Tribhuvan University
Amrit Campus

Final Year Project Report

on

"LOAN APPROVAL PREDICTION SYSTEM"

[CSC 412]

Under the supervision of

Asst. Prof. Dhirendra Kumar Yadav
Department of Computer Science and Information Technology
Amrit Campus

Submitted by

Abishek Gautam (23114/076)
Nisha Subedi (23168/076)
Raichung Pun Magar (23179/076)

Submitted to

Department of Computer Science and Information Technology
Amrit Campus
Institute of Science and Technology
Tribhuvan University

April 1, 2024

Tribhuvan University
Amrit Campus

Final Year Project Report

on

LOAN APPROVAL PREDICTION SYSTEM

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of **Bachelor of Science in Computer Science and Information Technology** awarded by **Tribhuvan University**

Submitted by

Abishek Gautam (23114/076)
Nisha Subedi (23168/076)
Raichung Pun Magar (23179/076)

Submitted to

Department of Computer Science and Information Technology
Amrit Campus
Institute of Science and Technology
Tribhuvan University

April 1, 2024

Tribhuvan University

Institute of Science and Technology

Amrit Campus

Department of Computer Science and Information Technology

Email: csitascol@gmail.com

Recommendation Letter of Supervisor

I hereby recommend that the report prepared under my supervision by Abishek Gautam (23114/076), Nisha Subedi (23168/076) and Raichung Pun Magar (23179/076) entitled **"LOAN APPROVAL PREDICTION SYSTEM"** be accepted as fulfilling in partial requirement for the degree of Bachelors of Science in Computer Science and Information Technology. To the best of my knowledge, this is an original work in Computer Science and Information Technology.

……..…….

Mr. Dhirendra Kumar Yadhav

Supervisor

Department of CSIT

Amrit Campus

Lainchaur, Kathmandu

Tribhuvan University

Institute of Science and Technology

Amrit Campus

Department of Computer Science and Information Technology

Email: csitascol@gmail.com

CERTIFICATE OF APPROVAL

This is to certify that this project prepared by Abishek Gautam (23114/076), Nisha Subedi (23168/076) and Raichung Pun Magar (23179/076) entitled **"LOAN APPROVAL PREDICTION SYSTEM"** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

Asst. Prof. Dhirendra Kumar Yadhav
Project Coordinator,
Department of Computer Science and IT
Amrit Campus

External Examiner
Central Department of Computer Science and IT
Tribhuvan University
Kirtipur, Nepal

ACKNOWLEDGEMENT

The team would like to honor and express our deepest appreciation to everyone who assisted us. We would like to extend our sincerest thanks and appreciation to Asst. Prof. Dhirendra Kumar Yadav, our supervisor of the project. His valuable expertise in the field provided guidance and inspiration to help us with the project. His insights and suggestions have helped the team to overcome many obstacles and challenges during the development phase.

We would like to convey our gratitude towards the Department of Computer Science and Information Technology, Amrit Science Campus, for providing a wonderful opportunity to complete this project and commitment from teachers for providing a proper environment for facing challenges during the project.

We would extend our heartfelt gratitude to all faculty members and well-wishers who provided efforts directly or indirectly in the project. We'd like to express our gratitude to our friends and colleagues for their tireless efforts in completing the project.

Sincerely,

Abishek Gautam (23114/076)

Nisha Subedi (23168/076)

Raichung Pun Magar (23179/076)

ABSTRACT

Loan lending process is a critical aspect of the financial sector and applicants. The complexity of the risk assessment process is very time consuming and takes enormous use of resources. There has been much research conducted for creating ease of this task. This project uses decision trees and random forest algorithms for classifying loan approvals. Entropy and Information gain are taken as impurity metrics of the project for splitting data. Different experiments like: Under sampling, over sampling, principal component analysis and hyperparameter tuning are conducted for selecting the best set of hyperparameters. The model provided 84.2% balanced accuracy as the performance of the system is robust with 32000 loan applications datasets. The system has potential to help financial institutions streamline their loan approval process, reduce risks and time and make informed decisions to applicants.

Keywords: ***Loan approval prediction, Decision Tree, Random Forest, Voting Classifier, Entropy, Information Gain, Hyperparameter tuning***

TABLE OF CONTENTS

ACKNOWLEDGEMENT i
ABSTRACT ii
LIST OF FIGURES v
LIST OF TABLES vi
LIST OF ABBREVIATIONS vii
CHAPTER 1 : INTRODUCTION 1
1.1 Introduction 1
1.2 Problem Statement 1
1.3 Objectives 2
1.4 Scope and Limitations 2
1.5 Development Methodology 2
CHAPTER 2 : BACKGROUND STUDY AND LITERATURE REVIEW 4
2.1 Literature Review 4
CHAPTER 3 : SYSTEM ANALYSIS 6
3.1 System Analysis 6
3.1.1 Requirement Analysis 6
3.1.2 Feasibility Analysis 10
3.1.3 Analysis 11
CHAPTER 4 : SYSTEM DESIGN 13
4.1 Design 13
4.1.1 Class Diagram 13
4.1.2 Activity Diagram 14
4.1.3 Sequence Diagram 15
4.1.4 System Architecture 16
4.2 Algorithm Details 17
4.2.1 Decision Node 18
4.2.2 Impurity Metrics 18
4.2.3 ID3 algorithm 20
4.2.4 Ensemble Learning 21
4.2.5 Bagging 22
4.2.6 Voting classifier 22
4.2.7 Random Forest Algorithm 22
CHAPTER 5 : IMPLEMENTATION AND TESTING 25
5.1 Implementation 25
5.1.1 Tools Used 25
5.1.2 Implementation Details 30
5.1.3 Experiments Conducted 37
5.2 Testing 40
5.2.1 Test Cases for Unit Testing 40

5.2.2 Test Case for Integration Testing 50
5.2.3 Test Cases for System Testing 51
5.3 Result Analysis 53
CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATIONS 58
6.1 Conclusion 58
6.2 Future Recommendations 58
REFERENCES 59
APPENDIX 60

LIST OF FIGURES

Figure 1.1 Incremental Delivery Model 3

Figure 3.1 Use case diagram of loan approval system 8

Figure 3.2 Gantt Chart of the project 13

Figure 3.3 ER diagram of the project 14

Figure 4.1 Descriptive class diagram of the project 17

Figure 4.2 Activity Diagram for Loan Request Process by Applicant 18

Figure 4.3 Sequence diagram of View Loan Prediction Process 19

Figure 4.4 System Architecture 20

Figure 4.5 Decision Tree 21

Figure 4.6 Plot between Entropy and Probability of class. 23

Figure 4.7 Random Forest Classifier 27

Figure 5.1 Algorithm Implementation Procedure 33

Figure 5.2 Model pipeline of the system 39

Figure 5.3 ROC Curve 62

LIST OF TABLES

Table 3.1 Use Case Description for Applicant Registration 9
Table 3.2 Use Case Description for Applicant Login 9
Table 3.3 Use Case Description for Request Loan 10
Table 3.4 Use Case Description for Provide Loan Data 10
Table 3.5 Use Case Description for View Loan Prediction 11
Table 3.6 Work Breakdown Structure 12
Table 5.1 Built-in python modules 31
Table 5.2 External Modules used in project 32
Table 5.3 Hardware configuration used in project 32
Table 5.4 Data columns 34
Table 5.5 Dataset split size 36
Table 5.6 One-hot encoding 37
Table 5.7 Label Encoding 38
Table 5.8 Baseline model metrics for training and test dataset 38
Table 5.9 Data Size for random data under sampling 40
Table 5.10 Data Size for random data oversampling 41
Table 5.11 Hyperparameters for Decision Tree 42
Table 5.12 Hyperparameters for Random Forest 42
Table 5.13 Test Case for Applicant Register 43
Table 5.14 Test Case for Applicant Login 46
Table 5.15 Test Case for Apply Loans 48
Table 5.16 Test Case for Employee Registration 49
Table 5.17 Test Case for provide loan data. 53
Table 5.18 Test Case for loan prediction 54
Table 5.19 Test set results with different models 63

LIST OF ABBREVIATIONS

ADAM	Adaptive Moment Estimation
ALR	Average lending rate
AUC	Area Under the Curve
BP	Back propagation
CART	Classification and regression tree
CASE	Computer Aided Software Engineering tools
CER	Cost-efficiency ratio
CSMAR	China Stock Market & Accounting Research
CSS	Cascading Style Sheet
ER	Entity Relation Diagram
FP	Financial performance
HTML	Hypertext Markup Language
ID3	Iterative Dichotomiser 3
IDE	Integrated Development Environment
JS	Javascript
KNN	K-nearest neighbors

LR	Liquidity ratio
MVC	Model View Controller
MVT	Model View Template
NN	Neural Network
NPL	Non-performing loan
PCA	Principal component analysis
PL	Performing loan
RDBMS	Relational Database Management System
SME	Small and medium sized enterprise
SQL	Structured Query Language
UML	Unified modeling language
W3C	World Wide Web Consortium

CHAPTER 1 : INTRODUCTION

1.1 Introduction

Loan approval is the process of authorizing the loan based on the applicant's nature. Credit risk management system is the system which looks after the probability of loss due to the applicant's failure to make payment on any type of debt. It is the practice of mitigating the risk of any debt or credit failures.

Loan approval prediction system is the credit risk management system where applicants can apply for loans and bank employees provide past records of applicants for classifying risks associated with the loan. This system is a web-based application and uses tree-based machine learning techniques for classifying the loan request. There are two different types of users associated with the system, namely: applicant and bank employee. Applicant applies for the loan and employee provides different banking details associated with applicant and system provides approval status of the applicant's loan.

1.2 Problem Statement

In recent years, financial Institutions have had significant challenges over managing the lent loan. There has been a huge growth of credit risks associated with those loans. There is a problem of manual processing loans for approval. This manual process may be time consuming, prone to human errors and lacks the ability to perform real time risk analysis. Automated systems are not easily scalable for handling large amounts of the data for this purpose.

Manual processes of looking after credit risks are very time consuming which may result in loss of customers/applicants to the banks. Also in the current scenario, there is a very hard procedure for facilitating effective risk management through the automated system. Machine learning technologies are less used for these automation related tasks which may be beneficial for the financial institutions.

Machine learning systems and solutions are much harder to optimize based on the user requirements such that financial institutions may not take the risks related to the credit risk management through automation approach. Due to this, there are huge applications of automations towards credit risk management sectors like: financial institutions.

1.3 Objectives

The primary objectives of the project are:

- To implement tree-based algorithms (decision tree, random forest) from scratch and use different optimization techniques for improving the model performance.
- To predict loans with the help of machine learning.

1.4 Scope and Limitations

In this project, different data like: employment length, credit history, credit default, etc. are taken as inputs to the machine learning model from employee and applicant and loan approval status is predicted based on the provided data.

Following are the limitations of the project:

- The system will not be able to conduct banking and financial services like: withdraw, balance check.
- This system will not be able to have loan dispatch and contract management related services.

1.5 Development Methodology

Incremental development model is used for developing this project. Incremental development model has provision of providing immediate feedback to the system, as a result it is suitable for this project. Phases of Incremental development model includes: Requirement analysis, Design and development, Implementation and testing.

Here are basic principles of incremental development model:

- Initially outline description is provided for the development. When developing outline is changed into specification through the proper use of requirement analysis.
- Requirements are properly used for developing and validating the system through model deployment and feedback at each iteration.
- Each new version will be improved based on feedback from the previous version.

Here is the diagram for the phases of incremental development model:

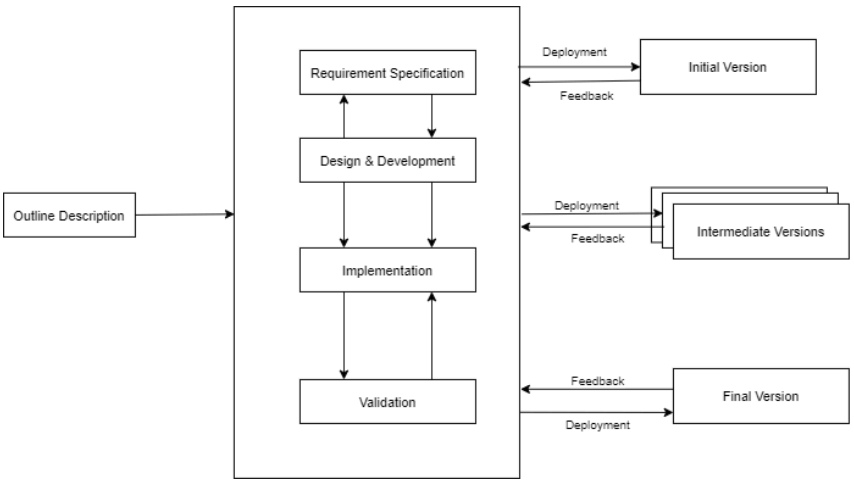


Figure 1.1 Incremental Delivery Model

CHAPTER 2 : BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Literature Review

Credit risk management is taken as huge importance for the financial institutions. Different tasks like: checking a credit score, previous credit history, collateral availability and valuation of the project is done in complex manner. This complex task can be eased through the help of machine learning using different algorithms for this domain area.

Recent research papers that have been published within specific domain of financial sectors and machine learning as well were properly studied for gaining domain knowledge to the project. These papers have given us proper insight over the loan approval system with their inner workings and problems.

Paper by Bansode have provided that primary source of benefit for financial institutions are through the interest of loan. Authors have also provided that loan recovery is a significant contributor to a bank's financial statements. There is extremely difficulty in predicting whether the customer will be able to pay back the loan [1]. They have used loan id, gender, marital status, dependents, employment status, loan amount term, credit history and property area for looking after loan prediction. They fed that information to logistic regression, support vector machine and k-nearest neighbor's classifiers and the accuracy of the algorithm were: 84.3, 80.2 and 76.7 percentage respectively.

Another study by Yacui Gao in 2018 has provided detailed research in credit risk assessment on SMEs in commercial banks [2]. Credit risk assessment system is referred to as the application of evaluation technology in commercial banks and other financial institution to quantitatively calculate the factors that may cause the risk of loan, which is to judge the borrower's risk of default or the possibility of repayment. Authors have built a credit risk assessment evaluation index system for comparing overall risks of the SMEs through the use of probabilistic algorithm like Bayesian Logistic Regression. Twenty-nine different features like: current ratio of assets and liabilities, cash ratio, current asset turnover, roe, industry position, etc. were used for evaluating SMEs. They have used KS test, Mann-whitney test, independent sample-t test was conducted for inspection of the indicator variable. For reducing the number of features PCA were conducted. Authors used a Bayesian logistic

regression model for properly analyzing the data, which resulted in accuracy of 93.4 percent in the training sample and 90 percent in the testing sample.

In the paper "*The effect of credit risk management and bank-specific factors on the financial performance of the south Asian banks*", authors have looked after PL and NPLs. Authors have captured the effect of credit risk management and bank specific factors on south Asian financial institutions like CER, ALR and LR were used for finding out the impact over the FP [3]. Autocorrelation tests, endogeneity tests, and ordinary least square methods were used for comparing the credit risks.

Another paper "*Research on credit risk evaluation of commercial banks based on artificial neural network*" used artificial neural network for looking out credit risks. In this paper, credit risks are given as major risks faced by commercial banks [4]. Credit return is largely determined by whether the borrower can repay the credit loan on schedule. Three different neural network architectures, namely: BP NN, Radial basis function network and perceptron network were used for loan approval prediction. Fourteen different financial indicators were used as the features of the dataset. CSMAR database which stored loan related activities of top 150 companies listed in the manufacturing industries were used for model training. Author also used clustering analysis for determining actual credit ratings of each company. BP NN had 92.5% training accuracy, for radial basis function network performed with training of 97.7% and test accuracy of 91%. Perceptron network had training accuracy of 98.8% whereas test accuracy of 95.8%.

Paper from P. Tamayo has provided loan approval assessment through the use of statistical approach. Authors used data from the 1980's and 1990's financial crisis and contribution of loan in the crisis. They also used statistical probit regression model, neural networks, KNN and CART model for this analysis. CART with 120 nodes provided 8.3% test error, neural network provided 11%, KNN provided 14.95% and probit algorithm provided 15.13 percent test error. Authors have provided risks like: mortgages, credit cards and other prepayment frauds for the credit assessment [5]. Through analysis of these risks, we can properly extend the system and create more accurate model.

CHAPTER 3 : SYSTEM ANALYSIS

3.1 System Analysis

System analysis is the process of studying and understanding a system's components, interactions, and functions to identify its requirements, constraints, and opportunities for improvement.

3.1.1 Requirement Analysis

Functional requirements describe the specific behavior or functions of a system, software, or product. They define what the system should do to fulfill the needs of its users.

3.1.1.1 Functional Requirements

Functional requirements are statements which system's functionality, reaction of system with particular inputs and behavior of the system within a particular situation.

Following are the functional requirements of the project:

- Allow applicants to request a loan.
- Allow employees to provide loan details.
- Allow both applicant and employee to view the classified loan status.
- Allow applicants to register to the system.

The following is the use case diagram of the functionality of the system and interaction between actors (employee and applicant):

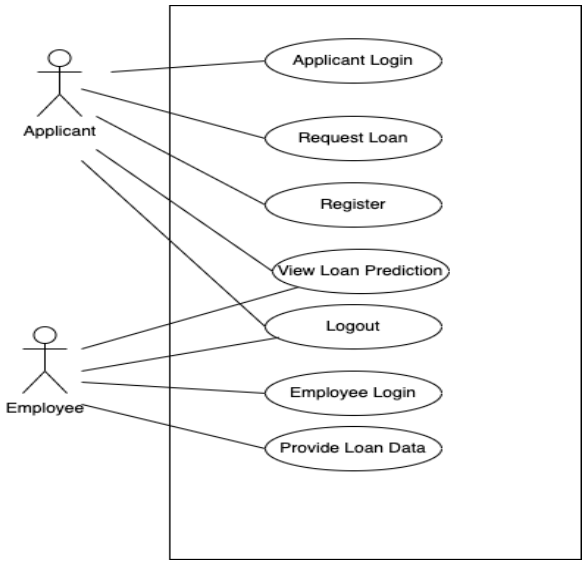


Figure 3.1 Use case diagram of loan approval system

Use Case Description

Here are some of the use case descriptions of the project:

Table 3.1 Use Case Description for Applicant Registration

Use Case ID	UC-01
Use Case Name	Applicant Registration
Primary Actor	Applicant
Secondary Actor	
Description	Registers employee to the system
Pre-condition	
Success Scenario	Applicant can login to the system. Applicant data is stored in the system.
Failure Scenario	Applicant is redirected to sign up again.

Table 3.2 Use Case Description for Applicant Login

Use Case ID	UC-02
Use Case Name	Applicant Login
Primary Actor	Applicant
Secondary Actor	
Description	Logs in applicant to the system
Pre-condition	Applicant must be registered to the system.
Success Scenario	Applicant is redirected to home page.
Failure Scenario	Error message is passed to the applicant.

Table 3.3 Use Case Description for Request Loan

Use Case ID	UC-03
Use Case Name	Request Loan
Primary Actor	Applicant
Secondary Actor	
Description	Applicant provides different loan related data to the system and requests for the loan.
Pre-condition	Applicant must be registered and logged in to the system.
Success Scenario	Loan request saved in to the database. Loan request is now forwarded to the employee.
Failure Scenario	Loan request is not saved in the database.

Table 3.4 Use Case Description for Provide Loan Data

Use Case ID	UC-04
Use Case Name	Provide Loan Data
Primary Actor	Employee
Secondary Actor	
Description	Employee provides credit history and loan related details to the system for further prediction.
Pre-condition	Employee must be logged in to the system.
Success Scenario	Loan details is saved in to the database.
Failure Scenario	Loan details is not saved in the database.

Table 3.5 Use Case Description for View Loan Prediction

Use Case ID	UC-05
Use Case Name	View Loan Prediction

Primary Actor	Employee or Applicant
Secondary Actor	
Description	System predicts whether the applicant is eligible for the loan.
Pre-condition	Loan requests and Loan details must be provided to the system.
Success Scenario	Loan prediction status is changed. Loan status is saved to the database.
Failure Scenario	Loan status is not saved to the database.

3.1.1.2 Non-functional Requirements

Non-functional requirements are the functions which are offered by the system which may fully impact the system rather than the individual system components. Non-functional requirements help in addressing overall system properties.

Here are some of the non-functional requirements of the project:

Security: Here in this system, users can properly register but users not registered to the system cannot log in to the system. Employees and applicants have different privilege levels to the system.

Usability: This system has a simple, responsive and user-friendly interface. Also, architecture and design used in the system is simple such that other systems can use this system easily.

Maintainability: The system is maintainable in nature. We will use an object-oriented approach for developing this system such that it can easily be changed if necessary.

3.1.2 Feasibility Analysis

Feasibility study is the focused study that takes place early in the requirement engineering process. The aim of the feasibility study is to find out whether the system can be implemented or not. There are following feasibility study done in the project:

3.1.2.1 Schedule Feasibility

Schedule feasibility looks after the potential time frame of the completion of the project. It also looks after major activities and their time period or constraints involved. Through thorough analysis, this project is feasible in schedule and can be completed in the time frame.

Here is the work breakdown structure of the project:

Table 3.6 Work Breakdown Structure

	A	B	C	D	E	F
1	ID	Task Name	Duration	Start	Finish	Predecessors
2	1	Project Start	0 days	Tue 12/5/23	Tue 12/5/23	
3	2	**Project Planning**	**8 days**	**Mon 12/4/23**	**Wed 12/13/23**	
4	3	Project Selection	3 days	Mon 12/4/23	Wed 12/6/23	
5	4	Study of Existing System	5 days	Thu 12/7/23	Wed 12/13/23	3
6	5	**Project Analysis**	**9 days**	**Thu 12/14/23**	**Tue 12/26/23**	**2**
7	6	Requirements Gathering	2 days	Thu 12/14/23	Fri 12/15/23	4
8	7	Feasibility Analysis	4 days	Sat 12/16/23	Thu 12/21/22	6
9	8	Algorithm Selection	3 days	Fri 12/22/23	Tue 12/26/23	7
10	9	Project Title Defense	0 days	Tue 12/26/23	Tue 12/26/23	8
11	10	**Project Design**	**7 days**	**Wed 12/27/23**	**Thu 1/4/24**	**5**
12	11	Database Design	3 days	Wed 12/27/23	Fri 12/29/23	9
13	12	Prototyping	4 days	Sat 12/30/23	Thu 1/4/24	11
14	13	**Implementation**	**32 days**	**Thu 1/4/24**	**Sat 2/17/23**	**10**
15	14	Front End Development	10 days	Fri 1/5/24	Thu 1/18/24	12
16	15	Data PreProcessing and EDA	2 days	Fri 1/19/24	Sat 1/20/24	14
17	16	Feature Engineering	3 days	Tue 1/23/24	Thu 1/25/24	15
18	17	Mid Term Defense	0 days	Fri 2/2/24	Fri 2/2/24	9,10
19	18	Backend Development	12 days	Sat 2/17/24	Fri 3/1/24	14
20	19	Algorithm Modeling and Evaluation	8 days	Mon 3/4/24	Wed 3/13/24	16,18
21	20	Model Pipeling and Integration	2 days	Thu 3/14/24	Fri 3/15/24	19,18
22	21	**Testing**	**9 days**	**Mon 3/18/24**	**Sat 3/30/2024**	**13**
23	22	Unit Testing	5 days	Mon 3/18/24	Fri 3/22/24	
24	23	Integration Testing	2 days	Mon 3/25/24	Tue 3/26/24	22
25	24	System Testing	2 days	Fri 3/29/24	Sat 3/30/2024	23
26	25	Final Defense	0 days	Sat 3/30/2024	Sat 3/30/2024	24

Here is the Gantt chart based on the work break down structure:

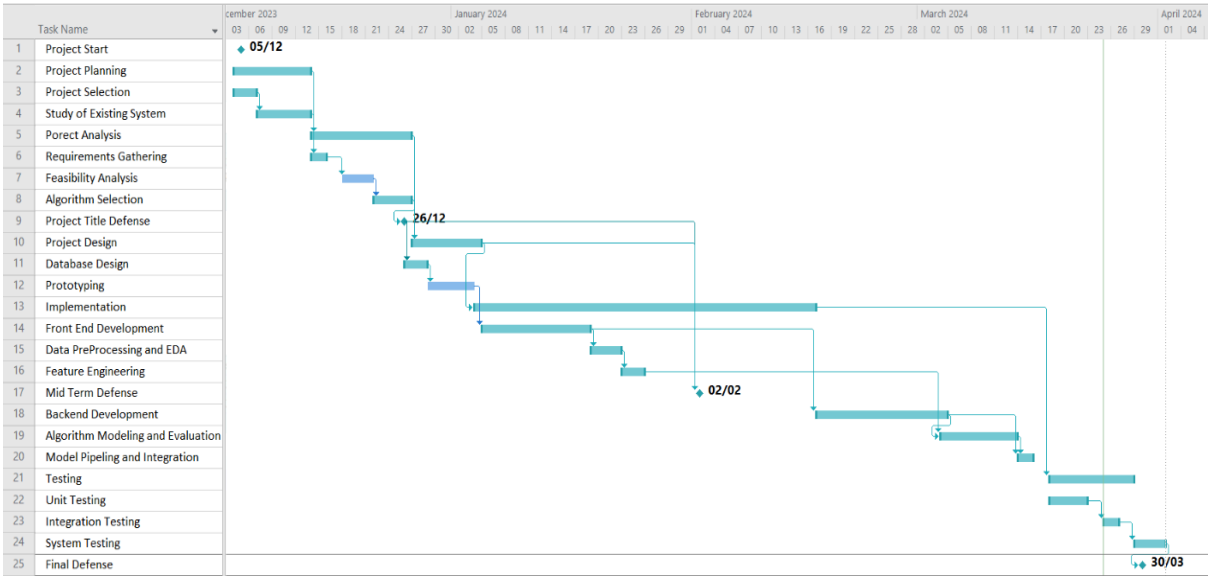


Figure 3.2 Gantt Chart of the project

3.1.2.2 Operational Feasibility

Operational feasibility is a process of assessing the degree to which a proposed system solves the business problems or takes an advantage of a business opportunities. Loan approval system is operationally feasible. User with basic knowledge can use this system.

3.1.2.3 Technical Feasibility

Technical feasibility is a process of assessing the development organizations or individual's ability to construct a proposed system. All members of the project are familiar with the tools and technologies used in the project. Hence, the project is technically feasible.

3.1.3 Analysis

In the analysis phase, requirements of the system are structured. Data modelling technique is used for structuring our requirements.

3.1.3.1 Data Modeling

Data Modeling is the process of preparing a detailed model that captures the overall system's structure through the data. For data modeling, ER diagrams are used as the major tool.

ER diagram

ER diagram represents the real-world entities and their relationships with each other. The following is the ER diagram of the projects:

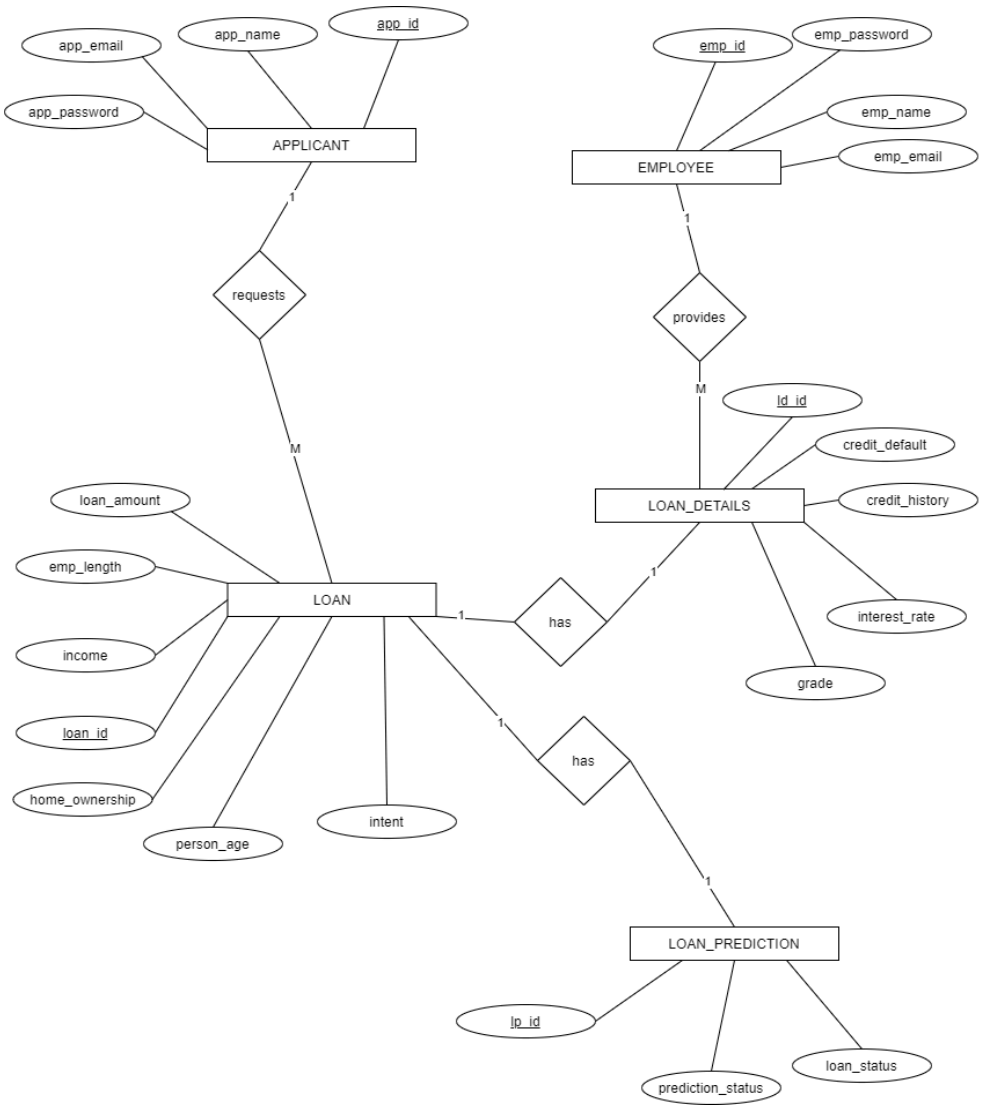


Figure 3.3 ER diagram of the project

In the given ER diagram, there are 5 different entities: applicant, employee, loan, loan details and loan prediction. There are 4 different relationships between these entities. Two of them are One to One Many relations (loan with loan details, loan with loan prediction) and two of them are One to Many relations (employee with loan detail and applicant with loan).

CHAPTER 4 : SYSTEM DESIGN

4.1 Design

Design of the system requires careful planning and thinking while developing the system. Overall performance of the system is completely dependent upon the design of the system. In this project class diagram, activity diagram and sequence diagram are used for showing the overall workflow of the system.

4.1.1 Class Diagram

Class diagram is the diagram which shows the static nature of the system in an object-oriented approach. It properly describes the classes in their relationships. It provides elements of the classes and their behavior while designing the system.

In the given project, there are 5 different classes namely: applicant, employee, loan, loan details and loan prediction. Applicant and Employee have name, email and password as their attributes respectively. Employees has staff status in the system which differentiates them from applicants. Both users have login and user_list methods. Loan has home_ownership, emp_length, loan_amount, person_age and income as attributes which are provided by the applicant. Loan details class have credit_default, credit_history, interest_rate, grade as attributes. These attributes are provided by employees. Both applicant and employee can look after loan prediction which predicts loan status provided by these users.

Here is the descriptive class diagram of the project:

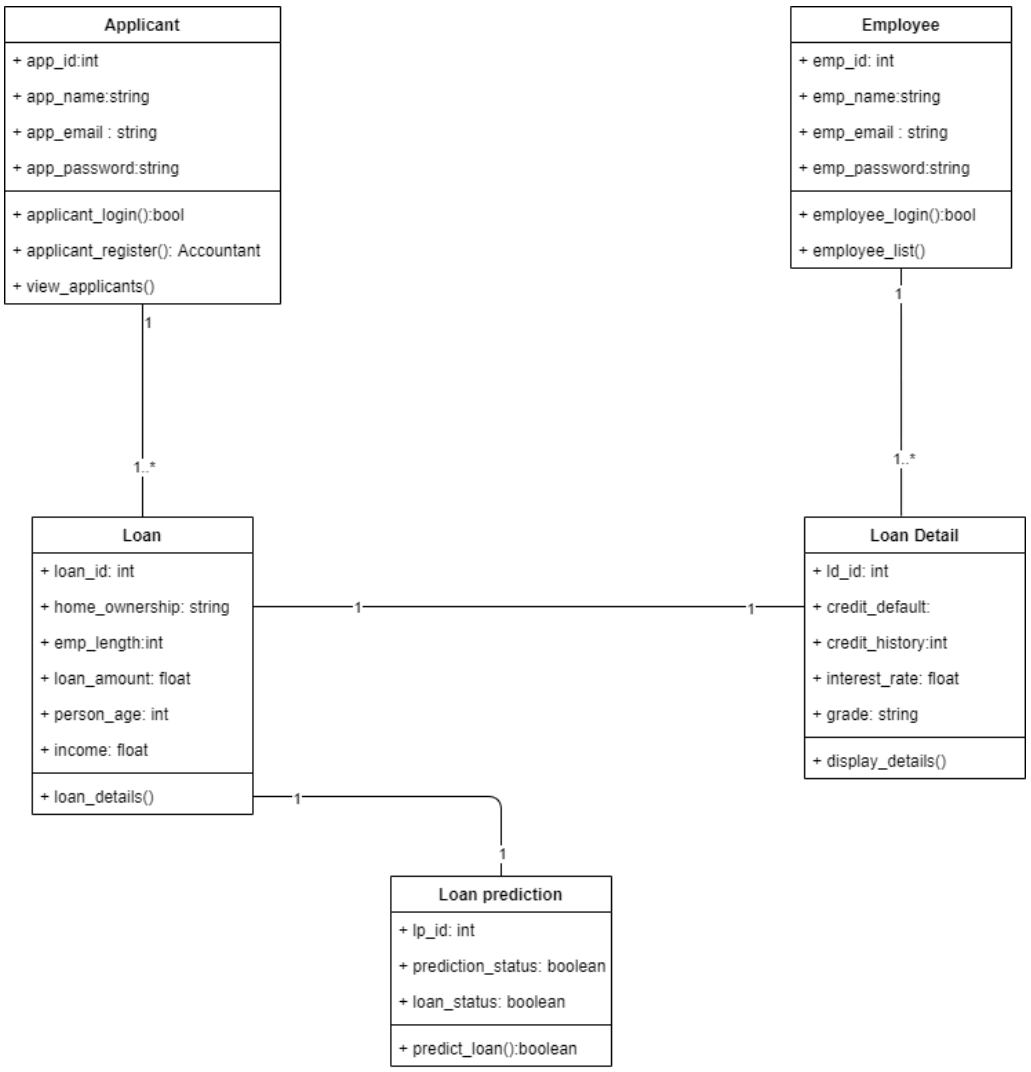


Figure 4.1 Descriptive class diagram of the project

4.1.2 Activity Diagram

Activity Diagrams are the diagrams in UML which are used for showing the flow of activities in a system. It represents the dynamic behavior of a system and provides how activities are performed and coordinated. [7]

Here is the activity diagram of loan request process:

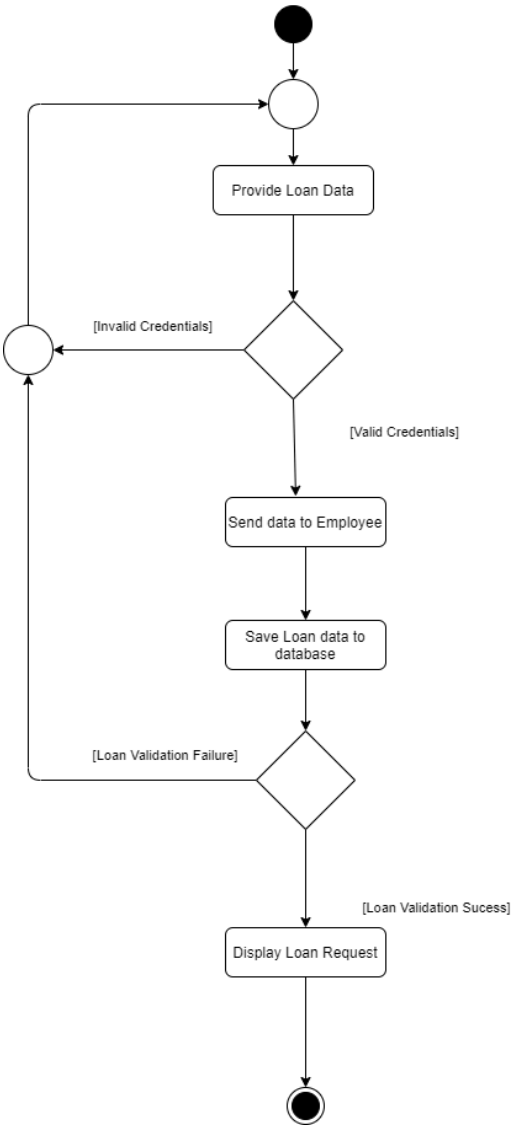


Figure 4.2 Activity Diagram for Loan Request Process by Applicant

4.1.3 Sequence Diagram

Sequence diagram is dynamic UML diagram which provides the interaction between two or more objects during a certain period of time. For each use cases, there are different functionality such that each sequence diagram shows interactions of specific use cases only.

Here is the sequence diagram for view loan prediction process:

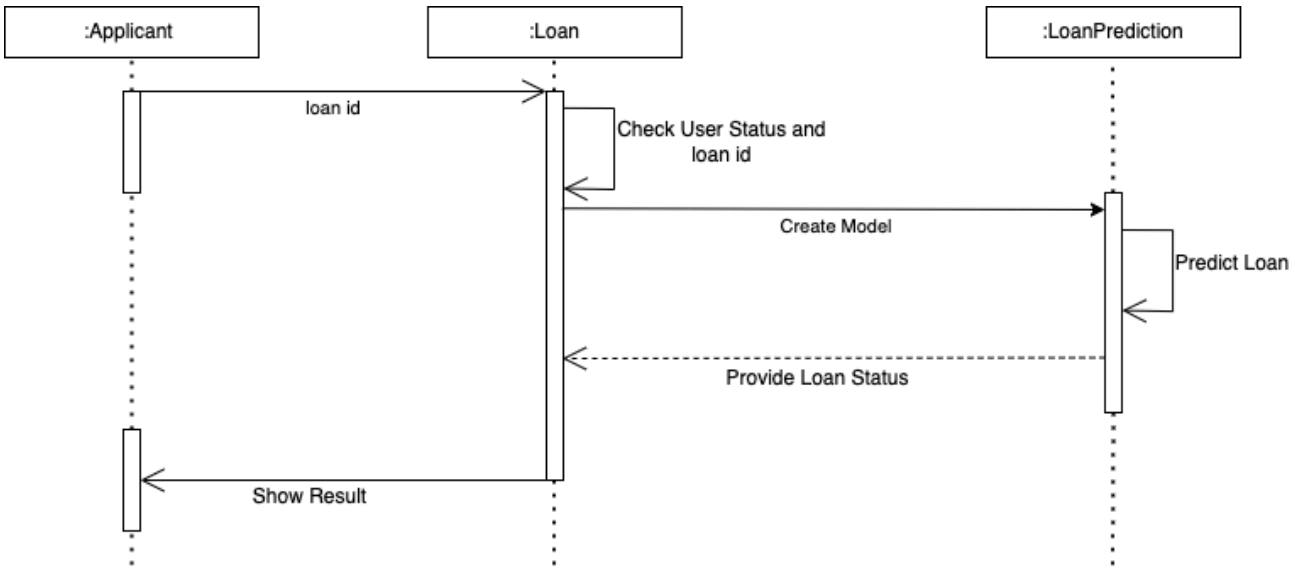


Figure 4.3 Sequence diagram of View Loan Prediction Process

In the given diagram, there are 3 different objects involved namely: Applicant, Loan and Loan Prediction. Overall process is initiated when Applicant clicks over the given loan for prediction. Loan id is passed as message to Loan object along with user's staff status. Loan object now checks loan id for loan prediction. If Loan id is not found, not found message is passed to Applicant. Also, User's status is checked while retrieving loan's information. Now, Loan object initiates Loan Prediction object by executing create model method. Loan prediction predicts loan's status based on loan's information. Loan status is sent to Loan object. Now, Loan status with detailed loan information is passed to Applicant in the form of result.

4.1.4 System Architecture

Architecture design of system is the design of the overall structure, components, modules and subsystems of a system based on requirements provided. Architectural design of a system is concerned with understanding organization and design of the overall structure of a system. It is a critical link between requirement engineering and design. [8]

This project is based on Server side MVC architecture. In this architecture, application's user interface is separated into three different components: Model, View and Controller. The MVC architecture in this project is implemented in client side of an application. Model is responsible for managing the system data and other associated operations related to data. Model primarily manages the database, migrations of the database, overall business logic of an application and provides an interface for interacting with data. Controller component is

responsible for managing interaction between View and Model. It handles inputs and updates the model and view accordingly. View Component represents the user interface of an application. It defines and manages the presentation of data to the user. The key benefit of server side MVC is it allows for highly scalable applications as servers can handle multiple requests from multiple clients simultaneously. It also provides easy integration with other services.

Here is the basic diagram of Server MVC architecture:

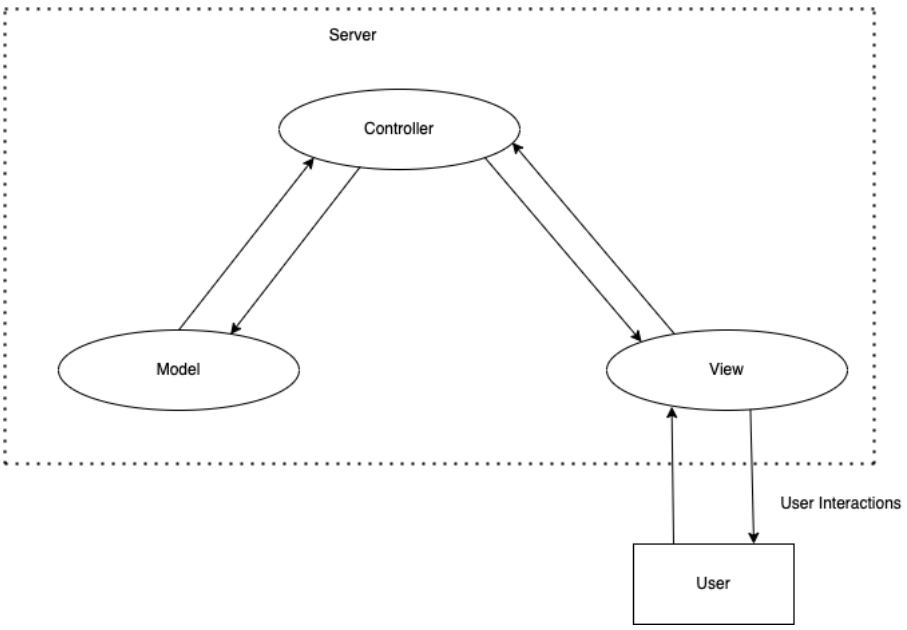


Figure 4.4 System Architecture

4.2 Algorithm Details

This project uses tree-based architecture with hierarchical structure of splitting each node with best feature column from dataset. Since, the project has fixed set of outcomes i.e., loan acceptance and loan rejection, categorical tree-based architecture is used for the project.

Here is the basic diagram for decision tree:

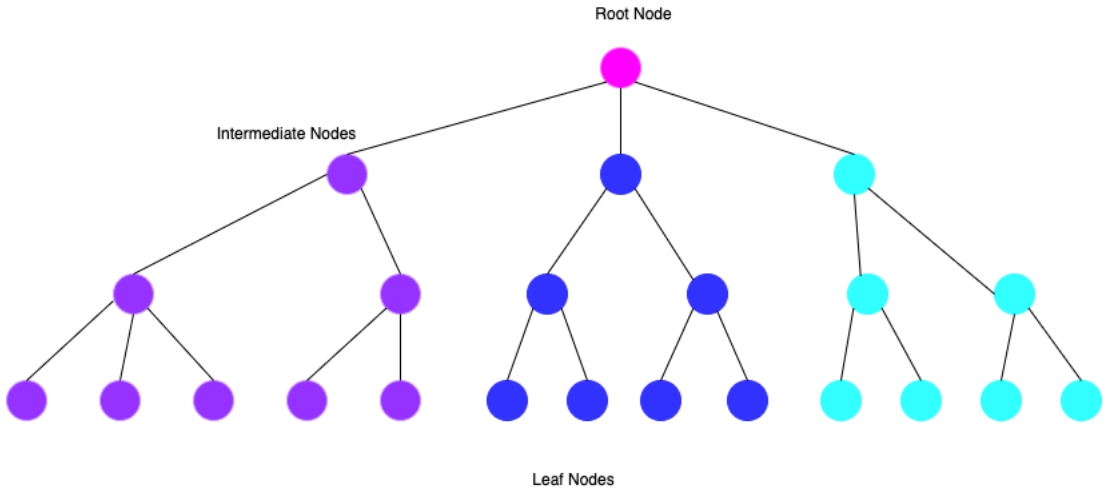


Figure 4.5 Decision Tree

4.2.1 Decision Node

Decision nodes are decision points which split data into two or more subgroups based on the value of a specific attribute in the data. Each decision node contains:

- Splitting column name of the dataset through which data items are splitted to left or right node.
- Threshold value for split.
- Left node object reference.
- Right node object reference.
- Impurity metrics of the selected column on the basis of which the data items are splitted to left or right node.
- Result value that is calculated for calculating an information gain of each column.

4.2.2 Impurity Metrics

Impurity metrics are the evaluation metrics used for determining the best split column of the data at each node of the decision tree. Impurity metrics simply look after the measure of purity at each feature column in the given decision node such that the best value of the overall data columns is selected for the further split. The goal of impurity metric is to maximize the homogeneity of the data in each partition.

Two different impurity metrics would be used in this project. They are:

Entropy: Entropy measures uncertainty associated with the data. It uses the probability of given set of values in columns and overall logarithmic measure for calculating impurity over the data columns.

Entropy is defined as:

$$H(x) = \sum_{i=1}^{c} - \ p(i) * log_2(p(i))$$

Where, i is feature name indices,

c is total number of columns in the dataset,

x is a dataset used for training.

Since, Entropy is a probabilistic measure it ranges from 0 to 1. The value of entropy is 0 when all values at a given data column are of the same class. The value of entropy is 1 when the value of data columns is evenly distributed over all classes.

Here is the plot of entropy for value of probability ranging from 0 to 1:

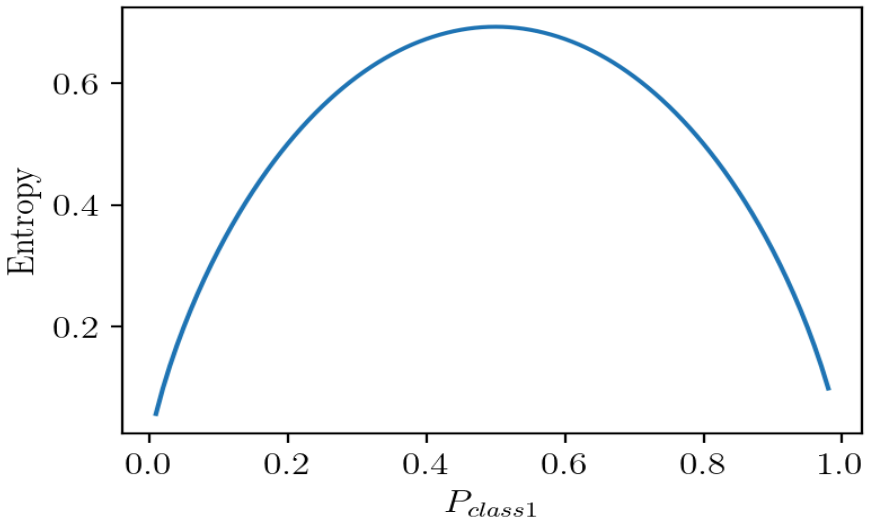


Figure 4.6 Plot between Entropy and Probability of class.

Information Gain: Information gain measures quality of split at given decision node based on amount of information gained by each column with the target variable. Here, the column which has gained more information to target variables are selected based on their occurrence.

Information gain not only considers the quality of splitted data items but also considers total size of each data split and size of outcomes for each data split.

Information gain is defined as:

$$Information\ Gain(x, A) = H(x) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} * H(S_v)$$

Where, A is a column name.

S is the probability of given target outcomes.

S_v is the probability of given target outcomes with the set of feature column values.

4.2.3 ID3 algorithm

ID3 algorithm is a classification algorithm which uses a greedy approach for building a decision tree by selecting the best column attributes which provides maximum information gain or minimum entropy. In each decision tree in ID3 algorithm, node represents a feature attribute of the dataset, branch represents a decision or rule or condition for the given feature attributes of the dataset. Leaf represents the outcome of a given data.

Here are steps of an ID3 algorithm:

Input: Dataset (S), Maximum Depth (L), Minimum Samples to split (M)

Step 1: Create a root node with all the datasets (both input features and their respective output labels), Set Current Level (l) as 0.

Step 2: Find the best input features to split the dataset using impurity metrics.

> Step 2.1: Calculate overall entropy of the given dataset with output labels (y) i.e. $H_y(S)$
>
> Step 2.2: For all input features (X)
>
> > Step 2.2.1: Calculate entropy of given column (X_i) with a different set of values. i.e. $H_{x_i}^{value}(S)$
> >
> > Step 2.2.2: Calculate information gain of X_i through the help of entropy i.e. $Information\ Gain\left(S, x_i\right)$

Step 2.3 Compare information gain of all columns.

Step 2.4: Select maximum information gain column as best split.

Step 3: Divide given S with the selected column as the split.

Step 4: Create child nodes (Right child node and Left child node) based on the given split value and threshold value. Left node being a dataset with smaller than threshold value. Right node being a dataset with larger than threshold value.

Step 5: Increase l by 1. Check M with no. of datasets, Check L with l. Such that if not satisfied, stop the iteration.

Step 6: Repeat Step 2 to Step 4 recursively, until leaf nodes are found.

4.2.4 Ensemble Learning

Ensemble learning is the technique in machine learning that involves combining two or more weak models of same type or different natures with low accuracy or results for predicting overall performance of a system. There are three different ways used for ensemble learning. They are:

- Bagging
- Boosting
- Stacking

Boosting sequentially builds models such that one model's output is passed and improved by another model. Stacking uses meta-models for combining predictions of individual models. Bagging simply splits the data and combines the result based on the nature of the result. Boosting and stacking are done for the complex nature of results.

Ensemble learning easily overcomes robustness, accuracy of the model. It provides generalized output than other forms of machine learning techniques. It also provides a solution towards overfitting by properly using multiple models.

4.2.5 Bagging

Bagging (also known as Bootstrap aggregation) is the technique of machine learning which involves turning the overall data into multiple sets of data through random selections of samples with replacement.

Bagging involves two different steps. They are:

Bootstrapping: Bootstrapping is the technique of sampling which splits the overall data into multiple samples with randomized selection of data with replacement.

Aggregation: Aggregation is the process of combining the results provided by independently trained models generated from bootstrapped data. Aggregation combines those data by either taking maximum value or taking minimum value or taking an average of the overall result.

4.2.6 Voting classifier

Voting classifier is an ensemble learning technique in machine learning which provides a combination of two or models and uses bagging for the splitting data and provides the result through use of voting mechanisms provided by individual models. There are two types of voting mechanisms conducted in voting classifiers. They are:

Hard Voting: In this voting classifier, the final model predicts the output class by taking majority voting of the output class labels.

Soft Voting: In this voting classifier, the final model predicts the output class based on the highest probability of the output class labels provided by the individual models.

4.2.7 Random Forest Algorithm

Random Forest is an ensemble learning algorithm which uses two or more decision trees to form voting classifiers for providing accurate and robust results to real-world data. Here hard voting is used as a voting mechanism for selecting the best prediction over other sets of output predictions.

Here is an algorithm for random forest algorithm:

Input: Dataset (S), Maximum Depth (L), Minimum Samples to split (M), Number of trees (N)

Step 1: Randomly sample S into S_1, S_2, S_3, …, S_N and create N different trees into T_1, T_2,.. T_N.

Step 2: For each decision tree T_i, use ID3 algorithm for given bootstrapped sample S_i with M as maximum samples to split and L as maximum depth of the decision tree.

Step 3: For given set of input data X, predict the output (P_i) with the help of trees T_1, T_2,.. T_N.

Step 4: Use hard voting method for aggregating the data into the output prediction P.

i.e., $P = max(P_1, P_2, ..., P_N)$

Here is the diagram showing ensemble of decision trees forming random forest:

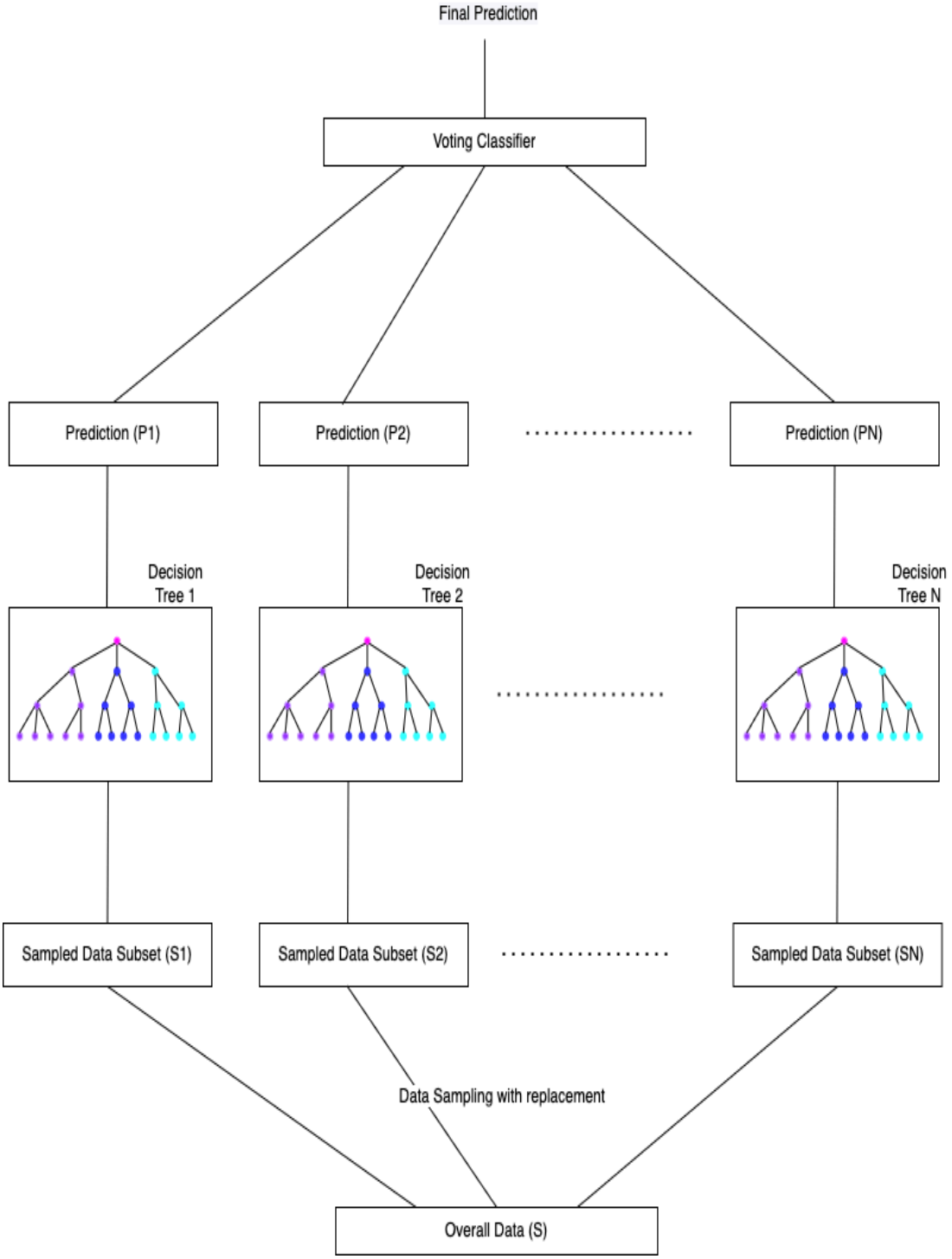


Figure 4.7 Random Forest Classifier

CHAPTER 5 : IMPLEMENTATION AND TESTING

5.1 Implementation

System implementation involves the development of a working system by the use of proper tools and technologies suitable for the project. This process provides translation of designs developed into a usable system.

5.1.1 Tools Used

The following tools were used for implementing the project and designing the documents in project:

Implementation Tools: Implementation tools are tools that help in overall implementation of the project. Different tools were used for development of systems in front end, back end and algorithm development of a system. They are:

Front-end Tools

Front end tools are those tools which provides implementation to an application where user can directly interact with the overall system. Front end of a system is an interface between the user and the back end. The following front-end tools were used for developing project:

HTML5

HTML is the markup language which is used for structuring the content of a webpage. HTML along with CSS is mostly used for developing web pages.

CSS3

CSS is the stylesheet developed by W3C for presenting the webpages. It uses HTML elements for presenting the web pages. The separation of presentation from HTML enables proper layout, content font and colors to the webpages.

JavaScript

JavaScript is the client-side scripting language. JavaScript and its other packages are used for making content dynamic and fetching data from the server asynchronously.

Back-end tools

Back-end tools are tools used for providing proper functionality of a system when a user accesses the web pages. Back-end of an application provides the required resources and access for data when front-end requests to the system. The following back-end tools were used for developing project:

Python (v3.9.13)

Python is a scripting language for general purpose programming. Python scripts in back-end are used for populating the database for visualization purpose, integrating machine learning models and interacting with web pages such as templating.

Django (v4.1.2)

Django is free and open-source web development framework used for developing complex and secure application. It is based on server-side MVT architecture and is similar to MVC architecture. Django is used for providing dynamic behavior of an application.

Sqlite (v3)

SQLite is light weight disk-based RDBMS which can be used for web applications. It is file on disk database system and can be easily accessed through the SQL commands for data definition, data manipulation and other related tasks. Because of its file-based nature, it can be easily used for embedding the program and does not require any type of engine for access of data.

Visual Studio Code

Visual Studio Code is a light weight, cross platform and open-source IDE for developing different applications. It provides wide variety of features of development of a project.

Algorithm development tools

These tools are used for development and deployment of algorithm to the back-end of the system. They are:

Anaconda (v22.11.1)

Anaconda is a cross package management software developed for python especially for data science and related field. Different Machine learning tools like: Jupyter, Pandas and Numpy are properly included while installation of the software. It also provides “conda” a command

line tool which provides users to manage packages and create environments based on projects requirements.

Jupyter notebook (v8.5.0)

Jupyter notebook is an interactive, open-source web-based environment used for analysis and development of different applications. It provides facility of visualization of data and collaboration with team members for faster development of projects.

Git (v2.38.1) and GitHub

Git is a software configuration management tool used for controlling versions of software in distributed form. Each change in features of the versions were tracked through git and team collaboration was properly conducted. GitHub is a web-based service which provides hosting for the git repositories. GitHub provides features related to pull requests, tracking issues and code reviews.

CASE tools

CASE tools are tools used for analyzing and designing the project. They are used for designing necessary diagrams and providing various project artifacts. Here are CASE tools used in our project:

Draw.io

Draw.io is free and open-source cross platform CASE tool used for creating different wireframes and diagrams like: ER diagram, Class diagram, Use case diagram, activity diagram.

Microsoft Project 2019

Microsoft project is the project management tool. In this project, Microsoft project is used for breaking down works, allocating resources, creating a work-schedules and looking after risks associated with the task.

Figma

Figma is a cloud-based design tool used for developing user interfaces and prototypes. Figma was used for designing wireframes and prototypes of web application in this project.

Dependencies

Dependencies are the libraries required for implementation and running the project. In this project there are two different types of dependencies used. They are:

1. **Python Built-in modules**

   These are the libraries developed by python software foundation. These libraries are preloaded with python software. Here are built-in python modules used for the project:

Table 5.1 Built-in python modules

i.	**os**	This package was used for interacting with file in the local storage.
ii.	**pickle**	This package was used for saving and using trained model.
iii.	**counter**	This package was used for selecting maximum occurrence for voting classifier.
iv.	**time**	This package was used for calculating time
v.	**sys**	This package was used for defining the frozen files.
vi.	**logging**	This package was used for logging the back-end interactions.

2. **External modules**

   These are the external libraries and packages developed by third party developers or community. External modules must be downloaded separately. Here are external modules of the project:

Table 5.2 External Modules used in project

i.	**numpy**	This package was used for mathematical operations of data.
ii.	**pandas**	This package was used for manipulation of dataframe.
iii.	**sklearn**	This package was used for data preprocessing and pipelining.
iv.	**joblib**	This package was used for saving pipelined models.

v.	**matplotlib**	This package was used for low level visualization of data.
vi.	**chart.js**	This package was used for visualizing the data to users (front end).
vii.	**imblearn**	This package was used for experimentation related purposes.
viii	**missingno**	This package was used for getting knowledge related to missing number.
ix.	**seaborn**	This package was used for advanced level plotting.

Hardware Configuration

For this project's algorithm development, hyperparameter tuning and evaluation related purposes, two different hardware configurations were used. Here are the details of the system configurations used for implementation:

Table 5.3 Hardware configuration used in project

Machines	Machine I	Machine II
Operating System	Windows 11	Windows 10
Processor	AMD Ryzen 5 5500U@2.10 Ghz	Intel Core i7-5500U @ 2.40 Ghz
Physical / Total Cores	6/12	2/4
Main Memory	8 GB	8 GB
Disk Space	256 GB	256 GB

5.1.2 Implementation Details

In this project, overall project is divided into three different components i.e., front-end section, back-end section and algorithm section for development. Here is the diagram showing implementation process of an algorithm in detail.

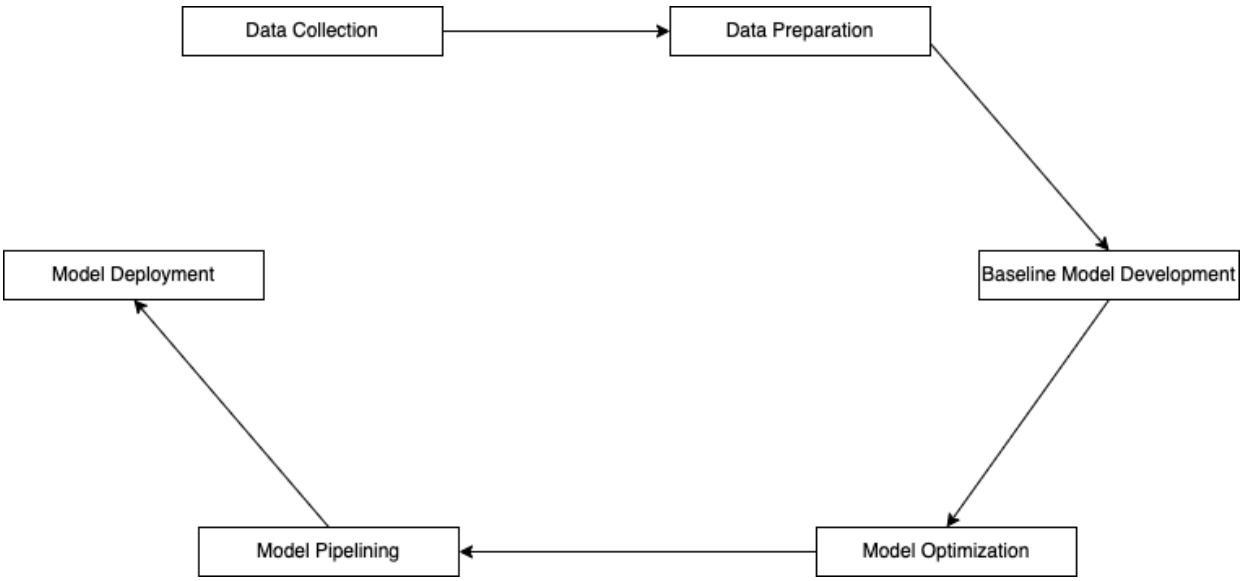


Figure 5.1 Algorithm Implementation Procedure

Data Collection

Two different techniques were used for data collection procedure. i.e. Manual and Automation. Since, the data was in zip format. Unzipping process was conducted manually. In manual procedure, the data was downloaded from publicly available repository. In automation process, python script for downloading the data to the local storage was created.

Here were the columns of the data downloaded.

Table 5.4 Data columns

Data Column	Data Type	Range/ Value
person_age	Integer	20-144
person_income	Floating point	4000-600000
home_ownership	Categorical	OWN, RENT, MORTGAGE, OTHER
emp_length	Floating point	0-123
loan_intent	Categorical	EDUCATIONAL, PERSONAL, MEDICAL, VENTURE, HOMEIMPROVEMENT, DEBTCONSOLIDATION
loan_grade	Categorical	A, B, C, D, E, F, G

loan_amount	Integer	500-35000
interest_rate	Floating point	5.42-23.22
loan_percent_income	Floating point	0-0.83
credit_default	Binary	yes, no
credit_history	Integer	2-30
loan_status	Binary	0,1

There were 32,581 instances of data among them 3116 instances of instances of data instances had missing data. There were 165 instances of repeated data. Loan_status column is taken as the output label of the algorithm.

Data Preparation

Data Preparation procedure consisted of seven different steps. They are:

1. **Exploratory Data Analysis**

   In Exploratory data analysis step, different visualization techniques were applied for gaining knowledge of data different operations for given feature.

   Correlation plot was created for looking after similarity between two columns. Highest similarity between two columns were credit_history and person_age with 86 percentage. Highest Dissimilar columns are loan_percent_income and person_income with 25 percentage.

   Count plot was created for looking after the count of the output label column. There were about 25000 instances of loan rejection data and 7000 instances of loan acceptance data. There was highly imbalanced data such that balancing techniques must be applied for experiments.

   Through distribution plot, skewness of the emp_length column was known such that the data were left skewed.

   Pie chart of the loan_grade column provided that, there were hugely imbalanced data. Loan grade values: E, F and G were very less. They must be properly experimented through under sampling or over sampling procedure.

2. **Train Test Split**

The overall data of 32581 instances was split into two sets of data. i.e., training set and test set with 25 percent test size. Since, the data was found to be imbalanced, the data is stratified based on the output label. Training set is used for training the model whereas test set is used for evaluating the model's performance.

Here is the data split and their size:

Table 5.5 Dataset split size

Dataset	Size
Training Features	(24435,11)
Training Labels	(24435,1)
Test Features	(8146,11)
Test Labels	(8146,1)

3. **Data Imputation**

Data imputation procedure fills in the missing data in the dataset. It removes problem errors and data omissions from the dataset. Median imputation for person_emp_length and interest_rate column. For missing data, overall median of the data was used for filling in the missing data. After data imputation procedure is used, the missing instances of both training and test set is reduced to 0.

4. **Scaling**

The dataset had different range for each column. Scaling of the data outputs the same standard format of the data. It is done for numerical data columns of the dataset. This project uses standard scaling procedure for scaling the data.

Here is the formula for standard scaling:

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

Here,

x_{scaled} is the scaled output of the data.

x is the original data that is passed for scaling.

μ is the mean of the data column.

σ is the standard deviation of the data column.

Input Columns Person_age, person_income, person_emp_length, loan_amount, interest_rate, loan_percent_income, credit_history was passed for standard scaling the data.

5. One-hot encoding

One hot encoding is the data preprocessing technique used for encoding each categorical column. Here category of the individual column is changed into the vector form. Categories after conversion are represented by binary form.i.e. 0 or 1.

Here is an example of one-hot encoding for loan_intent column.

Table 5.6 One-hot encoding

Original Column	Ohe_OWN	Ohe_RENT	Ohe_MORTGAGE	Ohe_OTHER	vector
OWN	1	0	0	0	[1,0,0,0]
RENT	0	1	0	0	[0,1,0,0]
MORTGAGE	0	0	1	0	[0,0,1,0]
OTHER	0	0	0	1	[0,0,0,1]

In this project, one-hot encoding of home_ownership, loan_intent and loan_grade column was done.

6. Label encoding

Label encoding is the data preprocessing technique which uses encoding of categorical column into a numerical form. Here each category of the column is assigned with specific numbers also known as labels. Label encoding is similar to one hot encoding but it has lower computation complexity than one hot encoding as there

is vector form of output result from one hot encoding but label encoding has single column output.

Here is an example of label encoding for credit_default column:

Table 5.7 Label Encoding

Origin Column	Label Encoded Column
Yes	1
No	2
Yes	1

Baseline Model Development

Baseline model of the project is the simple model through which we can test with other complex models for comparison of the performance. Baseline model is just the sophisticated version that will be used for the overall evaluation at the start of the project. Through result of baseline model, we can look after new improvements or other related tasks.

Decision tree with default parameters was taken as the baseline model of the project. Here is the performance result of the baseline model:

Table 5.8 Baseline model metrics for training and test dataset

Dataset	Imbalanced Accuracy	Balanced Accuracy	AUC score	Macro average Precision	Macro average Recall	Macro average F1-score
Training	86.74	75.65	75.65	0.83	0.76	0.78
Test	85.82	75.65	75.65	0.85	0.86	0.85

Model Optimization

Model optimization is the process of improving the model's performance with the help of adjustment of hyperparameters. Model optimization significantly increases the complexity of the machine learning model as there are number of parameters to tweak for the best performance of the model. The goal of model optimization is to select the best hyperparameters and increase the performance metrics of model.

In this system, we used two different hyperparameters for the decision tree model. They are:

1. Minimum samples to split
2. Maximum depth

For random forest algorithm, three different parameters were used. They are:

1. Minimum samples to split
2. Maximum depth
3. Number of decision trees.

Model Pipelining

Model Pipelining is the process of creating a step-by-step procedure of interconnected data preparation, model training procedure in a form of sequential pipeline form. It involves in combination of data preprocessing, feature extraction and model training in single workflow. Model pipelining procedure simplifies the model training process such that it can be easy to automate the training procedure.

Here is the model pipeline of the system:

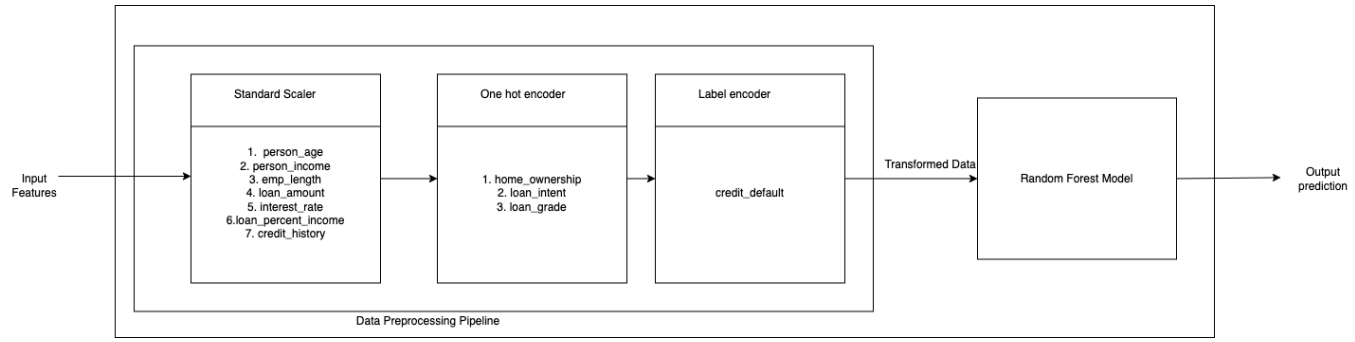


Figure 5.2 Model pipeline of the system

Model Deployment:

After model pipelining, the model was converted into pickled format through the help of pickle. The pickled model was now integrated with the web application. The hyperparameters selected while optimizing models were selected in model pipelining were frozen for the prediction purpose only. Loan requested by user and loan details data provided by employee of an organization is properly passed to the frozen model such that it can predict the output reliably.

5.1.3 Experiments Conducted

The data used in the project were imbalanced. Experimentation of balancing the data was conducted for the project. Three different experiments for algorithm development were used:

Experiment I: Data Under sampling

Data under sampling is a technique for data balancing where a subset of the majority class instance is randomly selected and removed from the dataset such that there will be equal instance of all classes. Data under sampling procedure removes bias towards the majority class. This can provide model's ability to improve in classifying instances of minority classes. For data under sampling, sampling strategy is the threshold value for selecting the data instances.

In the project, data under sampling is conducted with a sampling strategy of 90%. Here about 90% of the majority classes were selected and 10% of majority classes were dropped. Here is the number of data instances of training and test data before and after random data under sampling:

Table 5.9 Data Size for random data under sampling

Dataset	Previous Size	After Sampling Size
Training Features	(24435,11)	(12043,11)
Training Labels	(24435,1)	(12043,1)
Test Features	(8146,11)	(2961,11)
Test Labels	(8146,1)	(2961,1)

Experiment II: Data Oversampling

Data oversampling is a technique for data balancing where a subset of the minority class instance is randomly selected and duplicated from the dataset such that there will be equal instance of all classes. Data under sampling procedure removes bias and variance towards the majority class. This reduces mechanism of overfitting.

In the project, data oversampling is project is conducted with sampling strategy of 90% was used. In this experimentation about 10% of the data were used for duplication of minority class. Here is the number of data instances of training and test data before and after random data oversampling:

Table 5.10 Data Size for random data oversampling

Dataset	Previous Size	After Sampling Size
Training Features	(24435,11)	(40718,11)
Training Labels	(24435,1)	(40718,1)
Test Features	(8146,11)	(10228,11)
Test Labels	(8146,1)	(10228,1)

Experiment III: Hyperparameter tuning

Hyperparameter tuning is the process of selecting an optimal hyperparameter for improvement of performance of an algorithm. It is an iterative process of selecting a range of hyperparameter such that there is a robust algorithm. Through the given range of hyperparameter, different combinations of hyperparameters are selected and used for evaluating the performance of an algorithm. Some of the examples of hyperparameter tuning are: manual tuning, grid search, random search and Bayesian optimization.

In this project, manual tuning and grid search of hyperparameter tuning were used. Manual tuning is the procedure of tuning the hyperparameters by selecting hyperparameters manually. Manual tuning procedure is time consuming and selected by the model trainer manually.

Grid search is hyperparameter tuning procedure which involves in properly defining each possible combination grid of hyperparameters. In grid search procedure a simple script of training the possible combination is provided. Grid search is generally computationally expensive but it provides proper optimal model.

Here are the range of hyperparameters provided when conducting hyperparameters for decision tree and random forest models with manual tuning and grid search:

Table 5.11 Hyperparameters for Decision Tree

Hyperparameter Name	Range
Minimum samples to split	2, 5,10,25,50,100
Maximum depth	5,8,10

Table 5.12 Hyperparameters for Random Forest

Hyperparameter Name	Range
Minimum samples to split	2,5,10,25,50,100
Maximum depth	5,8,10
Number of trees	5,10

5.2 Testing

Testing process involves finding out bugs and errors, and fixing those bugs and errors. Testing process is performed parallelly while implementing the system. It is intended to show that the program does what it is intended to do and to discover if there is different behavior than expected behavior. This provides proper look after discovery of defects before a software application is put into use.

5.2.1 Test Cases for Unit Testing

Unit testing is the process of testing individual program units and methods or object classes to look after errors or bugs. Unit testing generally focuses on testing after the functionality of

given unit. Different modules and components are tested individually such that defects of the system can be found.

Here are test cases evaluated during unit testing process.

Test Case for Applicant Register

Table 5.13 Test Case for Applicant Register

Test Case Name	APPLICANT REGISTER http://127.0.0.1:8000/applicant/register/	Test Case ID	TC-01
Test Case Description	Test Case for Applicant register	Test Priority	High
Prerequisite	Provide Username Password and Password confirmation need to match for Submit and also provide Verification Email	Post-conditions	User is stored in database and successfully. The account should be validated session details are logged login to account.

Test Execution Steps:					
S. N.	Action	Test Steps	Expected Output	Actual Output	Test Result

1	Navigate to Applicant Registration		Applicant should be able to Register	Applicant is navigated to Login page	Pass
2	Verification of Login Page with Username, and Password for user who does not exit.	1.Provide applicant details: Username "abishekl" 2. Provide applicant password: "Nepal123" -Submit Button clicked	Error prompt: Please enter a correct username and password. Note that both fields may be case-sensitive.	Error prompt: Please enter a correct username and password. Note that both fields may be case-sensitive.	Pass
3	Verification of Applicant Register Page with Username, Password, Password confirmation and Email	1.Provide applicant details: Username "abishekl" 2. Provide applicant password: "Nepal123" 3.Provide confirmation Password: "Nepal123"	Applicant is redirected to Login Page.	Applicant is redirected to Login Page.	Pass

		4. Provide email: abishek44@gmail.com -Submit Button clicked			

Test Case for Applicant Login

Table 5.14 Test Case for Applicant Login

Test Case Name		Applicant login		Test Case ID	TC-02
Test Case Description		Test Case for logging in for applicant		Test Priority	High
Prerequisite		Application URL with valid registered data.		Post-Requisite	NA
Test Execution Steps:					
S. N.	Action	Input	Expected Output	Actual Output	Test Result
1	Navigate to Login Page		User should be able to	User is navigated to login page	Pass

			access login page.		
2	Provide Valid Username	abishek	Accept the username	As expected.	Pass
3	Provide Valid Username	abishek44@gmail.com	Accept the email	As expected.	Pass
4	Provide Valid result	Nepal@123	Accept the password	As expected.	Pass
5	Click on Register Button		User should be redirected to dashboard page.	As expected.	Pass

Test Case for Loan application request

Table 5.15 Test Case for Apply Loans

Test Case Name	Loan apply.	Test Case ID	TC-03

Test Case Description	Test Case for loan application.	Test Priority	High
Prerequisite	Applicant is Logged in to the system and has valid loan details. Applicant is in Dashboard page.	Post-conditions	Loan application is registered to database. The Application is transferred to Loan Manager for further process.

Test Execution Steps:

S. N.	Action	Input	Expected Output	Actual Output	Test Result
1	Navigate to “Apply” page		User is redirected to apply loans page.	User is redirected to apply loans page.	Pass
2	Provide Valid Age	24	Accept user’s age.	As expected.	Pass
3	Provide Valid Loan Intent	PERSONAL	Accept selected loan intent.	As expected.	Pass
4	Provide Valid Property Ownership	OWN	Accept selected ownership.	As expected.	Pass
5	Provide Yearly Income (in $)	3000	Accept user’s income.	As expected.	Pass
6	Provide Employment length (in years)	5	Accept employment length.	As expected.	Pass
7	Provide Loan Amount (in $)	30000	Accept Loan Amount.	As expected.	Pass

8	Select Loan Manager.	Nisha	Accept selected Loan Manager.	As expected.	Pass
9	Click on Submit button		Navigated to "Application Loan list"	"Application Loan list" appeared	Pass

Test Case for Provide Loan Data

Table 5.17 Test Case for provide loan data.

Test Case Name		Provide Loan Data		Test Case ID	TC-05
Test Case Description		Test Case for providing loan data.		Test Priority	High
Prerequisite		Applicant should have asked for the loan.		Post-conditions	Data is ready for system Evaluation and predication
Test Execution Steps:					
S. N.	Action	Input	Expected Output	Actual Output	Test Result
1	Navigate to "Employee Dash Board"		View to "Requested Loans"	As expected.	Pass
2	Click on "provide loan data"		Navigates to "Credit Info"	Navigated to "Credit Info"	Pass

3	Provide Credit history	"2"	Accept Selected Credit history	Accept the Credit history	Pass
4	Provide Grade	"D"	Accept the Provided Grade	Accept Grade	Pass
5	Provide Credit Default	Yes	Accept the Credit Default	Accept the Credit Default	Pass
6	Click on Submit button		Credit Info should attach to Loan data	Loan Data is available with Credit info in Loan Detail	Pass

Test Case for Loan Prediction

Table 5.18 Test Case for loan prediction

Test Case Name	LOAN PREDICTION	Test Case ID	TC-06
Test Case Description	Test Case for loan prediction	Test Priority	High
Prerequisite	Applicant is registered to the system. Loan is already applied by applicant Loan details are provided by the employee.	Post-conditions	User is stored in database and successfully. The account should be validated session details are logged login to account.

Test Execution Steps:					
S. N.	Action	Input	Expected Output	Actual Output	Test Result
1	Navigate to Predicted loans page.		Applicant should be able to view applied loans	Applicant views the applied loans list.	Pass
2	Click on the Predict button for previously applied loans.	Predict button is clicked .	The loan and loan details must be passed to the back-end.	The loan and loan details are passed to back-end.	Pass
3	System logs in loan and loan details		Loan Data and Loan Details should be printed.	Loan data and loan details of the given loan is shown.	Pass
4	Data is passed to Model Preprocessing	Applied loan and loan details are passed.	Preprocessing pipeline must be executed.	Following preprocessing were done: Standard Scaler for numerical columns were conducted. Label encoding for credit_default was conducted. One hot encoding for remaining	Pass

				columns were conducted.	
5	Preprocessed data are passed to random forest model.		Prediction of model must passed to front-end.	Prediction of model is passed to front-end.	Pass.
6.	Front-end processes provided data from back-end.	Click on predict button.	Predicted output must be shown to user	Result: “Approved” is shown as prediction to user.	Pass

5.2.2 Test Case for Integration Testing

Test Title		Integration Testing		Test Case ID	TC-07
Test Case Description		Integration Testing of Loan Data Module		Test Priority	High
Prerequisite		Process is followed by the Registered applicant and employee.		Post-con ditions	Loan Approved or Not Approved is viewed by both sides.
Test Execution Steps:					
S. N.	Action	Input	Expected Output	Actual Output	Test Result
1	Load Data	Provide The loan Data through selection	Load Detail of Applicant should pass to selected Employee.	Loan detail are available on employees "Applicant Loan list" and in "Requested	Pass

		specific Employee		Loans" with some option.	
2	Provide Data	Employee provide "Credit Information" for applicant Loan Request	Credit information attached Loan data.	All information of Loan Data with Credit Info can be view on "Loan Detail" by employee.	Pass
3	System evaluation	Loan detail provided are Evaluated by the system	Provide The Predication Loan is Approved or not Approved	Loan Predication is available for both Employee and Applicant	Pass
4	Loan Visualization	Predicated Loan with different except	Visual of Loan based on proprieties with charts	Loan Information are viewed according to different properties with charts	Pass

5.2.3 Test Cases for System Testing

Test Title	System Testing	Test Case ID	TC-08
Test Case Description	Overall System Testing	Test Priority	High
Prerequisite	Process is followed by the Registered applicant and employee.	Post-conditions	Loan Approved or Not Approved is viewed by both sides.

Test Execution Steps:					
S. N.	Action	Input	Expected Output	Actual Output	Test Result
1	Predict Loan	All necessary Approved information passed to predict loan status	Approve the loan.	Loan Status "Approved"	Pass
2	Provide Data	Loan detail with unproper data is provided to predict loan status	Not Approve the loan	Loan Status "Not Approved"	Pass
3	Performance evaluation	Different data with different instance are passed.	Provide The Predication Loan Result Approved or not Approved	Loan Predication is available for both Employee and Applicant in good success ratio as excepted	Pass
4	Operation	Provide the Web Address	website launches properly with all the relevant pages, features, and logo	As excepted	Pass
5	Functionality Testing	Use functionality of the product	If the major functionality like Load data, Credit Info, Predict, update,	As excepted	Pass

			delete, etc. work properly		
6	Predict	Click on "Predict" Button	Predication result is Available to both employee and applicant	Predication result is Available to both employee and applicant	Pass

5.3 Result Analysis

Result analysis involves looking after performance of the overall algorithm and interpretation of the result provided by the algorithm. In result analysis, different algorithms previously experimented were evaluated for the selection of the best performing model.

In this project following metrics are used for evaluating result:

Accuracy Score

Accuracy Score is the evaluation metric which measures percentage of correctly classified result over total number of instances. Here is the formula for accuracy score:

$$accuracy = \frac{total\ correctly\ classified\ instance}{total\ number\ of\ instances} * 100$$

Accuracy score only looks after correctly classified instances so it is biased when the data is imbalanced. Since, this project has imbalanced nature of data balanced accuracy score is also taken into an account. Accuracy ranges from 0 to 100 percentage.

Balanced Accuracy Score

Balanced accuracy is the evaluation metric is an average of true positive rate and true negative rate. Since, both positive and negative results are measured by this evaluation metric. Balanced accuracy score provides accurate representation of algorithm. Here is the formula for balanced accuracy score:

$$balanced\ accuracy = \frac{true\ positive\ rate + true\ negative\ rate}{2} * 100$$

Here,

$$true\ positive\ rate\ = \frac{True\ Positive}{True\ positive+False\ negative}$$

And,

$$true\ negative\ rate\ = \frac{True\ negative}{True\ negative+False\ positive}$$

Balanced accuracy ranges from 0 to 100%.

AUC Score

AUC score is a metric which looks after the ROC curve which provides graphical representation of the performance of a model at different threshold settings. ROC curve simply plots between True positive rate and False positive rate at different thresholds. AUC after the curve is plotted looks after the area covered under the curve. AUC score ranges from 0 to 1.

Here is the plot for ROC Curve:

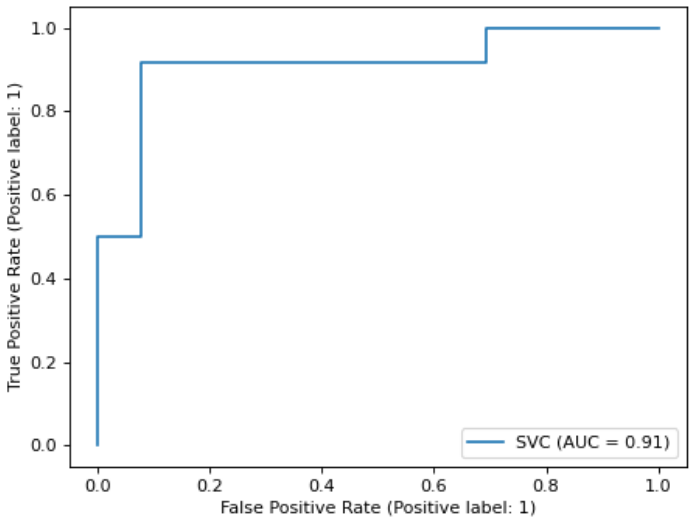


Figure 5.3 ROC Curve

Macro Average Precision Score

Macro Average precision score is an evaluation metric which calculates average precision over the number of result class. Here, Precision is proportion of true positive class over all positive predictions conducted. Here is the formula for Macro average precision:

$$Macro\ avg\ precision\ =\ \frac{precision\ 1+precision\ 2+..\ +precision\ N}{N}$$

where,

$$Precision\ =\ \frac{True\ Positive}{True\ positive+False\ Positive}$$

Macro Average Recall Score

Macro Average recall score is an evaluation metric which calculates average recall over the number of result class. Here, Recall is proportion of true positive by total true instances of the given class. Here is the formula for Macro average recall:

$$Macro\ avg\ recall\ =\ \frac{recall\ 1+recall\ 2+..\ +recall\ N}{N}$$

where,

$$Recall\ =\ \frac{True\ Positive}{True\ positive+False\ Negative}$$

Macro Average F1-Score

F1-score is the harmonic mean of macro-average precision score and macro average recall score. Here is the formula for macro average f1 score:

$$Macro\ average\ f1\ score\ =\ \frac{2*Macro\ average\ Precision*Macro\ average\ Recall}{Macro\ average\ Precision+\ Macro\ average\ Recall}$$

Macro average precision score, macro average recall score and macro average f1 score all ranges between 0 to 1.

Here is the result provided by different models while implementing algorithm:

Table 5.19 Test set results with different models

Model	Accuracy Score (in %)	Balanced Accuracy (in %)	AUC score	Macro average Precision	Macro average Recall	Macro average F1-score
Baseline Model (Decision Tree with no params)	85.82	75.65	0.75	0.85	0.86	0.85
Decision Tree model while under sampling	84.29	83.67	0.83	0.86	0.84	0.84
Random forest model while under sampling	84.22	83.65	0.83	0.86	0.84	0.84
Decision Tree model while over sampling	83.09	83.09	0.83	0.86	0.83	0.83
Random forest model while over sampling	83.31	83.31	0.83	0.86	0.83	0.83
Hyperparameter tuned decision tree (minimum samples to split = 2, max depth = 5)	91.68	82.52	0.82	0.93	0.83	0.86
Hyperparameter tuned random forest algorithm	91.54	84.0	0.84	0.92	0.82	0.86

After proper computation of hyper parameter tuned random forest algorithm is selected as final model. The algorithm had minimum samples to split of 100 samples, max depth of 5 levels and 10 trees. This model provided 84% balanced accuracy which is better thanx all other models used for comparison.

CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATIONS

6.1 Conclusion

Loan approval prediction system is a useful system which demonstrates the use of tree-based machine learning algorithm. The application takes loan and loan details as input from user and provides prediction of loan approval through use of random forest algorithm. The system properly fulfills the objective provided in the project. Algorithm was properly experimented for model performance evaluation with under sampling, over sampling and hyperparameter tuning. The model obtained 84% balanced accuracy score with unable to classify 551 instances of test data out of 6517 instances of test data. Hence, the project can provide proper loan evaluation before thorough analysis is done and can properly be used for financial institutions for better loan approvals to the loan applicants. Loan approval system can be reliable option for looking after credit risk when applicant applies for loan.

6.2 Future Recommendations

For even better results, few things can be added in the future work. This system specially uses bootstrapping but there are other forms of ensemble learning such as: Boosting and Stacking. These classifier uses different techniques of decision tree known as decision stumps which are stacked one after another. Instead of Entropy as impurity metrics, Gini can be used. This provides another form of decision tree structures like: CART and C4.5. For dimensional reduction, PCA can be used as better option. Furthermore, advanced techniques like neural networks and their ensemble with different techniques like: Batch Gradient Descent, Batch Normalization, Regularization, ADAM can be done for improving evaluation metrics of the overall system. For system, there can be different features added like: Loan tracking, Loan cut-off and contract management of the loan. Different features like: monthly expense, savings and investments, foreclosure history and outstanding debt can be considered as data for increasing the robustness of an algorithm and overall application. Hence, project can be taken into directions based on the requirement.

REFERENCES

[1] N. Bansode, A. Verma, A. Sharma and V. Bhole, "Predicting Loan Approval Using ML," *International Research Journal of Modernization in Engineering and Technology,* pp. 375-382, May 2022.

[2] Y. Gao and L. Zhang, "sn, Research on Credit risk assessment of Small and Medium Sized Enterprises in Commercial banks," *Open Access Library Journal,* vol. 5, pp. 1-11, November 2018.

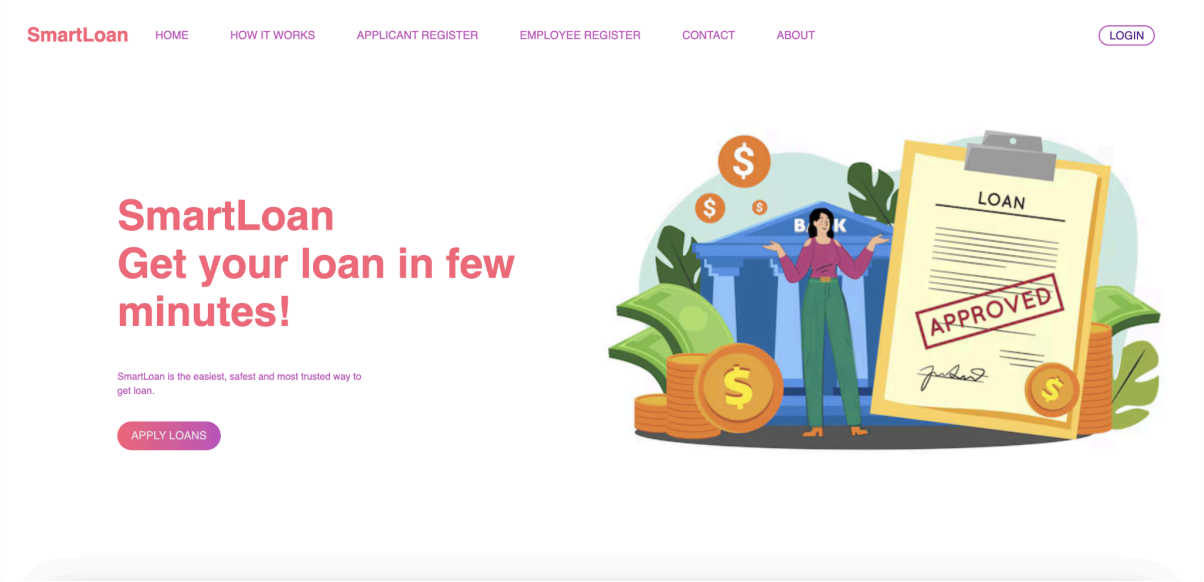
[3] A. Siddique, M. A. Khan and Z. Khan, "The effect of credit risk management and bank-specific factors on the financial performance of the south Asian banks," *Asian Journal of Accounting Research,* vol. 7, no. 2, 2021.

[4] Y. Hu and J. Su, "Research on Credit Risk Evaluation of Commercial Banks Based on Artificial Neural Network Model," in *International Conference of Information technology and quantitative management*, Beijing, 2022.

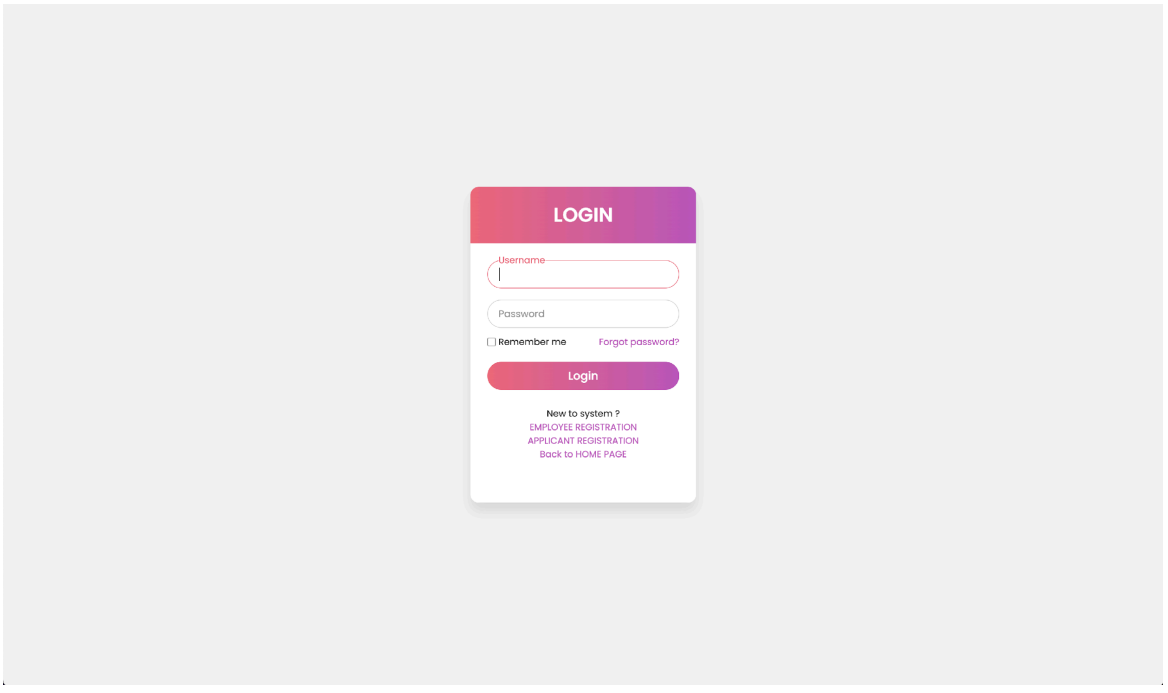
[5] J. Gayo and P. Tamayo, "Credit Risk Assessment Using Statistical and Machine Learning: Basic Methodology and Risk Modeling Applications," *Computation Studies stanford,* pp. 107-143, 2000.

APPENDIX

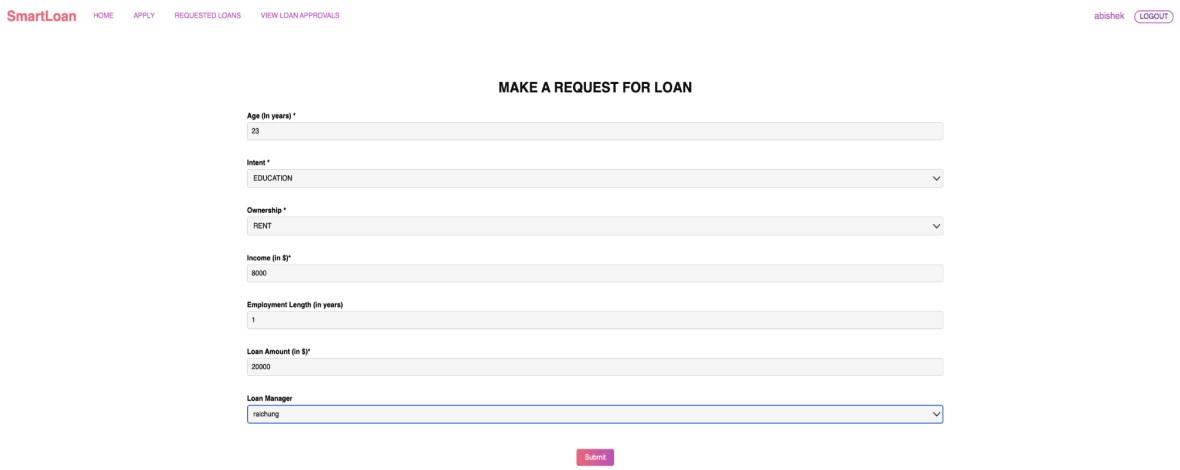
UI Screenshots



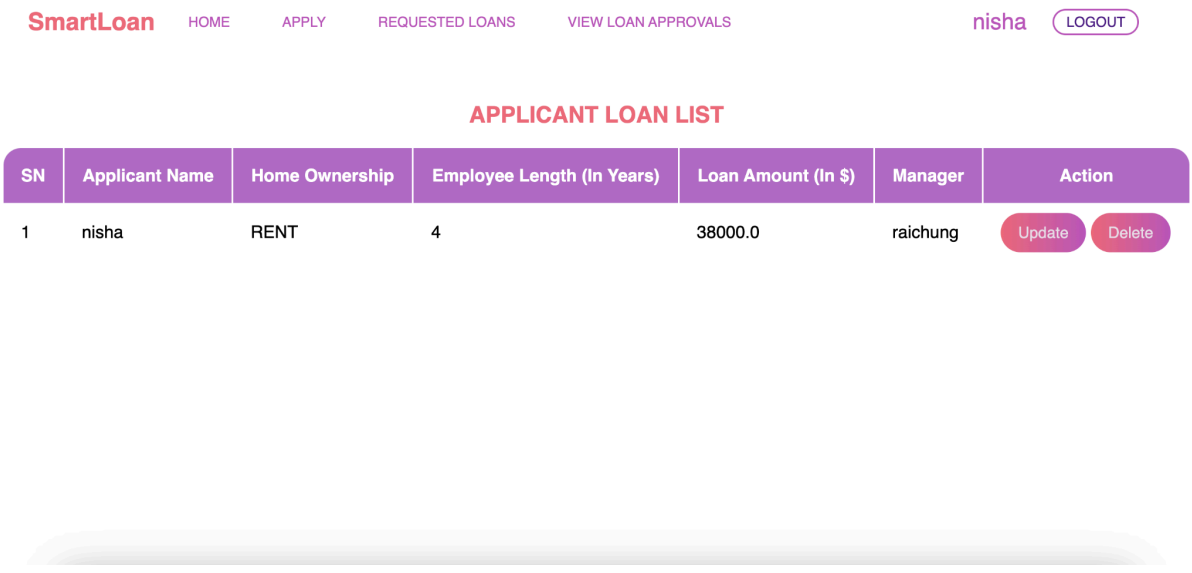
Appendix A Home Page



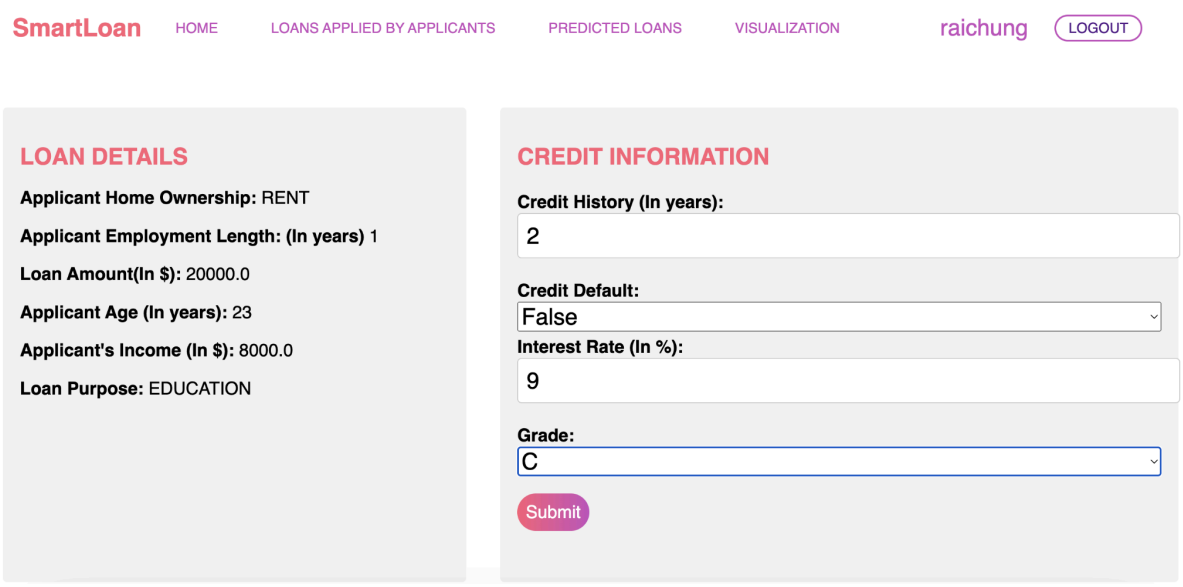
Appendix B Login Page



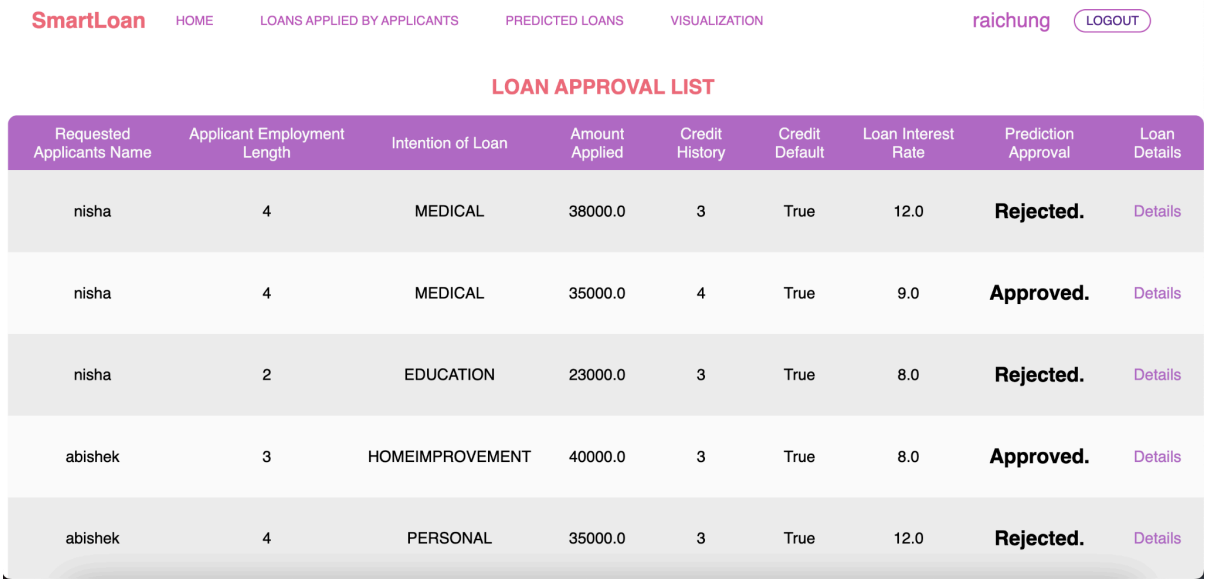
Appendix C Apply Loan Page



Appendix D Requested Loans page



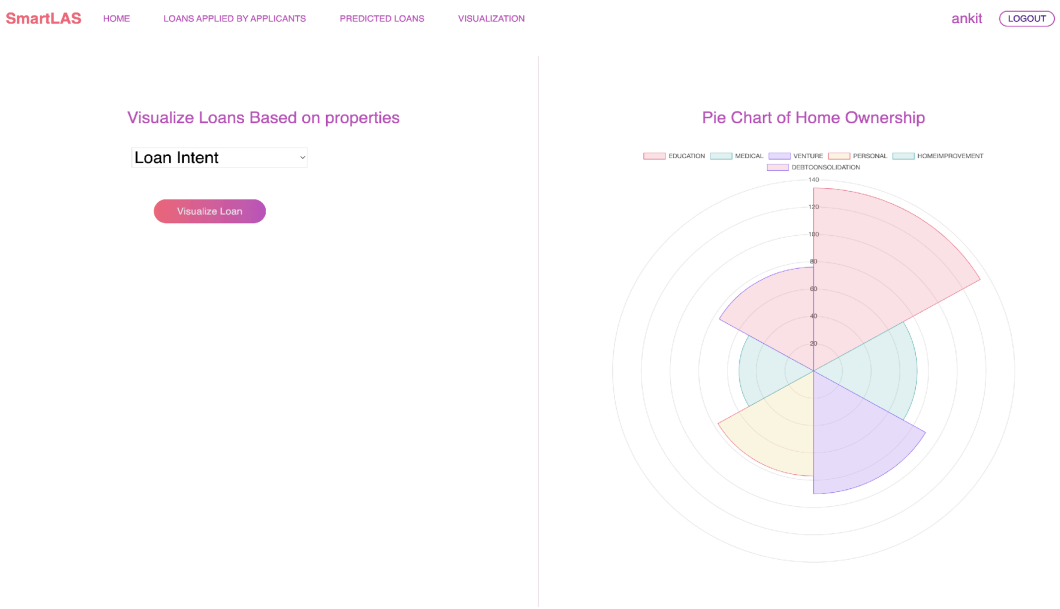
Appendix E Provide Loan Data by Employee page



Requested Applicants Name	Applicant Employment Length	Intention of Loan	Amount Applied	Credit History	Credit Default	Loan Interest Rate	Prediction Approval	Loan Details
nisha	4	MEDICAL	38000.0	3	True	12.0	**Rejected.**	Details
nisha	4	MEDICAL	35000.0	4	True	9.0	**Approved.**	Details
nisha	2	EDUCATION	23000.0	3	True	8.0	**Rejected.**	Details
abishek	3	HOMEIMPROVEMENT	40000.0	3	True	8.0	**Approved.**	Details
abishek	4	PERSONAL	35000.0	3	True	12.0	**Rejected.**	Details

Appendix F Loan Approval Prediction List page

Appendix G Loan Prediction Visualization



Appendix H Loan Intent Visualization

Source Code:

Here is the source code of implementation of decision tree and random forest algorithm:

Decision Node:

```
class Node:
        def __init__(self, feature = None, thresh=None, left=None, right=None, information_gain=None,value=None):
    self.feature = feature
    self.thresh = thresh
    self.left = left
    self.right = right
    self.information_gain = information_gain
    self.value = value


  def print_attributes(self):
    '''
    This is the method for printing the attributes of the given node.
    '''
    print("Feature to be split:"+(self.feature))
    print("Threshold of split:"+(self.thresh))
    print("Left child:"+(self.left))
    print("Right child:"+(self.right))
    print("Information Gain:"+(self.information_gain))
    print("Value at current node:"+(self.value))
```

Decision Tree:

```
from collections import Counter
```

```
from .node import Node
import numpy as np
class DecisionTree:
    def __init__(self, list_features, minimum_samples_to_split=2, max_depth=5):
        self.minimum_samples_to_split = minimum_samples_to_split
        self.max_depth = max_depth
        self.list_features = list_features
        # self.predict_level_check = 0
    @staticmethod
    def _entropy(data):
        count = np.bincount(np.array(data, dtype=np.int64))
        # probabilities of the all value in the class label
        prob = count/len(data)
        # initializing entropy
        entropy = 0
        # calculating entropy for all values
        for i in prob:
            if i>0:
                entropy += i * np.log2(i)
        return -(entropy)

    def _information_gain(self,parent_node, left_child_node, right_child_node):
        # Calculating number of probabilities of left and right child
        left_childs = len(left_child_node)/len(parent_node)
        right_childs = len(right_child_node)/len(parent_node)
        # Calculating entropy for left, right and parent node
        parent_entropy = self._entropy(parent_node)
        left_entropy = self._entropy(left_child_node)
        right_entropy = self._entropy(right_child_node)
```

```
        # Based on previous calculation, calculation of information gain
        information_gain = parent_entropy
-((left_entropy*left_childs)+(right_childs)*right_entropy)
        return information_gain


    def _calculate_best_split(self,features,label):
        # Initializing the values
        best_split = {}
        best_information_gain = -1
        (_,columns) = features.shape
        # print(columns)
        # Calculating best split
        for i in range(columns):
            # Selecting specific input feature column.
            x_current = features[:,i]
            for threshold in np.unique(x_current):
                # Creating  dataset by concatenating X and y.
                dataset = np.concatenate((features, label.reshape(1, -1).T), axis=1)
                # Splitting dataset into two halfs (left and right) based on rows.
                dataset_left = np.array([row for row in dataset if row[i] <= threshold])
                dataset_right = np.array([row for row in dataset if row[i] > threshold])
                # Selecting the best information gain.
                if (len(dataset_left) > 0 ) and (len(dataset_right) > 0):
                    y = dataset[:,-1]
                    y_left = dataset_left[:,-1]
                    y_right = dataset_right[:,-1]
                    # Calculating information gain.
                    information_gain = self._information_gain(y,y_left,y_right)
                    if (information_gain > best_information_gain):
                        best_split = {
```

```
                    "feature_index":i,
                    "threshold":threshold,
                    "left":dataset_left,
                    "right":dataset_right,
                    "information_gain":information_gain
                }
                best_information_gain = information_gain
        # print("Column Split:{0}".format(best_split["feature_index"]))
        print("Splitted_column Name:{0}".format(self.list_features[best_split['feature_index']]))
        return best_split


    def _build_tree(self, X,y, depth =0):
        num_rows, num_cols = X.shape
        print("--------------------------------------------------")
        print("At Level {0}:".format(depth))
        print("Number of instances of X: {0}".format(num_rows))
        print("Number of columns to split in X: {0}".format(num_cols))
        print("--------------------------------------------------")
        # condition 1 checks whether there is number of rows less than minimum samples to
split.
        condition_1 = (num_rows >=self.minimum_samples_to_split)
        # condition 2 checks whether the current depth is less than or equal to max depth defined
by the user.
        condition_2 = (depth<=self.max_depth)
        # checking both condition to build the tree.
        if condition_1 and condition_2:
            # Selecting the best split of the current depth.
            splitted_data = self._calculate_best_split(X, y)
            # Checking whether the best split given by the data is pure or not.
            if splitted_data['information_gain'] > 0:
```

```
            # Using recursion for getting left and right child to the current depth of the tree.
            # Left child split
            new_depth = depth+1
            print("Left Split to level:{0}".format(new_depth))
            X_left = splitted_data['left'][:,:-1]
            y_left = splitted_data['left'][:,-1]
            left_child = self._build_tree(X_left, y_left, new_depth)
            # right child split
            print("Right Split to level:{0}".format(new_depth))
            X_right = splitted_data['right'][:,:-1]
            y_right = splitted_data['right'][:,-1]
            right_child = self._build_tree(X_right, y_right, new_depth)
            # After calculating returning the data to the previous itreation of recursion.
            return node.Node(
                feature= splitted_data['feature_index'],
                thresh= splitted_data['threshold'],
                left = left_child,
                right = right_child,
                information_gain = splitted_data['information_gain']
            )
        # Returning the most common target value for the leaf node.
        return node.Node(value= Counter(y).most_common(1)[0][0])

    def train_model(self,X,y):
        print("---------------------------------------------------")
        print("Training Process Started.")
        self.root = self._build_tree(X, y)

    def _predict(self,x,tree):
```

```
        if tree.value != None:
            return tree.value
        feature = x[tree.feature]
        # print("Tree Feature :{0}".format(tree.feature))
        # go to left
        if feature <= tree.thresh:
            # self.predict_level_check+=1
            print("Left Split:{0}<={1} ".format(self.list_features[tree.feature], tree.thresh))
            return self._predict(x=x, tree = tree.left)
        if feature > tree.thresh:
            # self.predict_level_check+=1
            print("Right Split:{0}>{1} ".format(self.list_features[tree.feature], tree.thresh))
            return self._predict(x=x, tree=tree.right)

    def predict(self, X, list_features):
        self.list_features=list_features
        return [self._predict(x, self.root) for x in X]
```

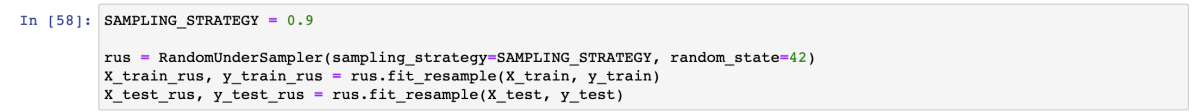
Random Forest:

```
import numpy as np
from collections import Counter
from .decision_tree import DecisionTree
class RandomForest:
    def __init__(self,list_features,num_trees, minimum_samples_to_split=2,max_depth=5):
        self.num_trees = num_trees
        self.minimum_samples_to_split = minimum_samples_to_split
        self.max_depth = max_depth
        self.trees = []
        self.list_features = list_features
```

```
@staticmethod
def __sample(X,y):
    n_rows, n_cols = X.shape
    # Sampling the dataset with replacements
    sample = np.random.choice(a=n_rows,size=n_rows, replace=True)
    samples_x = X[sample]
    samples_y = y[sample]
    return samples_x, samples_y
def train_model(self,X,y):
    i = 0
    if len(self.trees)>0:
        self.trees = []
    tree_built = 0
    while tree_built < self.num_trees:
        print("-------------------------------------------------")
        print("Itreation: {0}".format(i))
        tree = decision_tree.DecisionTree(
                list_features=self.list_features,
                minimum_samples_to_split= self.minimum_samples_to_split,
                max_depth= self.max_depth
            )
        sample_x, sample_y = self.__sample(X, y)
        tree.train_model(sample_x, sample_y)
        self.trees.append(tree)
        tree_built+=1
        i+=1
def predict(self,X, list_features):
    self.list_features = list_features
    labels = []
```

```
counter= 0
for tree in self.trees:
    counter+=1
    print("----------------------------")
    print("Tree:{0}".format(counter))
    labels.append(tree.predict(X,list_features))
labels = np.swapaxes(a=labels, axis1=0, axis2=1)
predictions = []
for preds in labels:
    counter = Counter(preds)
    predictions.append(counter.most_common(1)[0][0])
return predictions
```

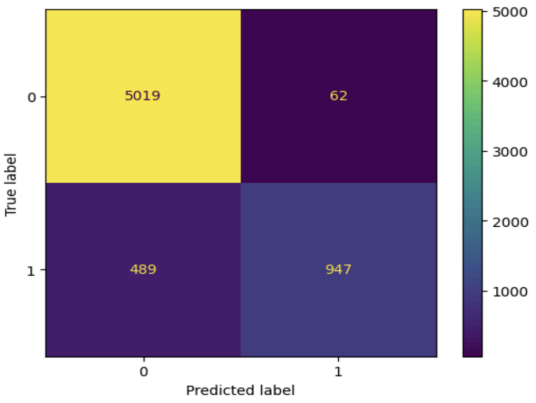
Notebook Screenshots



```
In [60]: # decision tree- undersampling
         dtree_rus = DecisionTree(list_features=X.columns)

In [61]: X_train_rus = X_train_rus.to_numpy()
         y_train_rus = y_train_rus.to_numpy()
         X_test_rus = X_test_rus.to_numpy()
         y_test_rus = y_test_rus.to_numpy()
         dtree_rus.train_model(X_train_rus,y_train_rus)

         ----------------------------------------------------
         Training Process Started.
         ----------------------------------------------------
         At Level 0:
         Number of instances of X: 11936
         Number of columns to split in X: 25
         ----------------------------------------------------
         Splitted_column Name:loan_percent_income
         Left Split to level:1
         ----------------------------------------------------
         At Level 1:
         Number of instances of X: 9525
         Number of columns to split in X: 25
         ----------------------------------------------------
         Splitted_column Name:loan_int_rate
         Left Split to level:2
         ----------------------------------------------------
         At Level 2:
         Number of instances of X: 7180

In [62]: y_pred_dtree_rus = dtree_rus.predict(X_test_rus,list_features = X.columns)

         Left Split:loan_percent_income<=1.2155493930839336
         Right Split:loan_int_rate>0.9671647617118444
         Left Split:loan_grade_C<=0.0
         Left Split:loan_intent_DEBTCONSOLIDATION<=0.0
         Right Split:loan_intent_MEDICAL>0.0
         Left Split:person_home_ownership_OWN<=0.0
         Left Split:loan_percent_income<=1.2155493930839336
         Left Split:loan_int_rate<=0.9671647617118444
         Right Split:person_income>-0.7434855862215002
         Left Split:loan_grade_D<=0.0
         Right Split:person_income>-0.501674102420751
         Left Split:loan_grade_A<=0.0
         Left Split:loan_percent_income<=1.2155493930839336
         Left Split:loan_int_rate<=0.9671647617118444
         Right Split:person_income>-0.7434855862215002
         Left Split:loan_grade_D<=0.0
         Right Split:person_income>-0.501674102420751
         Left Split:loan_grade_A<=0.0
         Left Split:loan_percent_income<=1.2155493930839336

In [67]: y_test_rus.shape
Out[67]: (3069,)

In [69]: y_pred_dtree_rus = np.array(y_pred_dtree_rus)

In [70]: print(y_test_rus.shape,y_pred_dtree_rus.shape)
         (3069,) (3069,)

In [79]: model_evaluation(y_test_rus, y_pred_dtree_rus)

Imbalanced Accuracy:0.8429455848810687
Balanced Accuracy:0.8367186069389024
Classification Report:              precision    recall  f1-score   support

           0       0.79      0.96      0.86      1615
           1       0.94      0.72      0.81      1454

    accuracy                           0.84      3069
   macro avg       0.86      0.84      0.84      3069
weighted avg       0.86      0.84      0.84      3069

AUC score:0.8367186069389024
(3069,) (3069,)
```

Appendix I Random Undersampling with their results

```
In [96]: from random_forest import RandomForest

In [98]: rf = RandomForest(list_features=X.columns, num_trees=10,minimum_samples_to_split=100)

In [99]: rf.train_model(new_x_train,new_y_train)

         ----------------------------------------------------
         Itreation: 0
         ----------------------------------------------------
         Training Process Started.
         ----------------------------------------------------
         At Level 0:
         Number of instances of X: 26064
         Number of columns to split in X: 25
         ----------------------------------------------------
         Splitted_column Name:loan_percent_income
         Left Split to level:1
         ----------------------------------------------------
         At Level 1:
         Number of instances of X: 23080
         Number of columns to split in X: 25
         ----------------------------------------------------
         Splitted_column Name:loan_int_rate
         Left Split to level:2
         ----------------------------------------------------

In [103]: y_predicted_rf = rf.predict(new_x_test,X.columns)

          Right Split:loan_int_rate>-0.6845939659191759
          Left Split:loan_percent_income<=1.2155493930839336
          Left Split:loan_int_rate<=0.9639196561565773
          Right Split:person_income>-0.7434855862215002
          Left Split:loan_grade_D<=0.0
          Right Split:person_income>-0.11407875368648351
          Right Split:loan_int_rate>-0.6845939659191759
          Left Split:loan_percent_income<=1.2155493930839336
          Left Split:loan_int_rate<=0.9639196561565773
          Right Split:person_income>-0.7434855862215002
          Left Split:loan_grade_D<=0.0
          Right Split:person_income>-0.11407875368648351
          Right Split:loan_int_rate>-0.6845939659191759
          Left Split:loan_percent_income<=1.2155493930839336
          Left Split:loan_int_rate<=0.9639196561565773
          Right Split:person_income>-0.7434855862215002
          Left Split:loan_grade_D<=0.0
          Left Split:person_income<=-0.11407875368648351
          Left Split:loan_intent_HOMEIMPROVEMENT<=0.0
          Left Split:loan_percent_income<=1.2155493930839336

In [104]: y_predicted_rf = np.array(y_predicted_rf)

In [106]: model_evaluation(new_y_test,y_predicted_rf)

          Imbalanced Accuracy:0.9154518950437318
          Balanced Accuracy:0.8236342148558259
          Classification Report:              precision    recall  f1-score   support

                     0       0.91      0.99      0.95      5081
                     1       0.94      0.66      0.77      1436


           accuracy                           0.92      6517
          macro avg       0.92      0.82      0.86      6517
       weighted avg       0.92      0.92      0.91      6517

       AUC score:0.8236342148558259
       (6517,) (6517,)
```



Appendix II Random Forest Hyper[OBJ]parameter Tuned model Prediction