**Tribhuvan University**

**Amrit Campus**



**Final Year Project Report**

**on**

**LOAN APPROVAL PREDICTION SYSTEM**

**[CSC 412]**

**Under the supervision of**

**Mr. Dhirendra Kumar Yadav**

**Coordinator**

**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](mailto: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](mailto: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.

| ------------------------------------------------  **Mr. 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 Mr. 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**](#_1fob9te) **i**

[**ABSTRACT**](#_3znysh7) **ii**

[**LIST OF FIGURES**](#_2et92p0) **iii**

[**LIST OF TABLES**](#_tyjcwt) **iv**

[**LIST OF ABBREVIATIONS**](#_3dy6vkm) **v**

[**CHAPTER 1 : INTRODUCTION 1**](#_1t3h5sf)

[**1.1 Introduction 1**](#_4d34og8)

[**1.2 Problem Statement 1**](#_2s8eyo1)

[**1.3 Objectives 2**](#_17dp8vu)

[**1.4 Scope and Limitations 2**](#_3rdcrjn)

[**1.5 Development Methodology 2**](#_26in1rg)

[**CHAPTER 2 : BACKGROUND STUDY AND LITERATURE REVIEW 4**](#_1ksv4uv)

[**2.1 Literature Review 4**](#_44sinio)

[**CHAPTER 3 : SYSTEM ANALYSIS 6**](#_2jxsxqh)

[**3.1 System Analysis 6**](#_z337ya)

[**3.1.1 Requirement Analysis 6**](#_3j2qqm3)

[**3.1.2 Feasibility Analysis 10**](#_qsh70q)

[**3.1.3 Analysis 11**](#_49x2ik5)

[**CHAPTER 4 : SYSTEM DESIGN 13**](#_147n2zr)

[**4.1 Design 13**](#_3o7alnk)

[**4.1.1 Class Diagram 13**](#_23ckvvd)

[**4.1.2 Activity Diagram 14**](#_32hioqz)

[**4.1.3 Sequence Diagram 15**](#_41mghml)

[**4.1.4 System Architecture 16**](#_vx1227)

[**4.2 Algorithm Details 17**](#_1v1yuxt)

[**4.2.1 Decision Node 18**](#_2u6wntf)

[**4.2.2 Impurity Metrics 18**](#_19c6y18)

[**4.2.3 ID3 algorithm 20**](#_28h4qwu)

[**4.2.4 Ensemble Learning 21**](#_nmf14n)

[**4.2.5 Bagging 22**](#_37m2jsg)

[**4.2.6 Voting classifier 22**](#_1mrcu09)

[**4.2.7 Random Forest Algorithm 22**](#_46r0co2)

[**CHAPTER 5 : IMPLEMENTATION AND TESTING 25**](#_111kx3o)

[**5.1 Implementation 25**](#_3l18frh)

[**5.1.1 Tools Used 25**](#_206ipza)

[5.1.2 Implementation Details 30](#)

[**5.1.3 Experiments Conducted 37**](#_3q5sasy)

[**5.2 Testing 40**](#_43ky6rz)

[**5.2.1 Test Cases for Unit Testing 40**](#_2iq8gzs)

[**5.2.2 Test Case for Integration Testing 50**](#_3vac5uf)

[**5.2.3 Test Cases for System Testing 51**](#_2afmg28)

[**5.3 Result Analysis 53**](#_pkwqa1)

[**CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATIONS 58**](#_48pi1tg)

[**6.1 Conclusion 58**](#_2nusc19)

[**6.2 Future Recommendations 58**](#_1302m92)

[**REFERENCES 59**](#_3mzq4wv)

[**APPENDIX 60**](#_2250f4o)

# LIST OF FIGURES

[Figure 1.1 Incremental Delivery Model 3](#_lnxbz9)

[Figure 3.1 Use case diagram of loan approval system 8](#_1y810tw)

[Figure 3.2 Gantt Chart of the project 13](#_1pxezwc)

[Figure 3.3 ER diagram of the project 14](#_2p2csry)

[Figure 4.1 Descriptive class diagram of the project 17](#_ihv636)

[Figure 4.2 Activity Diagram for Loan Request Process by Applicant 18](#_1hmsyys)

[Figure 4.3 Sequence diagram of View Loan Prediction Process 19](#_2grqrue)

[Figure 4.4 System Architecture 20](#_3fwokq0)

[Figure 4.5 Decision Tree 21](#_4f1mdlm)

[Figure 4.6 Plot between Entropy and Probability of class. 23](#_3tbugp1)

[Figure 4.7 Random Forest Classifier 27](#_2lwamvv)

[Figure 5.1 Algorithm Implementation Procedure 33](#_2dlolyb)

[Figure 5.2 Model pipeline of the system 39](#_1664s55)

[Figure 5.3 ROC Curve 62](#_39kk8xu)

# 

# 

# LIST OF TABLES

[Table 3.1 Use Case Description for Applicant Registration 9](#_4i7ojhp)

[Table 3.2 Use Case Description for Applicant Login 9](#_2xcytpi)

[Table 3.3 Use Case Description for Request Loan 10](#_1ci93xb)

[Table 3.4 Use Case Description for Provide Loan Data 10](#_3whwml4)

[Table 3.5 Use Case Description for View Loan Prediction 11](#_2bn6wsx)

[Table 3.6 Work Breakdown Structure 12](#_3as4poj)

[Table 5.1 Built-in python modules 31](#_4k668n3)

[Table 5.2 External Modules used in project 32](#_2zbgiuw)

[Table 5.3 Hardware configuration used in project 32](#_1egqt2p)

[Table 5.4 Data columns 34](#_sqyw64)

[Table 5.5 Dataset split size 36](#_3cqmetx)

[Table 5.6 One-hot encoding 37](#_1rvwp1q)

[Table 5.7 Label Encoding 38](#_4bvk7pj)

[Table 5.8 Baseline model metrics for training and test dataset 38](#_2r0uhxc)

[Table 5.9 Data Size for random data under sampling 40](#_25b2l0r)

[Table 5.10 Data Size for random data oversampling 41](#_kgcv8k)

[Table 5.11 Hyperparameters for Decision Tree 42](#_34g0dwd)

[Table 5.12 Hyperparameters for Random Forest 42](#_1jlao46)

[Table 5.13 Test Case for Applicant Register 43](#_xvir7l)

[Table 5.14 Test Case for Applicant Login 46](#_3hv69ve)

[Table 5.15 Test Case for Apply Loans 48](#_1x0gk37)

[Table 5.16 Test Case for Employee Registration 49](#_4h042r0)

[Table 5.17 Test Case for provide loan data. 53](#_2w5ecyt)

[Table 5.18 Test Case for loan prediction 54](#_1baon6m)

[Table 5.19 Test set results with different models 63](#_1opuj5n)

# LIST OF ABBREVIATIONS

| **SME** | Small and medium sized enterprise |
| --- | --- |
| **PCA** | Principal component analysis |
| **PL** | Performing loan |
| **NPL** | Non-performing loan |
| **CER** | Cost-efficiency ratio |
| **ALR** | Average lending rate |
| **LR** | Liquidity ratio |
| **FP** | Financial performance |
| **CSMAR** | China Stock Market & Accounting Research |
| **BP** | Back propagation |
| **KNN** | K-nearest neighbors |
| **CART** | Classification and regression tree |
| **KS** | Kolmogorov-Smirnov test |
| **NN** | Neural Network |
| **ER** | Entity Relation Diagram |
| **ID3** | Itreative Dichotomiser 3 |
| **UML** | Unified modeling language |
| **HTML** | Hypertext Markup Language |
| **CSS** | Cascading Style Sheet |
| **JS** | Javascript |
| **W3C** | World Wide Web Consortium |
| **MVC** | Model View Controller |
| **MVT** | Model View Template |
| **RDBMS** | Relational Database Management System |
| **IDE** | Integrated Development Environment |
| **SQL** | Structured Query Language |
| **CASE** | Computer Aided Software Engineering tools |
| **AUC** | Area Under the Curve |
| **ADAM** | Adaptive Moment Estimation |

# INTRODUCTION

## 1.1 Introduction

Loan approval is 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 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 employee provide past records of applicant for classifying risks associated with the loan. This system is web-based application and uses tree-based machine learning techniques for classifying the loan request. There are two different types of users associated with 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 significant challenge over managing the lent loan. There has been 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 system is not easily scalable for handling large amount of the data for this purpose.

Manual process of looking after credit risks are very time consuming which may result in loss of customers/applicants to the banks. Also in current scenario, there is very hard procedure for facilitating the effective risk managing 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 very 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 is 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.

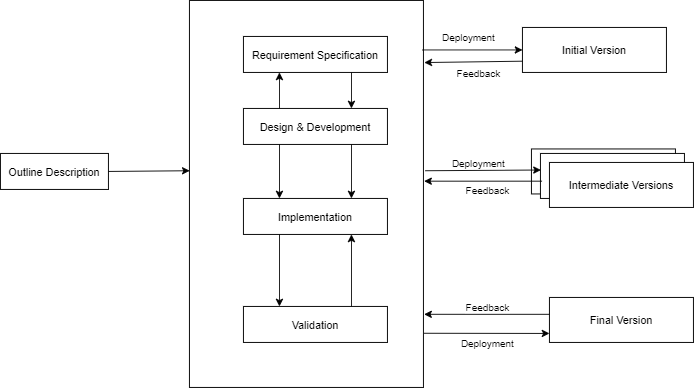
## 1.5 Development Methodology

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**

# 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, Verma et.al have provided that primary source of benefit for financial institution are through the interest of loan. Authors have also provided that loan recovery is 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 and Lidan Zhang in 2018 have 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 number of features PCA were conducted. Authors used Bayesian logistic regression model for properly analyzing the data, which resulted in accuracy of 93.4 percent in training sample and 90 percent in 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 and J. Galindo have 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.

# SYSTEM ANALYSIS

## 3.1 System Analysis

### 3.1.1 Requirement Analysis

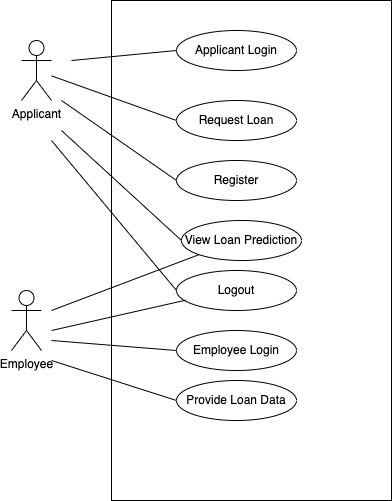
**3.1.1.1 Functional Requirements**

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

Following are the functional requirements of the project:

* Allow applicant to request loan.
* Allow employee to provide loan details.
* Allow both applicant and employee to view the classified loan status.
* Allow applicant to register to the system.
* Allow both applicants and employee to login and logout of 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 applicant is eligible for the loan. |
| **Pre-condition** | Loan request 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

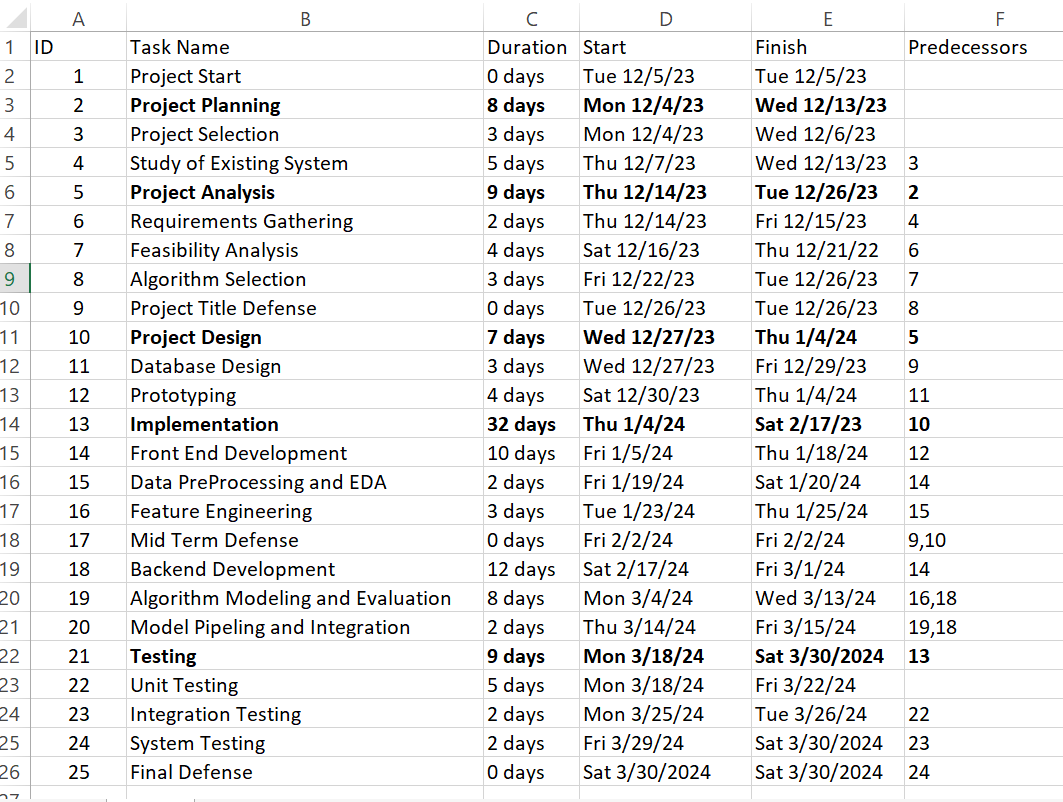
Feasibility study is the focused study that takes place early in the requirement engineering process. The aim of the feasibility study is to find out whether the system can be implemented or not. There are following feasibility study done in the project:

**3.1.2.1 Schedule Feasibility**

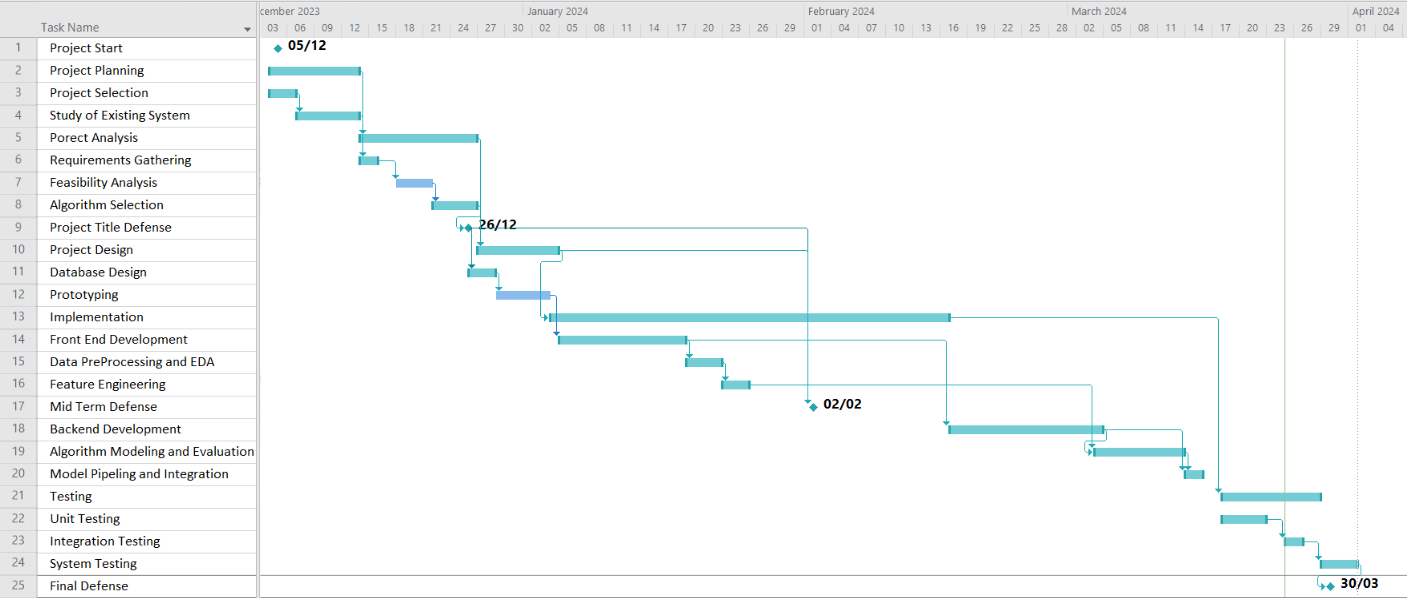
Schedule feasibility looks after the potential time frame of the completion of the project. It also looks after major activities and their time period or constraints involved. Through thorough analysis, this project is feasible in schedule and can be completed in the time frame.

Here is the work breakdown structure of the project:

**Table 3.6 Work Breakdown Structure**



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

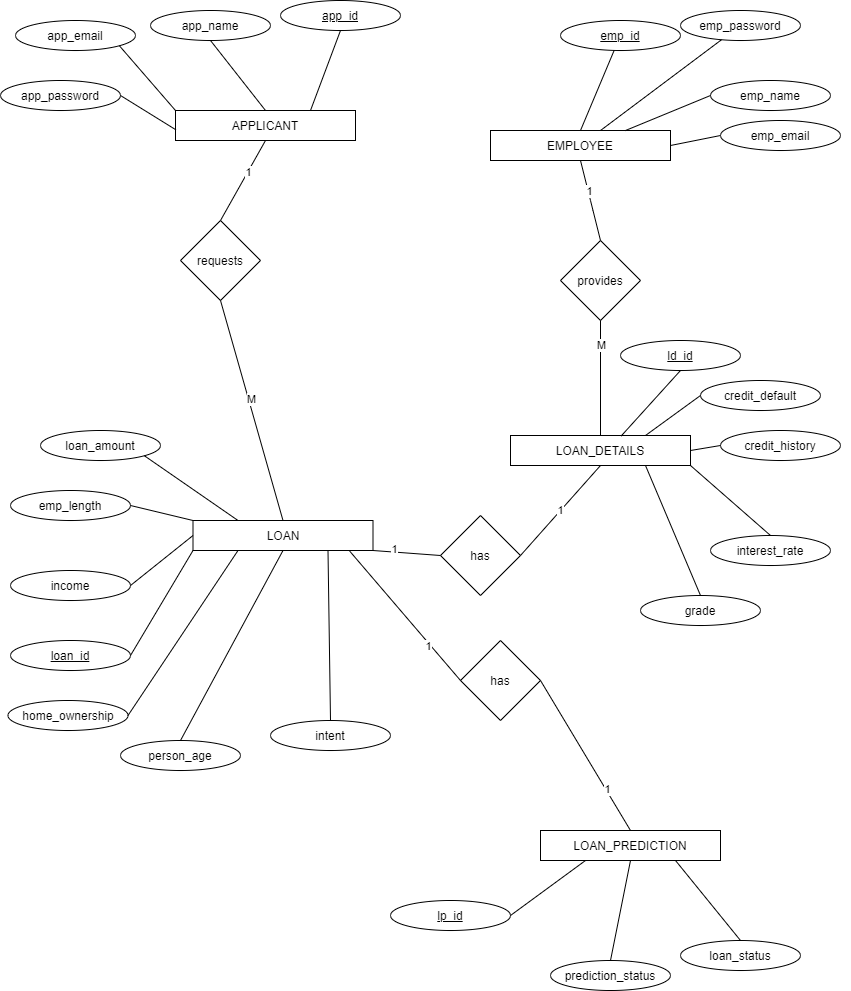
In analysis phase, requirements of the system are structured. Data modelling technique is used for structuring our requirements.

**3.1.3.1 Data Modeling**

Data Modeling is twhe process of preparing detailed model that captures overall system’s structure through the data. For data modeling, ER diagram is 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).

# SYSTEM DESIGN

## 4.1 Design

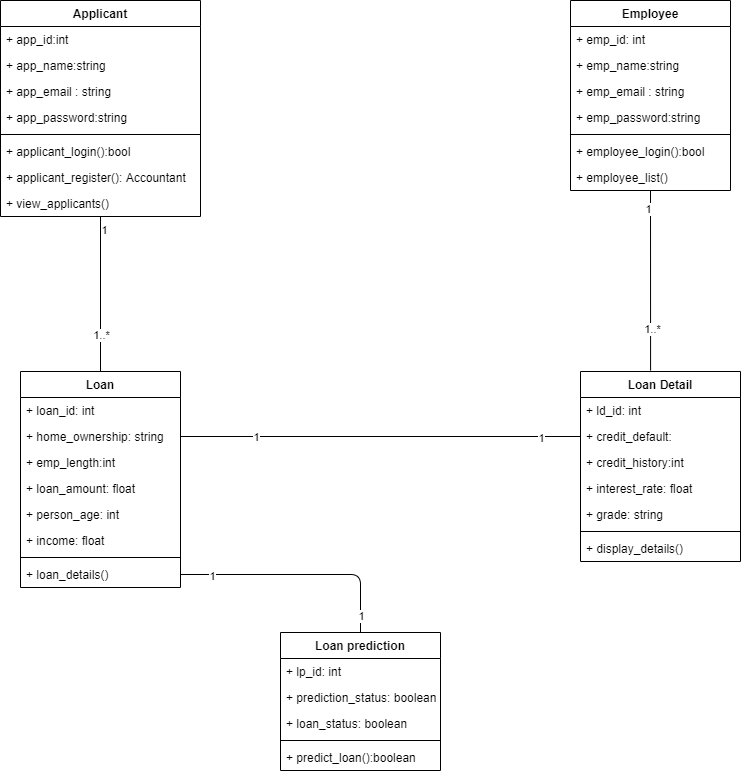
Design of the system requires careful planning and thinking while developing the system. Overall performance of the system is completely dependent upon the design of the system. In this project class diagram, activity diagram and sequence diagram are used for showing the overall workflow of the system.

### 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. Employee has staff status in the system which differentiates with applicant. Both user has 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 employee. 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:

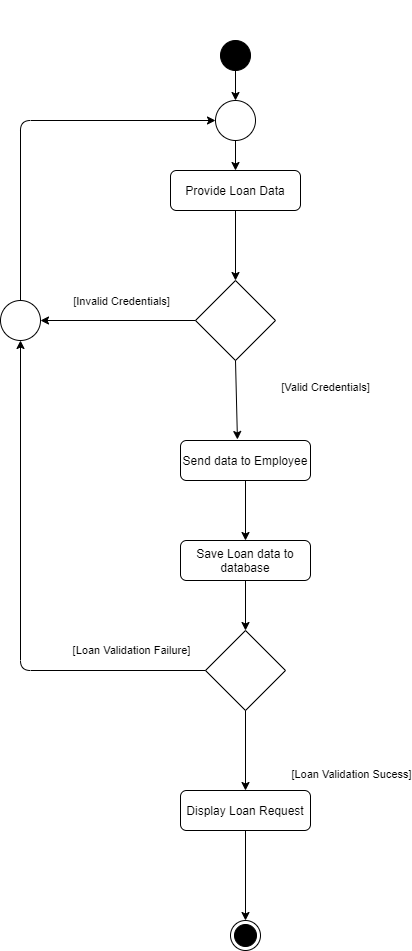


**Figure 4.1 Descriptive class diagram of the project**

### 4.1.2 Activity Diagram

Activity Diagram are the diagram in UML which is 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:

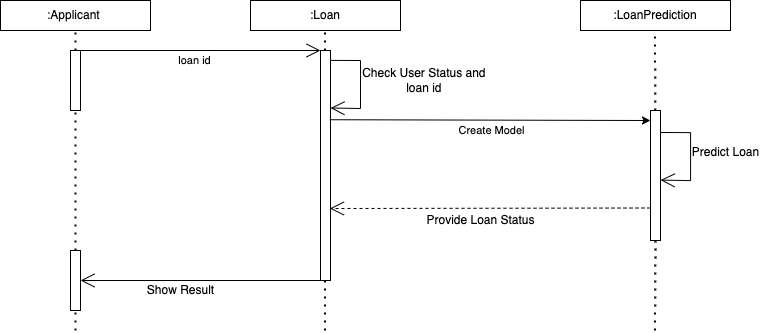


**Figure 4.2 Activity Diagram for Loan Request Process by Applicant**

### 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**

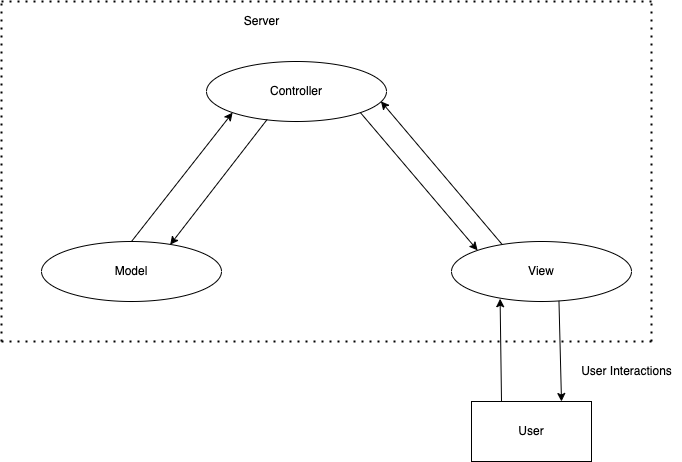
In the given diagram, there are 3 different objects involved namely: Applicant, Loan and Loan Prediction. Overall process is initiated when Applicant clicks over the given loan for prediction. Loan id is passed as message to Loan object along with user’s staff status. Loan object now checks loan id for 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 application as server 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:

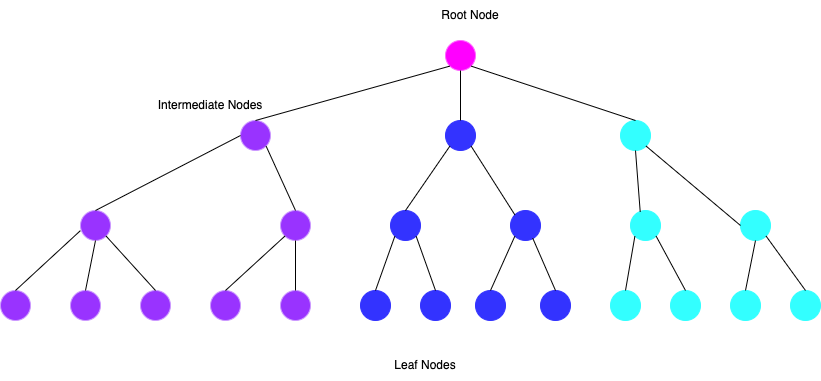


**Figure 4.4 System Architecture**

## 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 splits data into two or more subgroups based on value of 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 looks after measure of purity at each feature columns 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 probability of given set of values in columns and overall logarithmic measure for calculating impurity over the data columns.

Entropy is defined as:

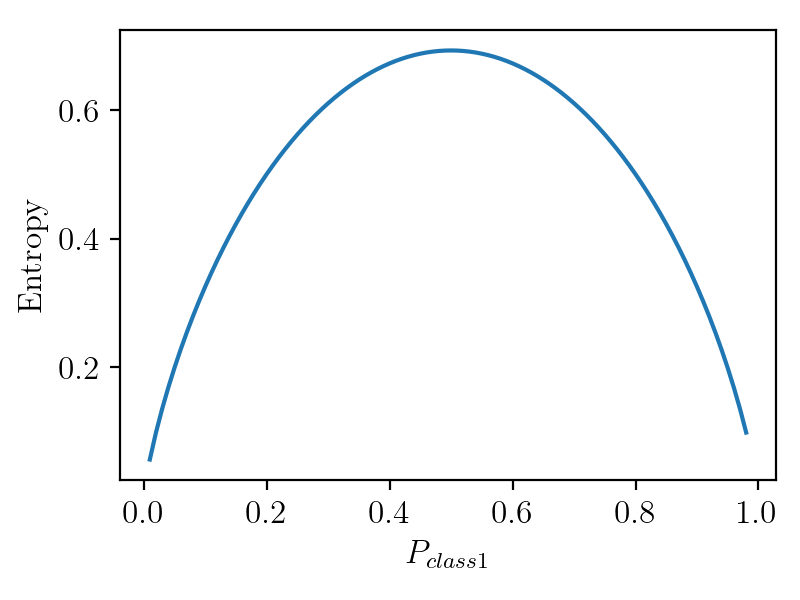
Where, is feature name indices,

is total number of columns in the dataset,

is 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 given data column are of 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 variable are selected based on their occurrence. Information gain not only considers quality of splitted data items but also consider total size of each data split and size of outcomes for each data split.

Information gain is defined as:

Where, is a column name.

is probability of given target outcomes.

is 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 greedy approach for building 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 decisions 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 (), Maximum Depth (), 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 () 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 () i.e.

Step 2.2: For all input features ()

Step 2.2.1: Calculate entropy of given column () with different set of values. i.e.

Step 2.2.2: Calculate information gain of through the help of entropy i.e.

Step 2.3 Compare information gain of all columns.

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

Step 3: Divide given with 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 dataset with smaller than threshold value. Right node being dataset with larger than threshold value.

Step 5: Increase by 1. Check with no. of datasets, Check with . 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 in 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-model for combining prediction of individual models. Bagging simply splits the data and combines the result based on the nature of result. Boosting and stacking are done for 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 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 which the overall data into multiple sets of data through random selections of samples with replacement.

Bagging involves in 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 combination of two or models and uses bagging for the splitting data and provides the result through use of voting mechanism provided by an individual models. There are two types of voting mechanisms conducted in voting classifiers. They are:

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

**Soft Voting**: In this voting classifier, 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 result to real-world data. Here hard voting is used for voting mechanism for selecting the best prediction over other sets of output predictions.

Here is an algorithm for random forest algorithm:

Input: Dataset (), Maximum Depth (), Minimum Samples to split (), Number of trees ()

Step 1: Randomly sample into , , , …, and create different trees into ,,...

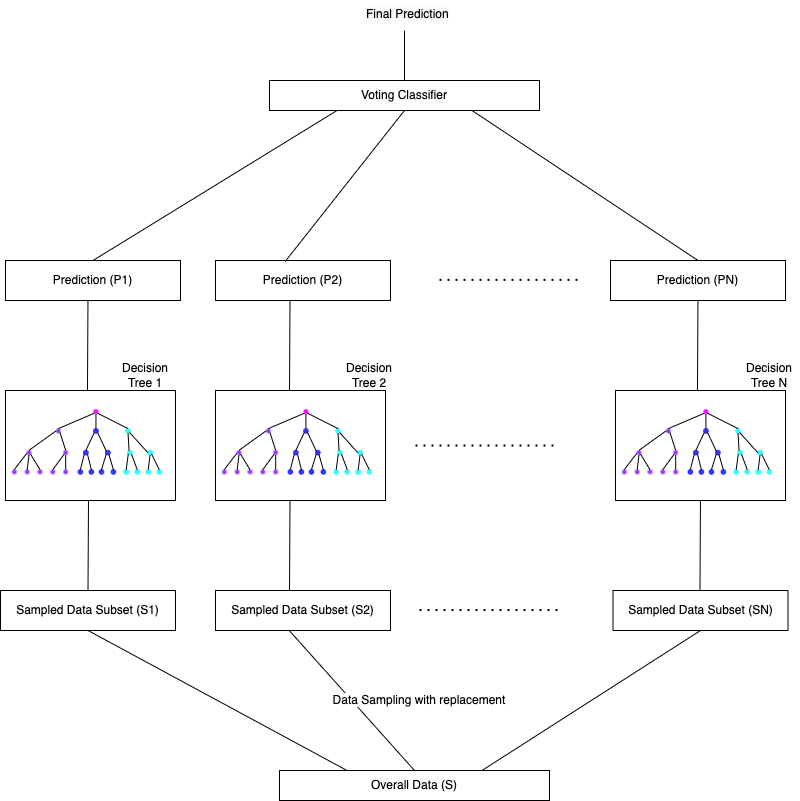
Step 2: For each decision tree , use ID3 algorithm for given bootstrapped sample with as maximum samples to split and as maximum depth of the decision tree.

Step 3: For given set of input data , predict the output () with the help of trees ,,...

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

i.e.,

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



**Figure 4.7 Random Forest Classifier**

# IMPLEMENTATION AND TESTING

## 5.1 Implementation

System implementation involves in development of 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 helps in overall implementation of the project. Different tools were used for development of system 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 user and 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 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 server asynchronously.

**Back-end tools**

Back-end tools are tools used for providing proper functionality of system when 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. |

1. **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:

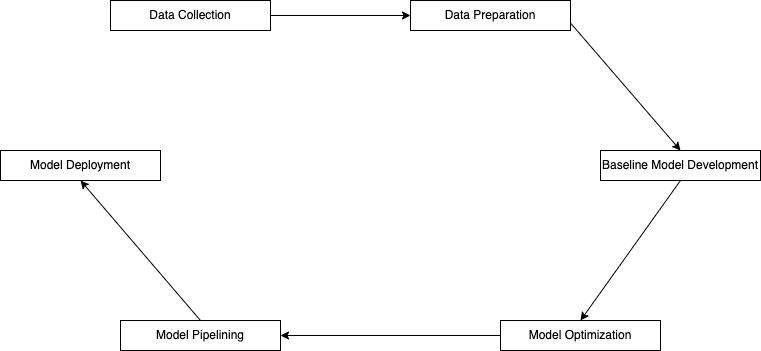
**Table 5.3 Hardware configuration used in project**

| Machines | Machine I | Machine II |
| --- | --- | --- |
| Operating System | Windows 11 | Windows 10 |
| Processor | AMD Ryzen 5 [5500U@2.10](mailto: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.

1. **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) |

1. **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.

1. **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:

Here,

is the scaled output of the data.

is the original data that is passed for scaling.

is the mean of the data column.

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.

1. **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.

1. **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

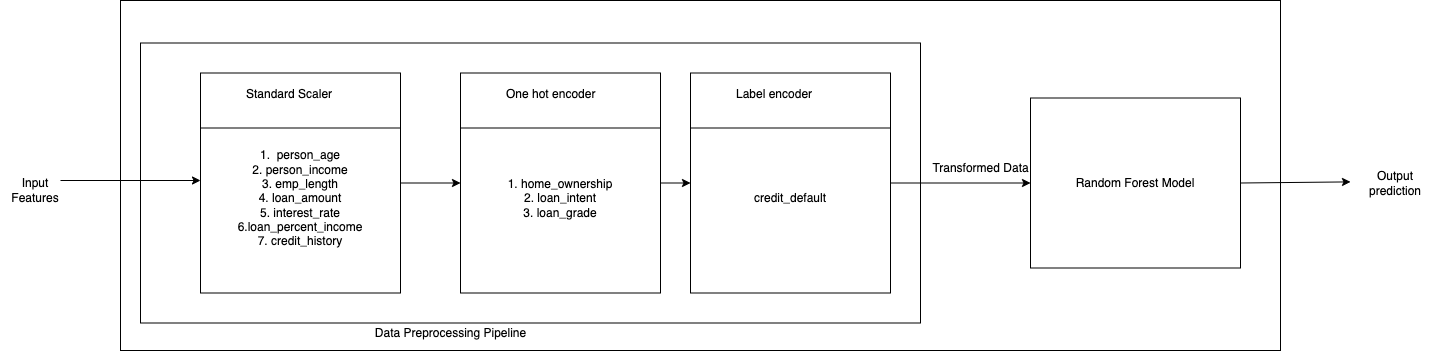
For random forest algorithm, three different parameters were used. They are:

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 used integrated with the web application. The hyperparameters selected while optimizing model 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 project is conducted with sampling strategy of 90% was used. 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"  4. Provide email: abishek44@gmail.com  -Submit Button clicked | Applicant is redirected to Login Page. | Applicant is redirected to Login Page. | | Pass |

**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 access login page. | User is navigated to login page | | Pass |
| 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 columns were conducted. | | Pass |
| 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-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 | Load Data | Provide The loan Data through selection specific Employee | Load Detail of Applicant should pass to selected Employee. | Loan detail are available on employees "Applicant Loan list" and in "Requested Loans" with some option. | | Pass |
| 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, delete, etc. work properly | As excepted | | Pass |
| 6 | [Usability Testing](https://www.softwaretestinghelp.com/usability-testing-guide/) | Provide proper information to user about all forms and conditions. | Make sure that the system is easy to use, learn and operate | As excepted "Tested by different new users and review " | | Pass |
| 7 | 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 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:

Here,

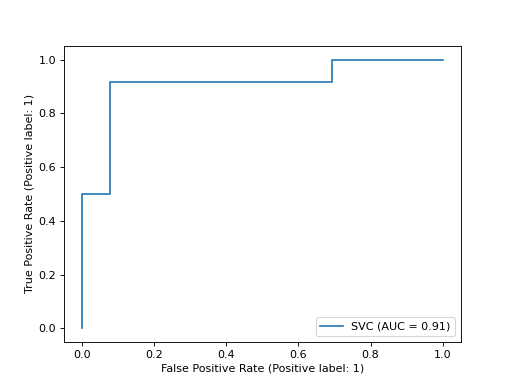
And,

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:

where,

**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:

where,

**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 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.

# 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

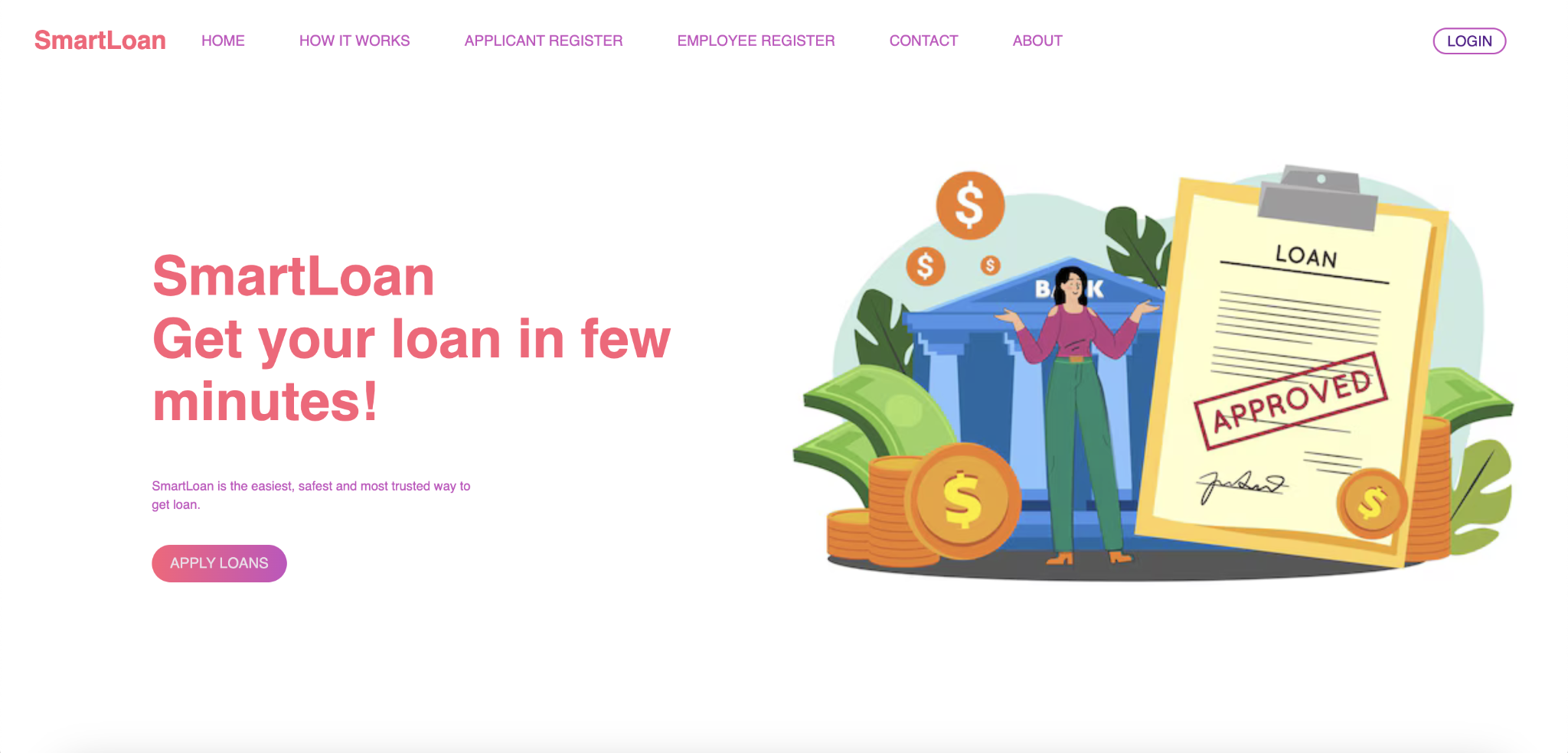
For even better results, few things can be added in the future work. This system specially uses bootstrapping but there are other forms of ensemble learning such as: Boosting and Stacking. These classifier uses different techniques of decision tree known as decision stumps which are stacked one after 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

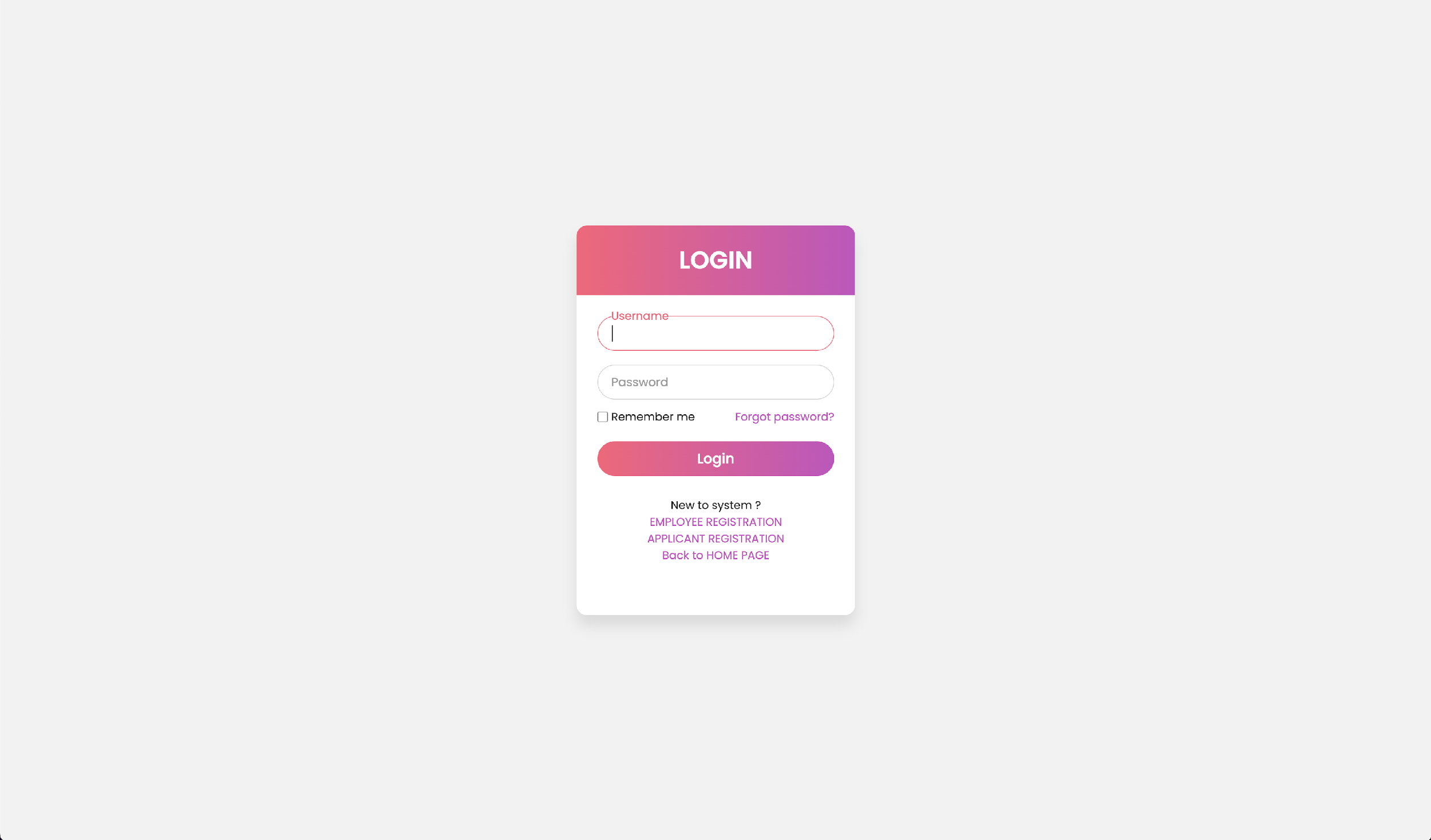
| [1] | N. Bansode, A. Verma, A. Sharma and V. Bhole, "Predicting Loan Approval Using ML," *International Research Journal of Modernization in Engineering and Technology,* pp. 375-382, May 2022. |
| --- | --- |
| [2] | Y. Gao and L. Zhang, "sn, Research on Credit risk assessment of Small and Medium Sized 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

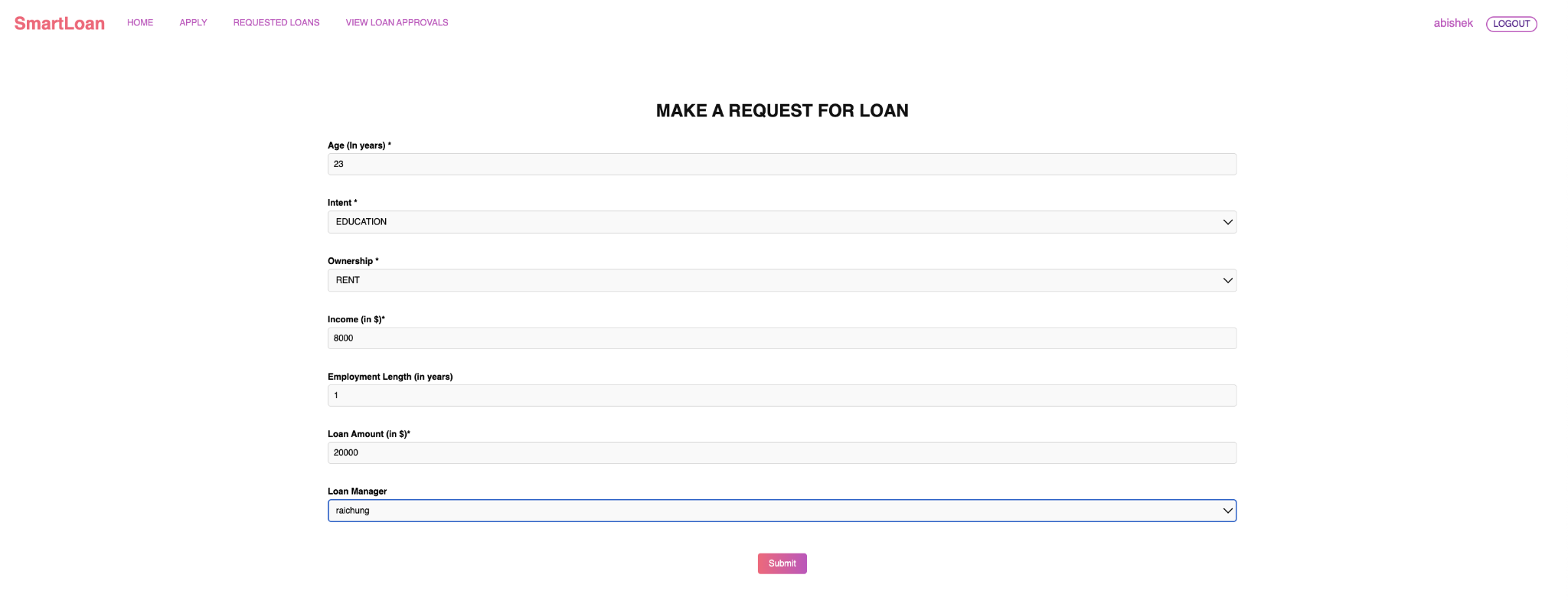
**UI Screenshots**



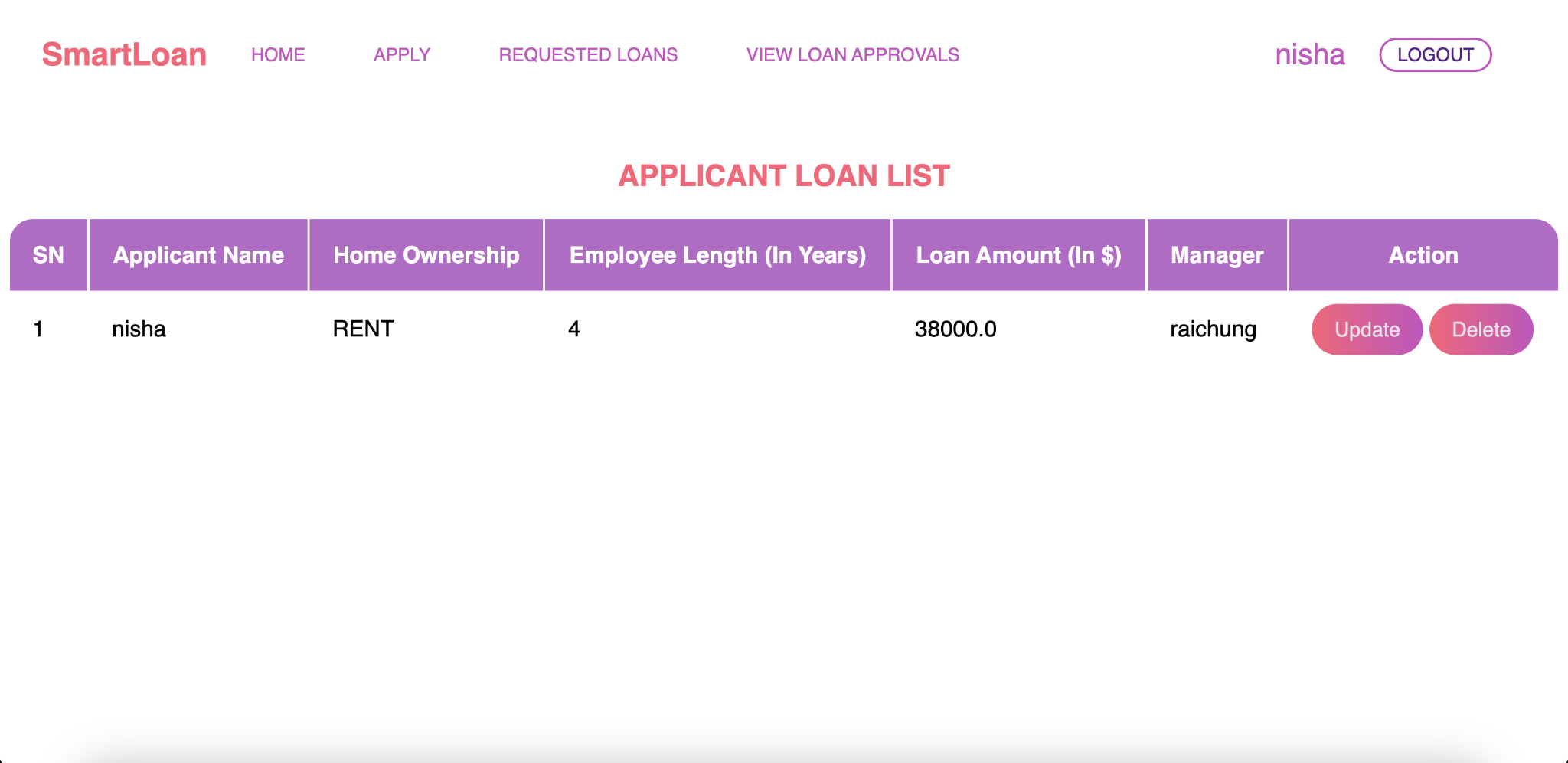
**Appendix A Home Page**



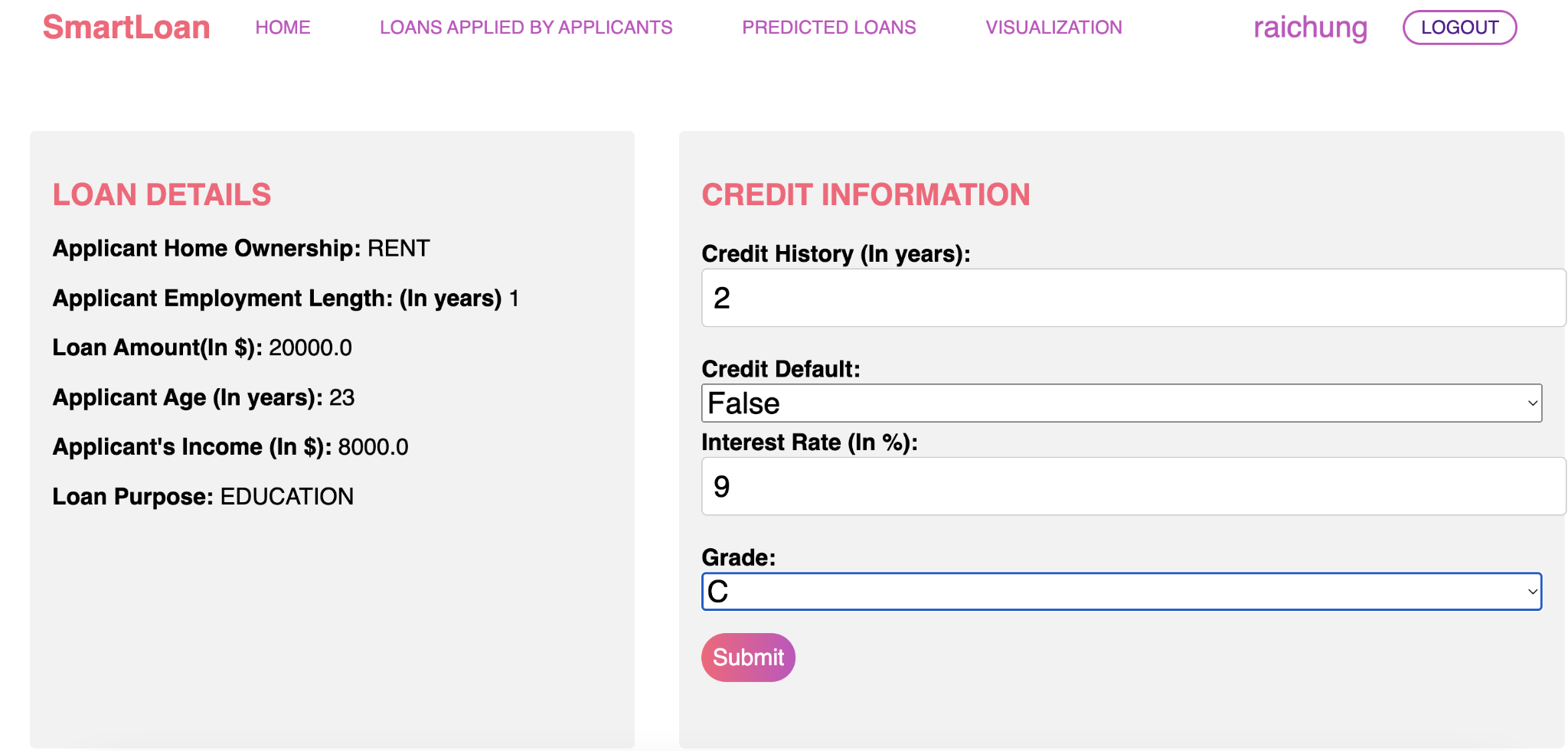
**Appendix B Login Page**



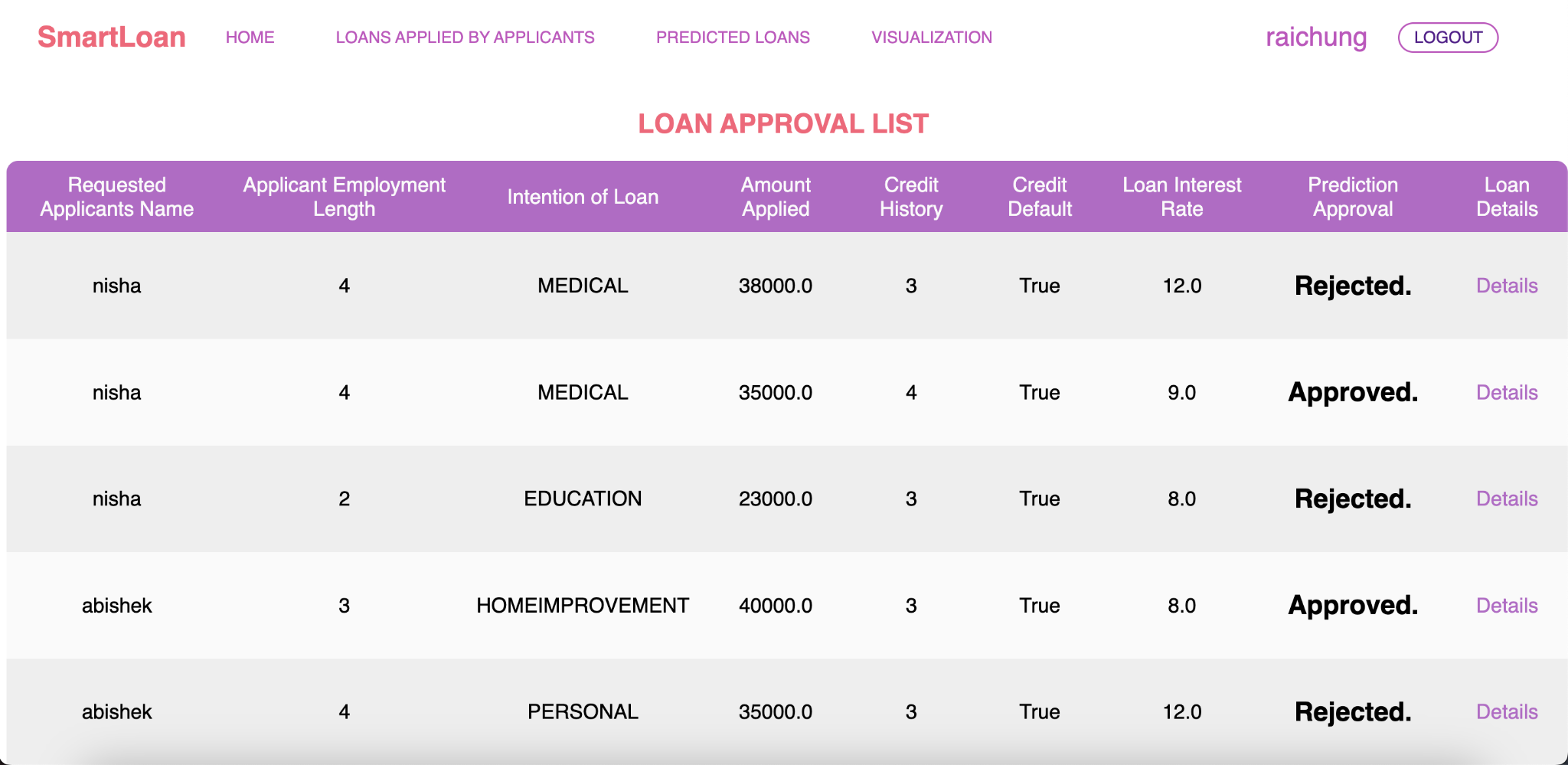
**Appendix C Apply Loan Page**



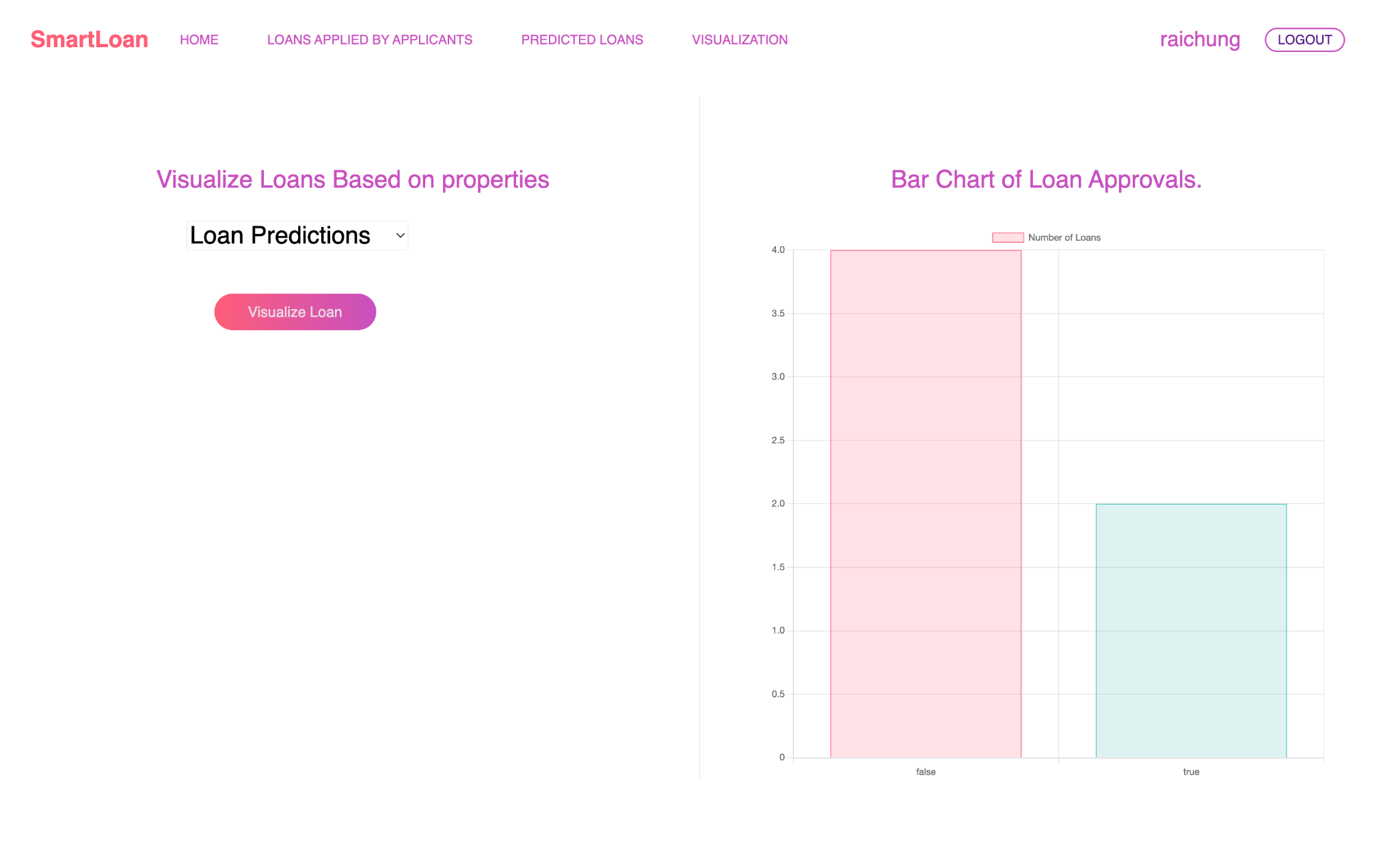
**Appendix D Requested Loans page**



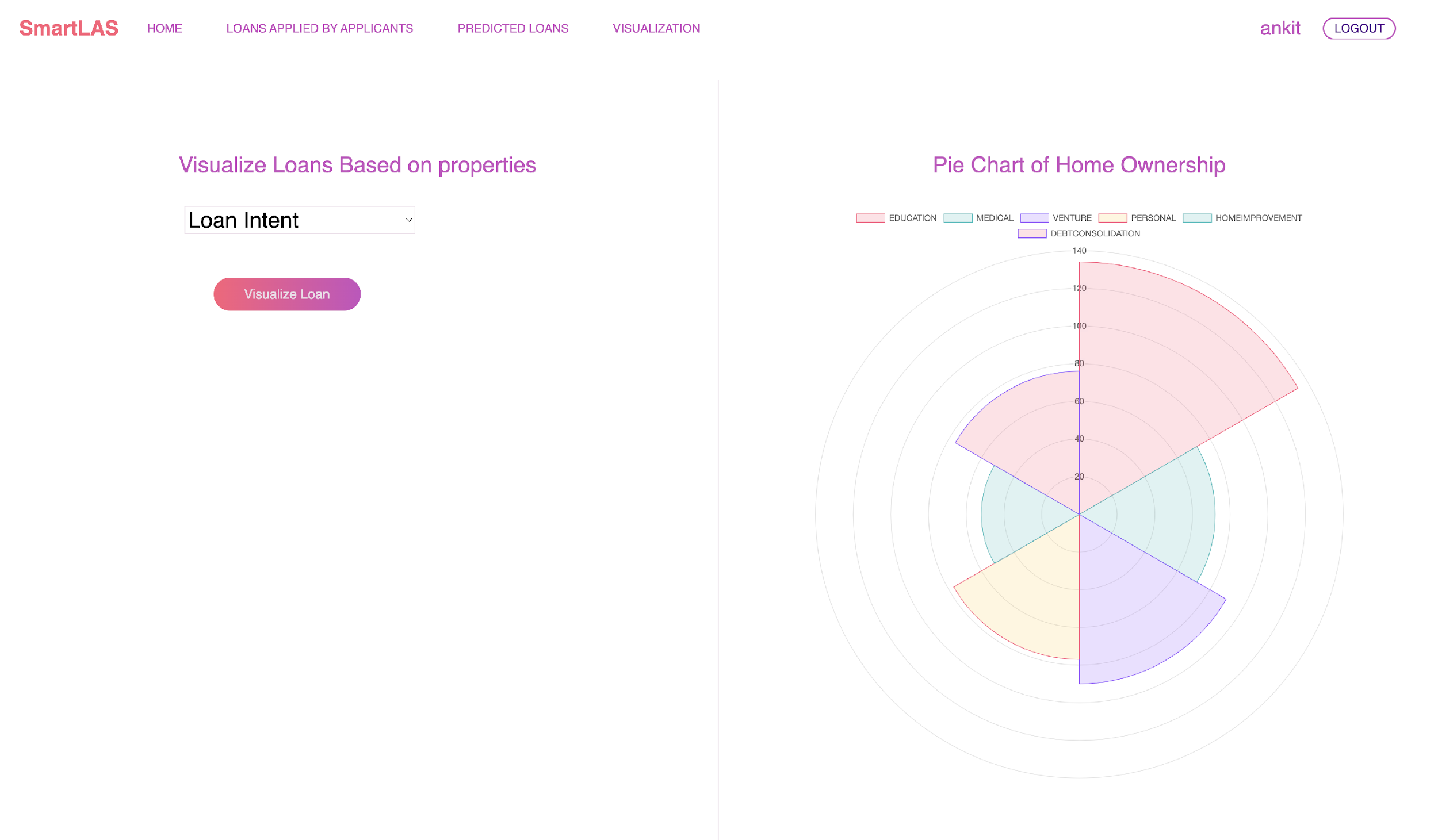
**Appendix E Provide Loan Data by Employee page**

****

**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 = []

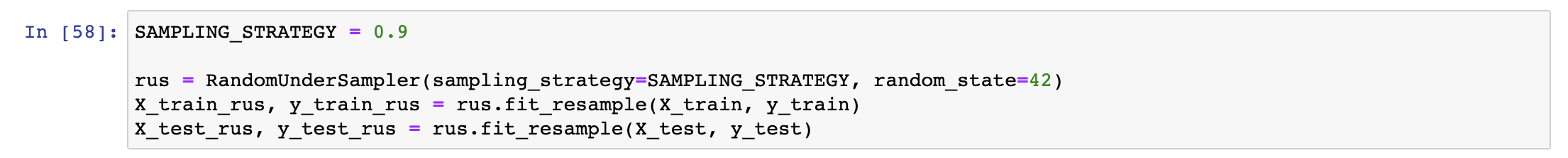
for preds in labels:

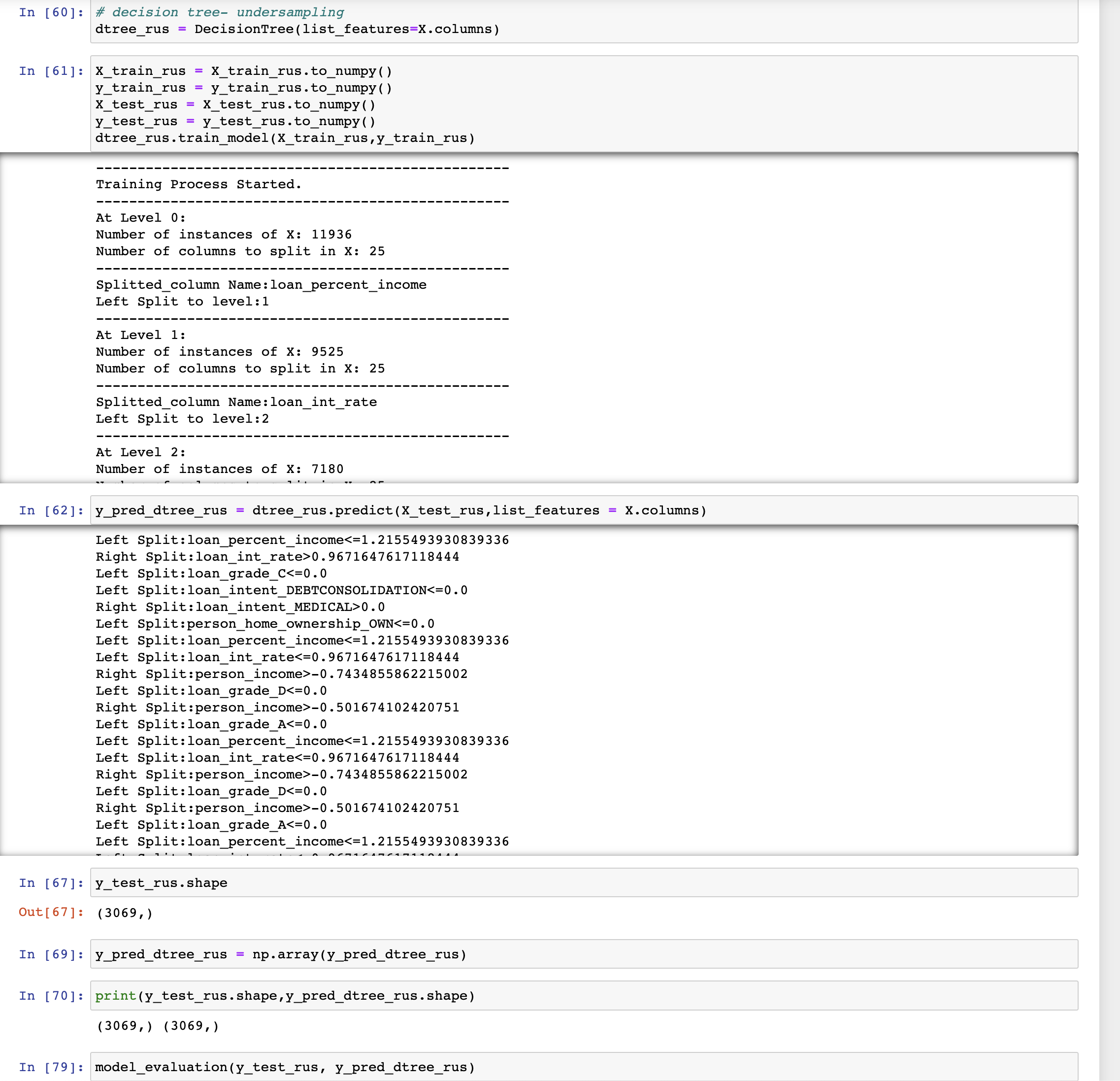
counter = Counter(preds)

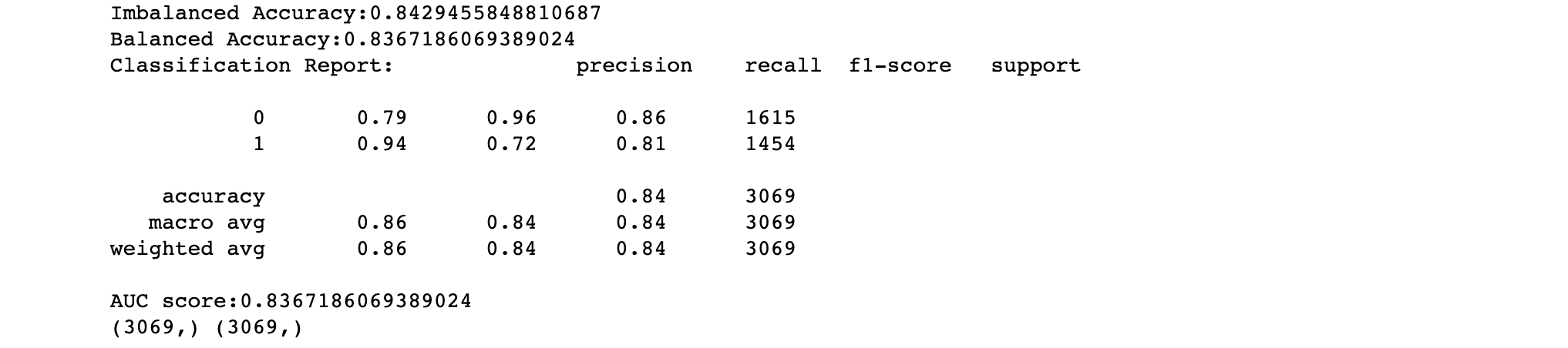
predictions.append(counter.most\_common(1)[0][0])

return predictions

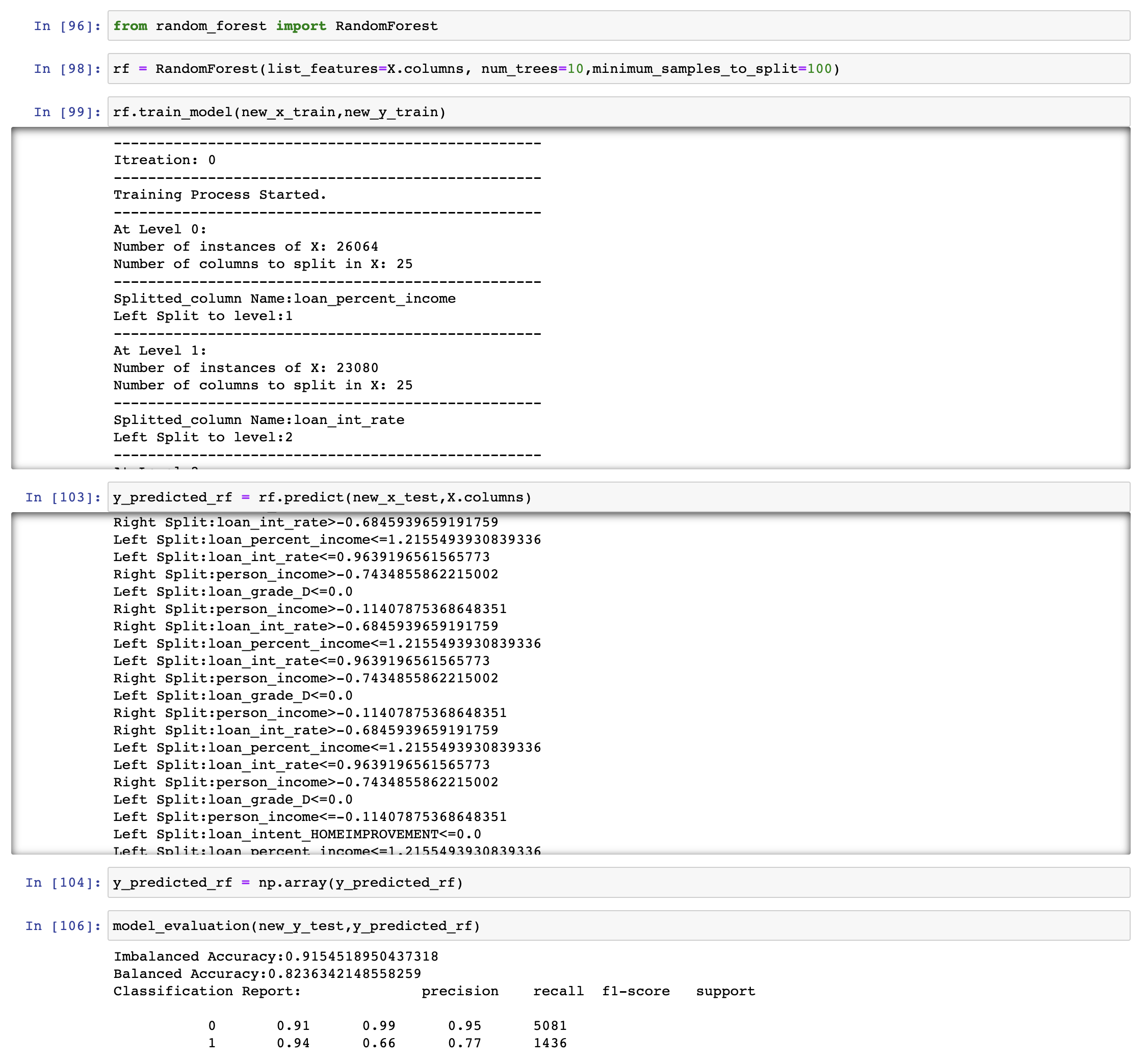
**Notebook Screenshots**

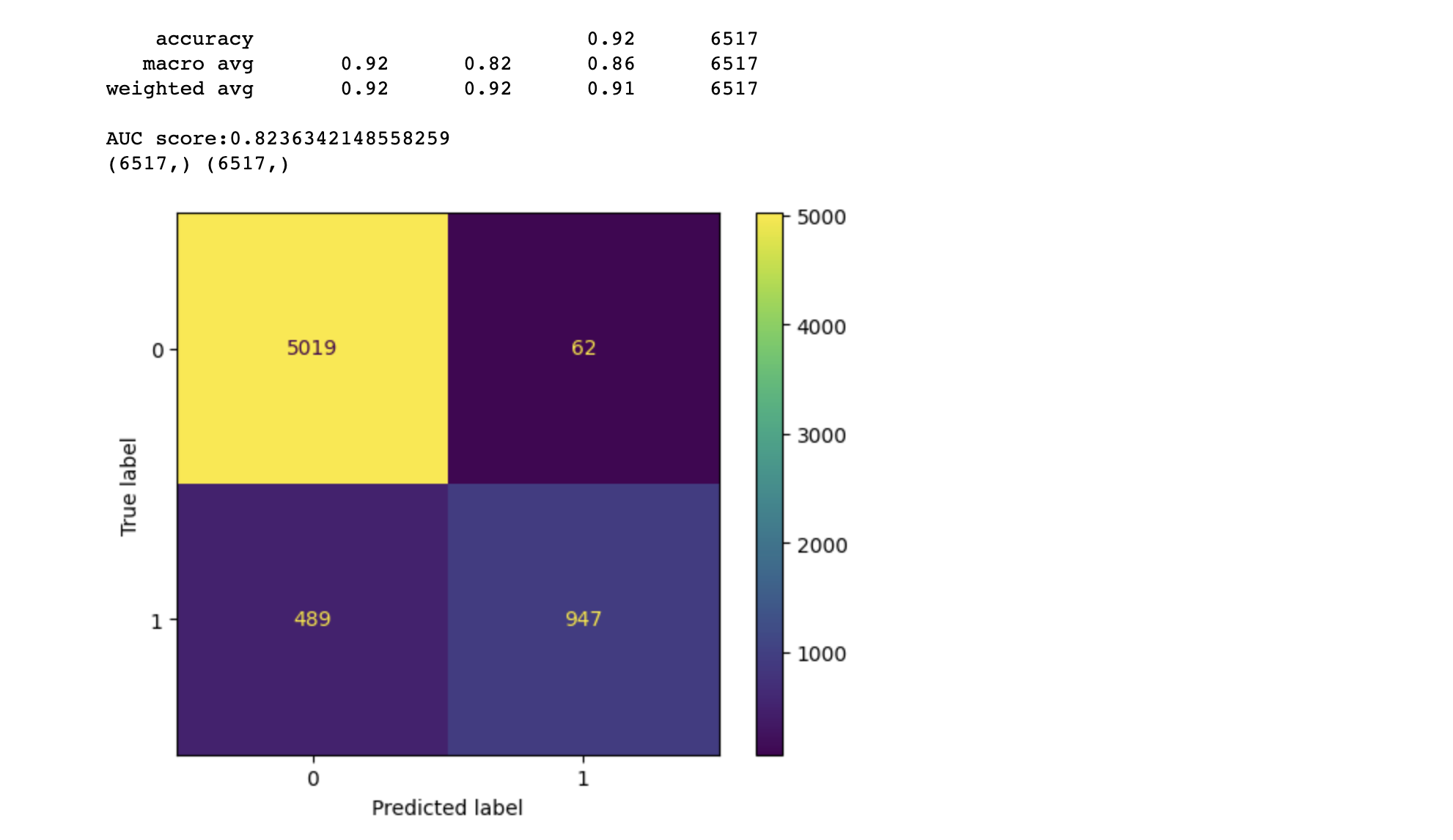
****

****

****

**Appendix I Random Undersampling with their results**

****

****

**Appendix II Random Forest Hyperparameter Tuned model Prediction**