['no_of_dependents','income_annum','loan_amount','loan_term','cibil_score','residential_a
ssets_value','commercial_assets_value','luxury_assets_value','bank_asset_value']
for co in col:
    loan[co].plot.box(vert=False)
    plt.figure(figsize=(5, 10))
    plt.tight_layout()
    plt.show()
loan.isnull().sum()
loan.select_dtypes(include='number')
col=['no_of_dependents','income_annum','loan_amount','loan_term','cibil_score','residential_a
ssets_value','commercial_assets_value','luxury_assets_value','bank_asset_value']
for i in range(1):
    for co in col:
        Q1 = loan[co].quantile(0.25)
        Q3 = loan[co].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        loan = loan[(loan[co] >= lower_bound) & (loan[co] <= upper_bound)]
for co in col:
    loan[co].plot.box(vert=False)
    plt.figure(figsize=(5, 10))
    plt.tight_layout()
    plt.show()
loan.shape
loan['education'] = loan['education'].map({' Graduate': 1, ' Not Graduate': 0,'Graduate':1,'Not
Graduate':0})
loan['self_employed'] = loan['self_employed'].map({' Yes': 1, ' No': 0,'Yes': 1,'No': 0})
loan['loan_status'] = loan['loan_status'].map({' Approved': 1, ' Rejected': 0,'Approved': 1,
'Rejected': 0})
np.random.seed(42)
random_indices = np.random.permutation(loan.index)
loan = loan.loc[random_indices]
for column in loan.columns:
    loan[column].plot(kind='density', figsize=(6, 4))
    plt.title(f'Distribution of {column}')
    plt.show()
X = loan.drop(['loan_status', 'loan_amount', 'loan_id'], axis=1).values
y = loan['loan_status'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(np.count_nonzero(y_train== 0))
rf_approval = RandomForest(n_trees=19,max_depth=15,random_state=42)
rf_approval.fit(X_train, y_train)
y_pred= rf_approval.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.5f}")
import seaborn as sns
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)  #
```

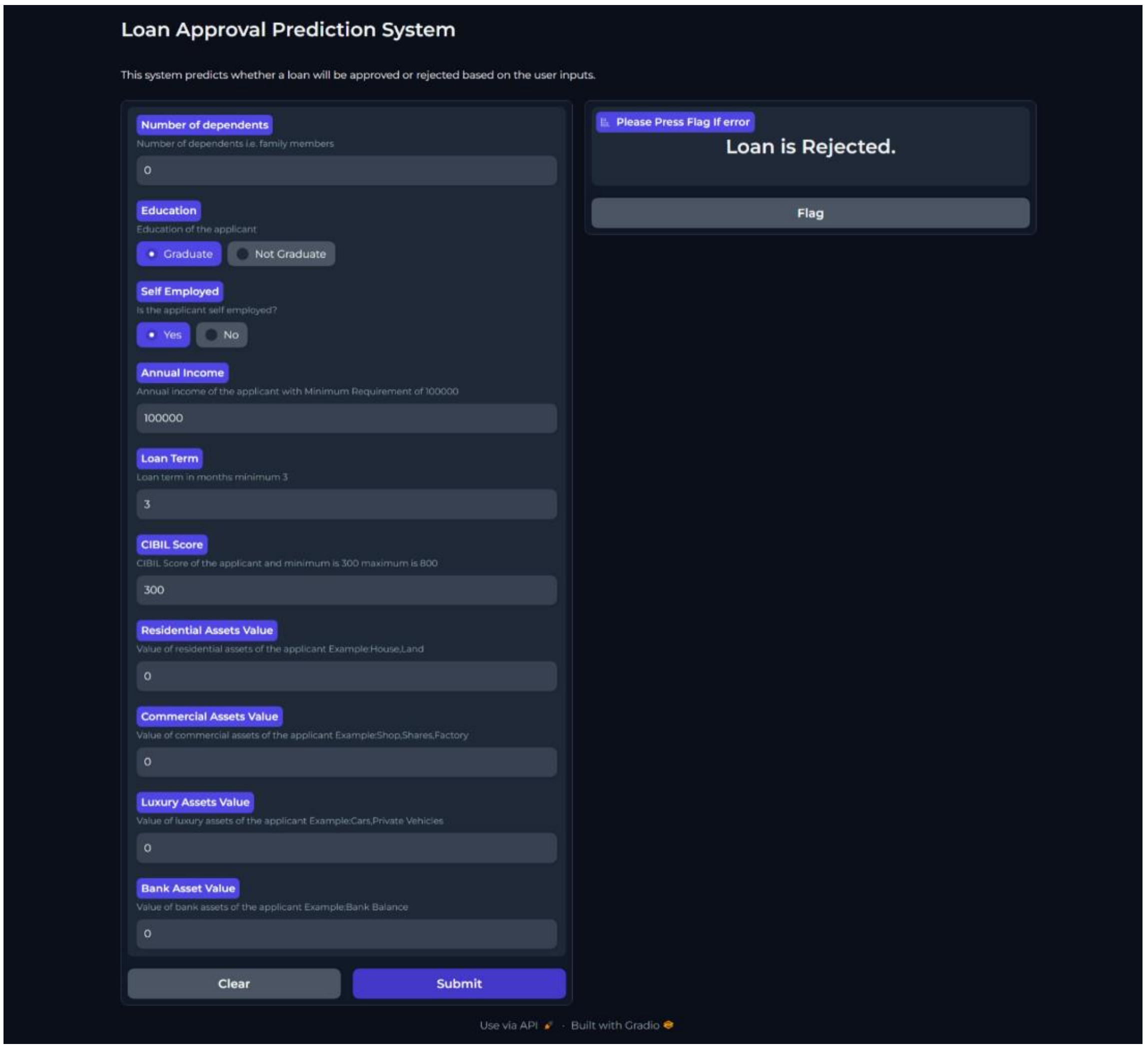
```
labels = ['Negative', 'Positive']
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
joblib.dump(rf_approval, "models/approval.joblib")
amount=loan[loan['loan_status']==1]
amount.head()
amount = amount.drop(['loan_status','loan_id'], axis=1)
coll=['no_of_dependents','income_annum','loan_amount','loan_term','cibil_score','residential_
assets_value','commercial_assets_value','luxury_assets_value',
'bank_asset_value']
A = amount.drop('loan_amount', axis=1)
b = amount['loan_amount']
A_train, A_test, b_train, b_test = train_test_split(A, b, test_size=0.2, random_state=42)
rf_amount = RandomForestRegressor(random_state=42)
rf_amount.fit(A_train, b_train)
predictions = rf_amount.predict(A_test)
r2 = r2_score(b_test, predictions)
print(r2)
plt.scatter(b_test, predictions, color='blue', label='Actual vs Predicted')
plt.plot([min(b_test), max(b_test)], [min(b_test), max(b_test)], color='red', linestyle='--',
label='Perfect Prediction')
plt.title('Actual vs Predicted Values (Loan Amount Prediction Model Assesment )')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.grid(True)
plt.show()
```

The code for random_forest.py

```
import numpy as np
class DecisionTree:
    def __init__(self, max_depth=None):
        self.max_depth = max_depth
        self.tree = None
    def fit(self, X, y, depth=0):
        unique_classes, counts = np.unique(y, return_counts=True)
        if len(unique_classes) == 1 or depth == self.max_depth:
            return {'value': unique_classes[0]}
        feature_index, threshold = self.get_best_split(X, y)
        left_indices = X[:, feature_index] <= threshold
        right_indices = ~left_indices
        left_subtree = self.fit(X[left_indices], y[left_indices], depth + 1)
        right_subtree = self.fit(X[right_indices], y[right_indices], depth + 1)
        return {'feature_index': feature_index, 'threshold': threshold,
              'left': left_subtree, 'right': right_subtree
   def get_best_split(self, X, y):
        best_info_gain = -1
        best_feature, best_threshold = None, None
        for feature_index in range(X.shape[1]):
            unique_values = np.unique(X[:, feature_index])
            for threshold in unique_values:
                info_gain = self.calculate_information_gain(X, y, feature_index, threshold)
                if info_gain > best_info_gain:
                    best_info_gain = info_gain
                    best_feature = feature_index
                    best_threshold = threshold
        return best_feature, best_threshold
    def calculate_information_gain(self, X, y, feature_index, threshold):
        left_indices = X[:, feature_index] <= threshold
        right_indices = ~left_indices
        left_entropy = self.calculate_entropy(y[left_indices])
        right_entropy = self.calculate_entropy(y[right_indices])
        parent_entropy = self.calculate_entropy(y)
        weight_left = len(y[left_indices]) / len(y)
        weight_right = len(y[right_indices]) / len(y)
        information_gain = parent_entropy - (weight_left * left_entropy + weight_right *
right_entropy)
        return information_gain
    def calculate_entropy(self, y):
        _, counts = np.unique(y, return_counts=True)
        probabilities = counts / len(y)
        entropy = -np.sum(probabilities * np.log2(probabilities + 1e-10))
        return entropy
    def predict_single(self, node, sample):
        if 'value' in node:
            return node['value']
        if sample[node['feature_index']] <= node['threshold']:
            return self.predict_single(node['left'], sample)
        else:
            return self.predict_single(node['right'], sample)
    def predict(self, X):
        return [self.predict_single(self.tree, sample) for sample in X]
class RandomForest:
    def __init__(self, n_trees=100, max_depth=None,random_state=None):
        self.n_trees = n_trees
        self.max_depth = max_depth
        self.random_state=random_state
        self.trees = []
    def fit(self, X, y):
        for _ in range(self.n_trees):
            tree = DecisionTree(max_depth=self.max_depth)
            tree.tree = tree.fit(X, y)
            self.trees.append(tree)
    def predict(self, X):
        predictions = np.array([tree.predict(X) for tree in self.trees])
        return np.round(np.mean(predictions, axis=0))
    def predict_proba(self, X):
        predictions = np.array([tree.predict(X) for tree in self.trees])
        return np.mean(predictions, axis=0)
```

APPENDIX II

On providing minimum values for input parameters



Loan Approval Prediction System

This system predicts whether a loan will be approved or rejected based on the user inputs.

Number of dependents
Number of dependents i.e. family members
0

Education
Education of the applicant
[✓ Graduate] Not Graduate

Self Employed
Is the applicant self employed?
[✓ Yes] No

Annual Income
Annual income of the applicant with Minimum Requirement of 100000
100000

Loan Term
Loan term in months minimum 3
3

CIBIL Score
CIBIL Score of the applicant and minimum is 300 maximum is 800
300

Residential Assets Value
Value of residential assets of the applicant Example:House,Land
0

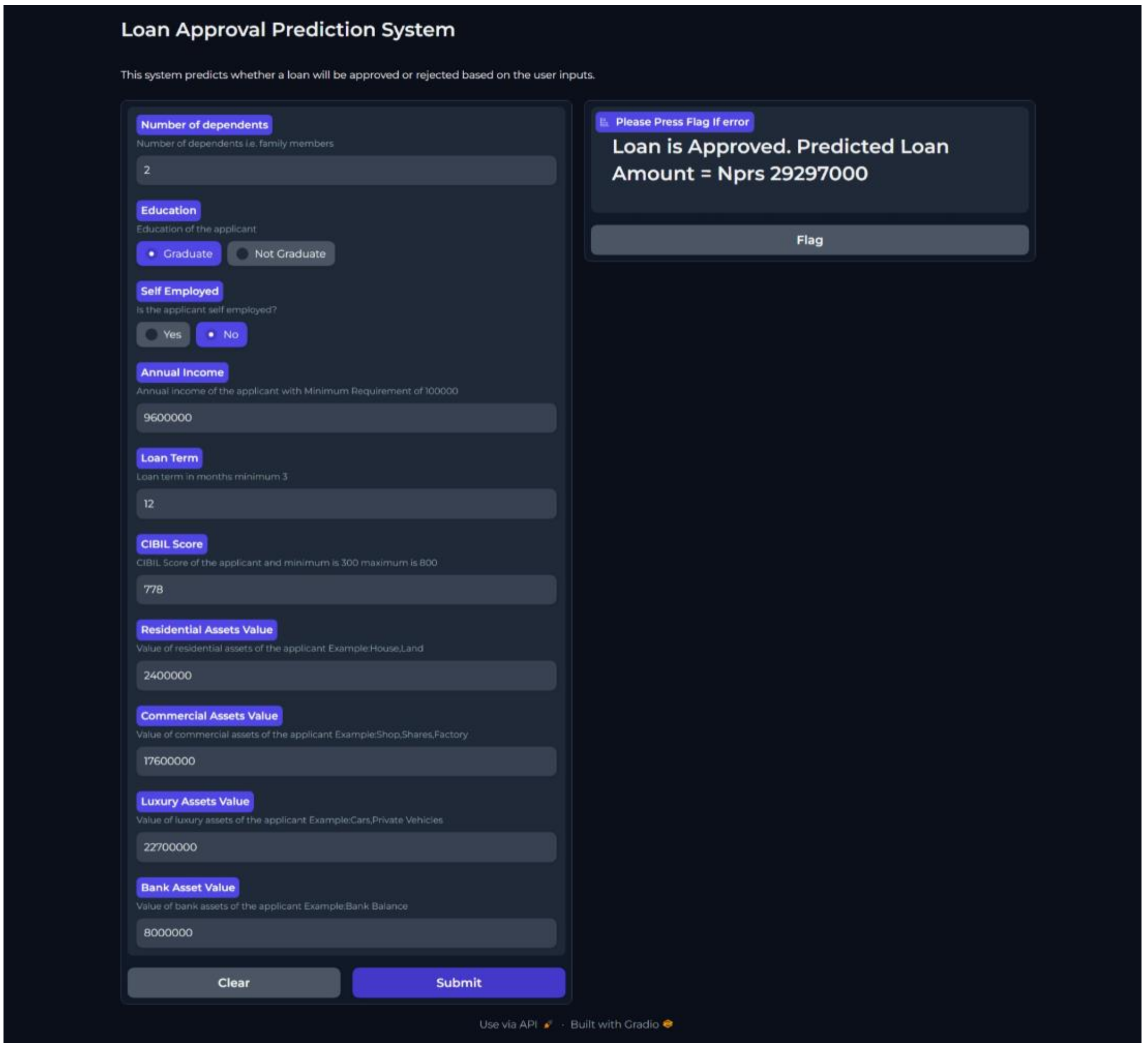
Commercial Assets Value
Value of commercial assets of the applicant Example:Shop,Shares,Factory
0

Luxury Assets Value
Value of luxury assets of the applicant Example:Cars,Private Vehicles
0

Bank Asset Value
Value of bank assets of the applicant Example:Bank Balance
0

Please Press Flag If error
Loan is Rejected.

Loan gets approved and system predicts the loan amount.



Loan Approval Prediction System

This system predicts whether a loan will be approved or rejected based on the user inputs.

Number of dependents
Number of dependents i.e. family members
2

Education
Education of the applicant
[✓ Graduate] Not Graduate

Self Employed
Is the applicant self employed?
Yes [✓ No]

Annual Income
Annual income of the applicant with Minimum Requirement of 100000
9600000

Loan Term
Loan term in months minimum 3
12

CIBIL Score
CIBIL Score of the applicant and minimum is 300 maximum is 800
778

Residential Assets Value
Value of residential assets of the applicant Example:House,Land
2400000

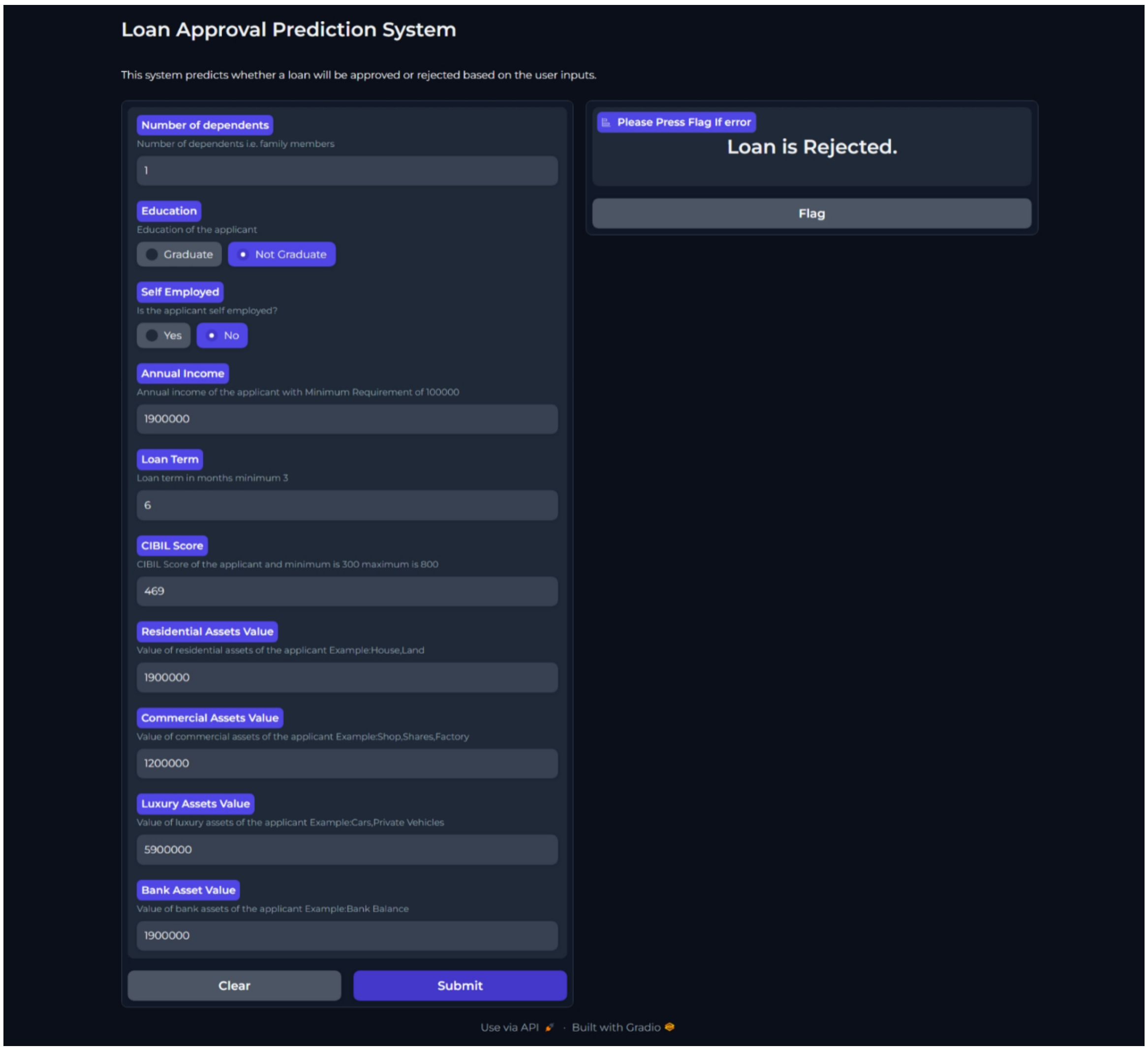
Commercial Assets Value
Value of commercial assets of the applicant Example:Shop,Shares,Factory
17600000

Luxury Assets Value
Value of luxury assets of the applicant Example:Cars,Private Vehicles
22700000

Bank Asset Value
Value of bank assets of the applicant Example:Bank Balance
8000000

Please Press Flag If error
Loan is Approved. Predicted Loan Amount = Nprs 29297000

Loan gets rejected



Loan Approval Prediction System

This system predicts whether a loan will be approved or rejected based on the user inputs.

Number of dependents
Number of dependents i.e. family members
1

Education
Education of the applicant
Graduate [✓ Not Graduate]

Self Employed
Is the applicant self employed?
Yes [✓ No]

Annual Income
Annual income of the applicant with Minimum Requirement of 100000
1900000

Loan Term
Loan term in months minimum 3
6

CIBIL Score
CIBIL Score of the applicant and minimum is 300 maximum is 800
469

Residential Assets Value
Value of residential assets of the applicant Example:House,Land
1900000

Commercial Assets Value
Value of commercial assets of the applicant Example:Shop,Shares,Factory
1200000

Luxury Assets Value
Value of luxury assets of the applicant Example:Cars,Private Vehicles
5900000

Bank Asset Value
Value of bank assets of the applicant Example:Bank Balance
1900000

Please Press Flag If error
Loan is Rejected.

REFERENCES

[1] J. Xue, "Financial Risk Prediction and Evaluation Model of P2P Network Loan Platform," 2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Phuket, Thailand, 2020, pp. 1060-1064, Doi: 10.1109/ICMTMA50254.2020.00227.

[2] L. Zhu, D. Qiu, D. Ergu, C. Ying, and K. Liu, “A study on predicting loan default based on the random forest algorithm,” Procedia Computer Science, vol. 162, pp. 503–513,2019, Doi: https://doi.org/10.1016/j.procs.2019.12.017.

[3] L. Zhou and H. Wang, “Loan Default Prediction on Large Imbalanced Data Using Random Forests,” International Research Journal of Engineering and Technology (IRJET), vol. 10, no. 6, pp. 1519–1525.

[4] K. Gautam, A. P. Singh, K. Tyagi, and S. Kumar, “Loan Prediction using Decision Tree and Random Forest,” International Research Journal of Engineering and Technology (IRJET), vol. 07, no. 08, pp. 853–856, Aug. 2020, Available: https://www.irjet.net/