Sri Lanka Institute of Information Technology

Fundamentals of Data Mining - IT3051

Development & Deployment of Classification Model(s) for Bank

Group Project - Final Report

Group Number : G22

Student Number	Name
IT19994406	Basnayake N.S.N.
IT20133504	Weerasekara N.N.
IT20017088	Prabhashi P.A.N.
IT20005726	Liyanage S.R.
IT19169736	Gamage M.G.U.D.

Table of Contents

Only two operational areas of the bank are the focus of the current application, and the dataset is only available for a limited time period. Furthermore, security features like user

1 Terms of Reference

The report is submitted in fulfilment of the requirements for the Fundamentals of Data Mining [IT3051] module, Faculty of Computing, Sri Lankan Institute of Information Technology (SLIIT).

2 Acknowledgment

First of all, we would like to express our sincere thanks to Mr. Prasanna Sumathipala, the module leader, and the lab instructors for their invaluable advice and assistance throughout this group effort. Thank you for your contribution; it was crucial to the success of this initiative.

3 Abstract

The primary purpose of the project is to measure the risk associated with a client's loan. In this case, a historical data set will be used as a source that contains important characteristics that were collected before lending the loan to customers. So, while finding the solution, the risk of lending the loan will be predicted based on the customers' background which will be identified by key related factors such as annual income, income category, age, house ownership, etc. Accordingly, since the objective is to predict, a predictive model is needed and hence classification technique will be used.

Firstly, the data set will be prepared according to the requirements. After that, the selected dataset will be pre-processed. Following this, using the final tools and technologies, the model will be developed. Then the web application will be designed using specific tools and technologies, and then the model will be integrated with that. Thereafter, the finalized web application will be presented to the end user of the banking sector as the final product, which ends users can use according to their needs. In addition, a video clip will also be given for the convenience of the user, who will describe the process of using the application more simply. Finally, the final report will be provided along with all required details.

4 Assumption(s)

Here, although the data set related to the details of loans involving a foreign country, we assume that it may apply to the Sri Lankan banking sector as well.

5 Introduction

5.1 Project Background

The Banking Sector

The banking sector acts as one of the key controllers of the macro economy in any country. When it comes to Sri Lanka has several banks which are under the governance of the central bank of Sri Lanka. Mainly, the banking sector in Sri Lanka is divided into two categories such as comprises Licensed Commercial Banks (LCBs) and Licensed Specialized banks (LSBs). The banking sector dominates the financial system and accounts for the highest share of the total assets in the financial system. Banks play a critical role within the Sri Lankan financial system, as they are engaged in the provision of liquidity to the entire economy while transforming the risk characteristics of assets. Banks also engaged in providing payment services such as Advancing of Loans, Discounting of Bills of Exchange, Check Payment, Collection and Payment of Credit Instruments, Foreign Currency Exchange, etc. Therefore, the banking sector plays a huge role in the economy, facilitating all entities to carry out their financial transactions. On the other hand, banks can create vulnerabilities of a systemic nature, partly due to a mismatch in the maturity of assets and liabilities and their interconnectedness. Therefore, the soundness of banks is important, as it contributes towards maintaining confidence in the financial system, and any failure may have the potential to impact on activities of all other financial and non-financial entities, and finally the economy. So, this critical impact of the banking sector on a country leads to the problem domain which has been selected. All credit-related activities are important to a bank. Interest received on various loans and advances to industries, corporates and individuals is the bank's main source of income.

Olenzro Bank

Olenzro Bank is recognized as a LCB or otherwise Licensed Commercial Bank which is offering various facilities for the customers. Advancing of Loans, Discounting of Bills of Exchange, Check Payment, Collection and Payment of Credit Instruments, Foreign Currency Exchange are some of the services that providing. As for banks money is the product that use in the business. And as for the main thing there is a risk that when dealing with the money. So, that risk needs to be considered as a huge point in Olenzro bank’s day to day and long-term operations. Therefore, taking out the above explained matter for the problem and make up a solution for that will be a secure step to take as a bank.

5.2 Problem Statement

Loans are the cornerstone of any bank. All businesses sell products, and a bank's product is money. Loans allow for growth in the overall money supply in an economy and open competition by lending to new businesses. Banks generally make money by borrowing money from depositors and compensating them with a certain interest rate. The banks will lend the money out to borrowers, charging the borrowers a higher interest rate and profiting off the interest rate spread. But on the other hand, lending any kind of loan to a customer is a risk to a bank. Therefore, Credit risk is considered one of the biggest risks for banks. Accordingly, the importance of credit risk measurement is paramount, before lending any kind of loan facility. This is the basis on which the bank can calculate the likelihood that a client will default on a loan or comply with other contractual obligations. Proper risk assessment can help the Bank promptly and accurately identify risks and apply appropriate controls to mitigate risks or identify unacceptable risks to be avoided. So, in that important problem, the objective of this project is to help the bank sector to measure the borrower's capacity to repay the loan before lending the loan. If it says in other words the motive is to identify the risk of lending a loan, based on the customer's historical data that are relevant to the problem. To accomplish this task, the group will be using a data set that contains past details of customers who obtained loans stating whether it is paid accurately or not while mentioning whether it is a good or bad loan. The data set contains important details of customers such as annual income, home ownership, age, etc., which were collected when a customer was applying for the loan.

5.3 Business Objective

The objective of the selected problem and the whole project process is about supplying credit-related services, especially lending loans to customers who are suitable and have a high possibility to pay back.

5.4 Proposed Solution

The primary purpose of the project is to measure the risk associated with a client's loan. In this case, a historical data set will be used as a source that contains important characteristics that were collected before lending the loan to customers. So, while finding the solution, the risk of lending the loan will be predicted based on the customers' background which will be identified by key related factors such as annual income, income category, age, house ownership, etc. Accordingly, since the objective is to predict, a predictive model is needed and hence classification technique will be used. Firstly, the data set will be prepared according to the requirements. After that, the selected dataset will be pre-processed. Following this, using the final tools and technologies, the model will be developed. Then the web application will be designed using specific tools and technologies, and then the model will be integrated with that. Thereafter, the finalized web application will be presented to the end user of the banking sector as the final product, which ends users can use according to their needs. In addition, a video clip will also be given for the convenience of the user, who will describe the process of using the application more simply. Finally, the final report will be submitted along with all required details.

5.5 Development Team's Approach

As for the development team the main purpose of this project is to deliver a web application which is presenting all needed information. Most importantly it must be user friendly so anyone, even without any expertise knowledge can use the application. And for more the accuracy of the model must be ensured within the development. For that data preprocessing is a needed step. And the data is contained within varied time period which has both past and current data study the area and the scope is a must to take only the relevant data. After considering all these facts to guarantee the business requirements and the scope is correctly identified and done within the application, consistent testing and the evaluation will be done by the development team as needed.

6 Dataset Analysis & Preparation

6.1 Overview

6.1.1 Dataset: Loan Condition of Customers (Good Loans vs Bad Loans)

Field	Description	Type
Id	System generated unique number sequence to identify records.	Int
Year	Defines the year of the loan application	Int
Issue_d	Specifies the loan issue date to the customer	Date
Final_d	Specifies the date the loan matures	Int
Emp_length_int	Specifies the employment duration of the customer	Float
Home_ownership	Specifies house ownership of the customer (Rent,Mortgage, Own)	String
Home_ownership_cat	User generated number sequences for the house ownership values	Int
Income_category	Specifies the income classes of the customer (High,Low)	String
Annual_inc	Specifies annual income earned by the customer	Int
Income_cat	User generated number sequences for the income classes	Int
Loan_amount	Specifies the loan amount applied by the customer	Int
Term	Specifies the loan duration (36 months, 60 months)	String
Term_cat	User generated number sequences for loan duration	Int
Application_type	Specifies the type of the loan application submitted by customer (Individual, Joint)	String
Application_type_cat	User generated number sequences for type of the loan application	Int
Purpose	Defines the purpose of the loan applied for	String
Interest_payments	Specifies the interest payment category (High,Low)	String
Interest-payments_cat	User generated number sequences for the interest payment categories	Int
Loan_condition	Specifies the risk status of the loan (Good Loan, Bad Loan)	String
Loan_condition_cat	User generated number sequences for loan risk status	Int
Interest_rate	Assigned interest for the specific loan	Float
Grade	Assigned loan grade based on the credit score	String
Grade_cat	User generated number sequences for loan grades	Int

Dti	Specifies debt to income ratio of the customer	Float
Total_pymnt	Specifies the total payment to be received for the given loan	Float
Total_rec_prncp	Specifies total recorded principals for the given loan	Float
Recoveries	Specifies total recoveries for a given loan	Float
Installment	Specifies the monthly installment amount for a given loan	Float
Region	Specifies the region a particular customer belong to	String
Age	Specifies the age of the customer	Int

6.2 Analyze the dataset

To get a brief understanding about the dataset, we follow these steps.

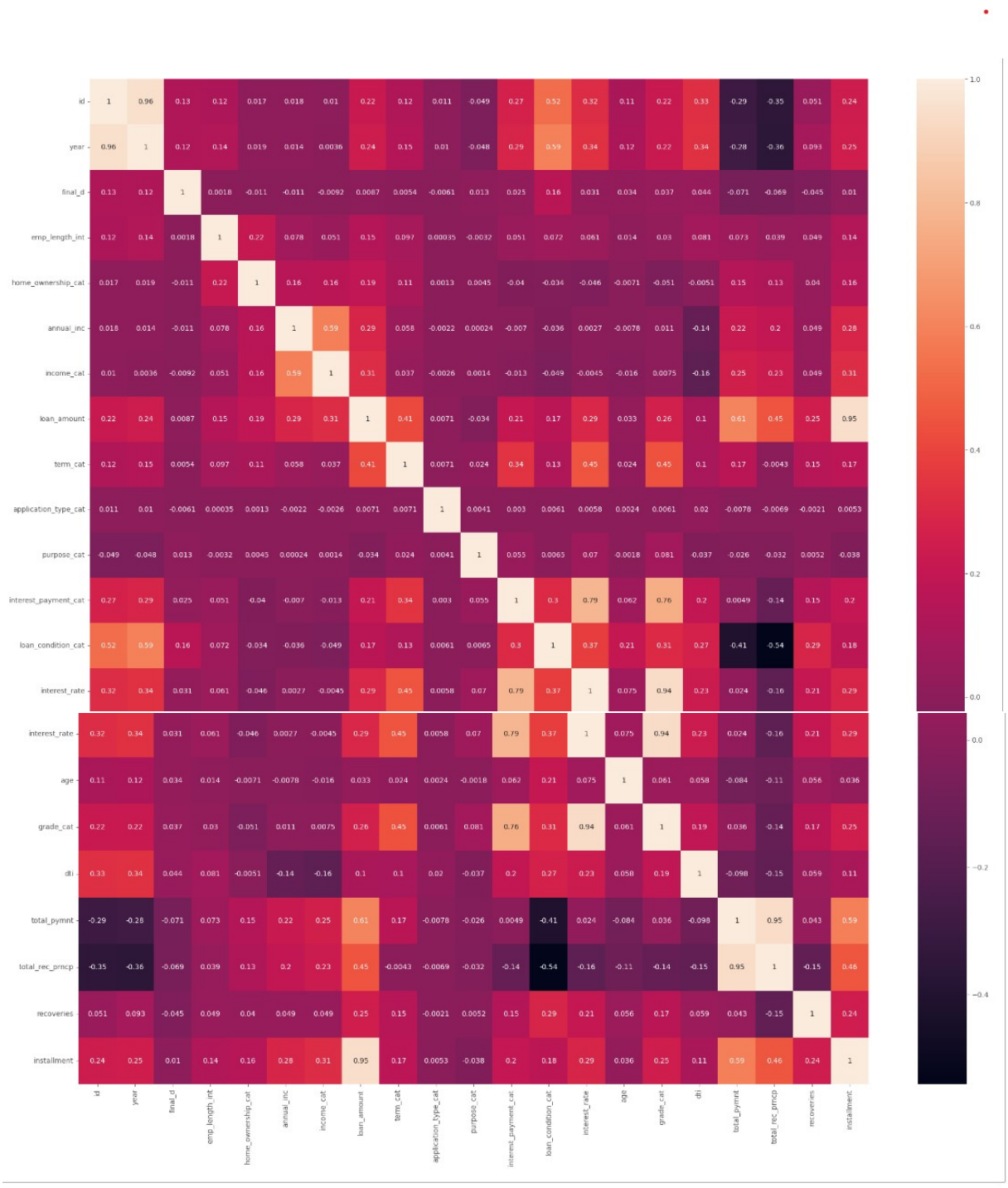
Step 01: Calculate the size of the dataset with number of features/attributes and check data types of the columns

```
In [4]: Train_Data.shape
Out[4]: (113280, 31)

In [5]: #cheking  data types of the coulmn
        Train_Data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113280 entries, 0 to 113279
Data columns (total 31 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   id                    113280 non-null  int64
 1   year                  113280 non-null  int64
 2   issue_d               113280 non-null  object
 3   final_d               113280 non-null  int64
 4   emp_length_int        113280 non-null  float64
 5   home_ownership        113280 non-null  object
 6   home_ownership_cat    113280 non-null  int64
 7   income_category       113280 non-null  object
 8   annual_inc            113280 non-null  int64
 9   income_cat            113280 non-null  int64
 10  loan_amount           113280 non-null  int64
 11  term                  113280 non-null  object
 12  term_cat              113280 non-null  int64
 13  application_type      113280 non-null  object
 14  application_type_cat  113280 non-null  int64
 15  purpose               113280 non-null  object
 16  purpose_cat           113280 non-null  int64
 17  interest_payments     113280 non-null  object
 18  interest_payment_cat  113280 non-null  int64
 19  loan_condition        113280 non-null  object
 20  loan_condition_cat    113280 non-null  int64
 21  interest_rate         113272 non-null  float64
 22  age                   113280 non-null  int64
 23  grade                 113280 non-null  object
 24  grade_cat             113280 non-null  int64
 25  dti                   113280 non-null  float64
 26  total_pymnt           113280 non-null  float64
 27  total_rec_prncp       113280 non-null  float64
 28  recoveries            113280 non-null  float64
 29  installment           113280 non-null  float64
 30  region                113280 non-null  object
dtypes: float64(7), int64(14), object(10)
memory usage: 26.8+ MB
```

Step 02: Get an idea about columns and generate column names

```
In [6]: #cheking the available columns in the data set
        Train_Data.columns
Out[6]: Index(['id', 'year', 'issue_d', 'final_d', 'emp_length_int', 'home_ownership',
               'home_ownership_cat', 'income_category', 'annual_inc', 'income_cat',
               'loan_amount', 'term', 'term_cat', 'application_type',
               'application_type_cat', 'purpose', 'purpose_cat', 'interest_payments',
               'interest_payment_cat', 'loan_condition', 'loan_condition_cat',
               'interest_rate', 'age', 'grade', 'grade_cat', 'dti', 'total_pymnt',
               'total_rec_prncp', 'recoveries', 'installment', 'region'],
              dtype='object')
```

Step 03: Based on the observation of the dataset, plot a diagram to identify the correlation between the features



6.3 Data Preprocessing

After a brief understanding of the dataset, applied necessary techniques to understand the nature of the dataset further. Meanwhile it preprocessed the high-quality dataset

Step 01: Identify the null values in the dataset

Identify null values

```
In [9]: #check null values of the trainin gdata set

        Train_Data.isnull()
```

Out[9]:

	id	year	issue_d	final_d	emp_length_int	home_ownership	home_ownership_cat	income_category	annual_inc	income_cat	...	interest_rate	age	gr
0	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
1	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
3	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3275	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
3276	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
3277	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
3278	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa
3279	False	False	False	False	False	False	False	False	False	False	...	False	False	Fa

280 rows × 31 columns

Step 02: Handle numerical missing data with mean and check null values replaced or not

handling numerical missing data

```
In [11]: # handle numerical missing data
         Train_Data['interest_rate'] = Train_Data['interest_rate'].fillna(Train_Data['interest_rate'].mean())

In [12]: # cheking whether the null values replaced or not
         Train_Data.isnull().sum().sort_values(ascending=False)
```

Out[12]:

```
id                      0
purpose_cat             0
installment             0
recoveries              0
total_rec_prncp         0
total_pymnt             0
dti                     0
grade_cat               0
grade                   0
age                     0
interest_rate           0
loan_condition_cat      0
loan_condition          0
interest_payment_cat    0
interest_payments       0
purpose                 0
year                    0
application_type_cat    0
application_type        0
term_cat                0
term                    0
loan_amount             0
income_cat              0
annual_inc              0
income_category         0
home_ownership_cat      0
home_ownership          0
emp_length_int          0
final_d                 0
issue_d                 0
region                  0
dtype: int64
```

Step 03: Check duplicate values in the dataset

```
[14]: #check duplicate values of the trainin gdata set


      # Count of duplicate records persent in the data

      duplicateRows = Train_Data[Train_Data.duplicated()]
      print("Number of duplicate rows:",duplicateRows.shape)
```

```
Number of duplicate rows: (0, 31)
```

Step 04: Change the attribute values to a meaningful manner to maintain the consistency between the fields of the training dataset and user inputs

adding meaningful attrubute values to maintain the consistency

```
In [15]: Train_Data.loc[Train_Data['term'] == '36 months', 'term'] = 36
         Train_Data.loc[Train_Data['term'] == '60 months', 'term'] = 60
         Train_Data.loc[Train_Data['home_ownership'] == 'MORTGAGE', 'home_ownership'] = 'Mortgage'
         Train_Data.loc[Train_Data['home_ownership'] == 'ANY', 'home_ownership'] = 'Any'
         Train_Data.loc[Train_Data['home_ownership'] == 'RENT', 'home_ownership'] = 'Rent'
         Train_Data.loc[Train_Data['home_ownership'] == 'OWN', 'home_ownership'] = 'Own'
         Train_Data.loc[Train_Data['home_ownership'] == 'NONE', 'home_ownership'] = 'None'
         Train_Data.loc[Train_Data['home_ownership'] == 'OTHER', 'home_ownership'] = 'Other'
         Train_Data.loc[Train_Data['application_type'] == 'INDIVIDUAL', 'application_type'] = 'Individual'
         Train_Data.loc[Train_Data['application_type'] == 'JOINT', 'application_type'] = 'Joint'
         Train_Data.loc[Train_Data['purpose'] == 'credit_card', 'purpose'] = 'Credit Card'
         Train_Data.loc[Train_Data['purpose'] == 'debt_consolidation', 'purpose'] = 'Debt consolidation'
         Train_Data.loc[Train_Data['purpose'] == 'renewable_energy', 'purpose'] = 'Renewable energy'
         Train_Data.loc[Train_Data['purpose'] == 'major_purchase', 'purpose'] = 'Major Purchase'
         Train_Data.loc[Train_Data['purpose'] == 'small_business', 'purpose'] = 'Small Business'
         Train_Data.loc[Train_Data['purpose'] == 'home_improvment', 'purpose'] = 'Home Improvment'
         Train_Data.loc[Train_Data['purpose'] == 'car', 'purpose'] = 'Car'
         Train_Data.loc[Train_Data['purpose'] == 'educational', 'purpose'] = 'Educational'
         Train_Data.loc[Train_Data['purpose'] == 'house', 'purpose'] = 'House'
         Train_Data.loc[Train_Data['purpose'] == 'medical', 'purpose'] = 'Medical'
         Train_Data.loc[Train_Data['purpose'] == 'moving', 'purpose'] = 'Moving'
         Train_Data.loc[Train_Data['purpose'] == 'other', 'purpose'] = 'Other'
         Train_Data.loc[Train_Data['purpose'] == 'vacation', 'purpose'] = 'Vacation'
         Train_Data.loc[Train_Data['purpose'] == 'wedding', 'purpose'] = 'Wedding'
         Train_Data
```

Out[15]:

	id	year	issue_d	final_d	emp_length_int	home_ownership	home_ownership_cat	income_category	annual_inc	income_cat	...	interest_rate
0	1	2009	01/08/2009	1102011	0.5	Rent	1	Low	85000	1	...	11.89
1	2	2008	01/07/2008	1032010	0.5	Rent	1	Low	30000	1	...	16.08
2	3	2008	01/05/2008	1062011	0.5	Rent	1	Low	65000	1	...	10.71
3	4	2008	01/04/2008	1102008	10.0	Mortgage	3	Medium	189500	2	...	16.08
4	5	2014	01/08/2014	1122014	1.0	Rent	1	Low	70000	1	...	16.99
...	...	...	...	...	...	...	...	...	...	...	...	...

Step 05: Drop columns which are not related and less important like user generated fields

In [16]:
```
#drop column which are not related
Train_Data = Train_Data.drop(columns=['id',
'year',
'issue_d',
'final_d',
'home_ownership_cat',
'income_cat',
'term_cat',
'application_type_cat',
'purpose_cat',
'interest_payment_cat',
'loan_condition_cat',
'grade_cat',
'total_rec_prncp',
'recoveries',
'region'],axis=1)

Train_Data
```

Out[16]:

	emp_length_int	home_ownership	income_category	annual_inc	loan_amount	term	application_type	purpose	interest_payments	loan_condition
0	0.5	Rent	Low	85000	25000	36 months	Individual	Debt consolidation	Low	Good Loan
1	0.5	Rent	Low	30000	1000	36 months	Individual	Debt consolidation	High	Good Loan
2	0.5	Rent	Low	65000	7000	36 months	Individual	Credit Card	Low	Good Loan
3	10.0	Mortgage	Medium	189500	7000	36 months	Individual	Debt consolidation	High	Bad Loan
4	1.0	Rent	Low	70000	25000	36 months	Individual	Debt consolidation	High	Bad Loan
...	...	...	...	...	...	...	...	...	...	...
113275	0.5	Rent	Low	56000	15000	36 months	Individual	Debt consolidation	High	Bad Loan

6.4 Data Visualizing

6.4.1 Exploratory data analysis

Step 01: Check whether the categorical default or not based on loan condition. Furthermore, check the features with respect to target

In [17]:
```
#setting a value for indentify good loan and bad loan with a function
def defaulted(x):
    if x == 'Good Loan':
        return 0
    else:
        return 1
```

In [18]:
```
Train_Data['loan_cat'] = Train_Data['loan_condition'].apply(lambda x: defaulted(x))
```

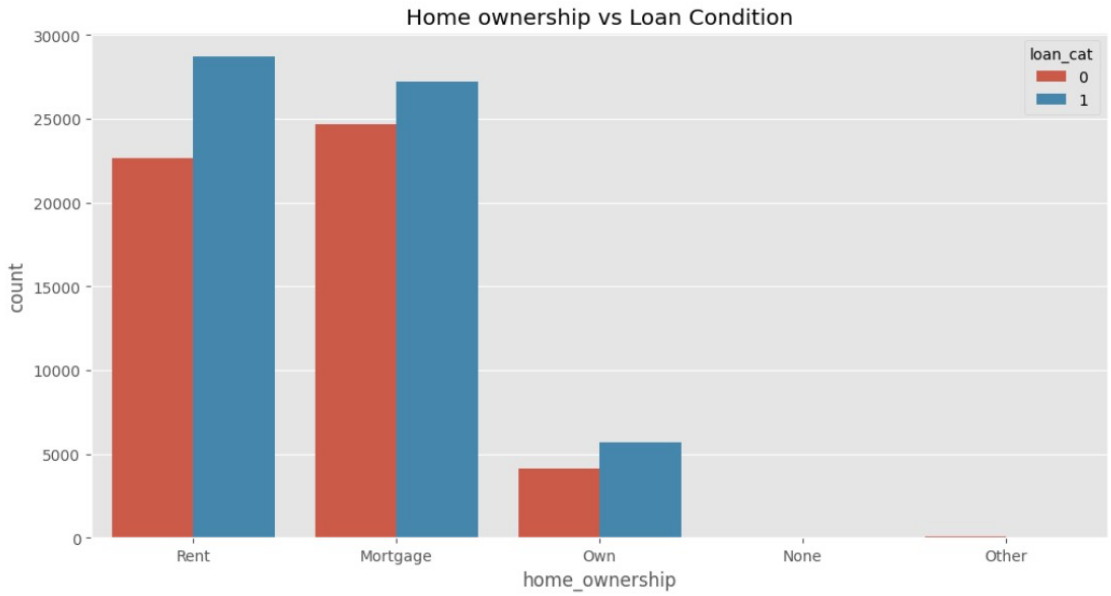
Step 02: Get the plots to get a clear idea about how the features affect to the loan condition

- Home ownership status vs loan condition

**home_ownership: Home Owndership status

```
In [19]: plt.figure(figsize=(12,6))
         plt.title("Home ownership vs Loan Condition")
         sns.countplot(x='home_ownership',data=Train_Data, hue='loan_cat')
```

```
Out[19]: <AxesSubplot: title={'center': 'Home ownership vs Loan Condition'}, xlabel='home_ownership', ylabel='count'>
```

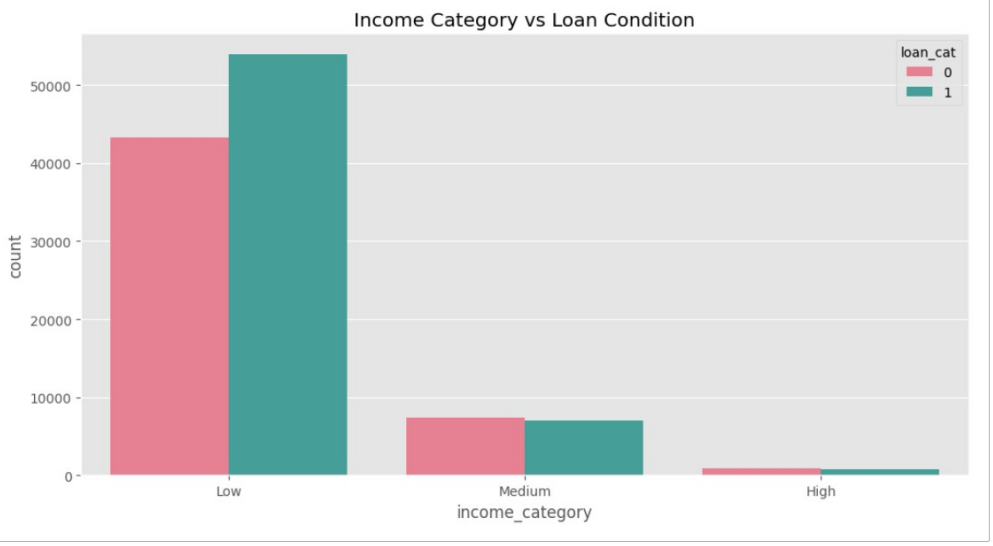


- Income category vs loan condition

**Income category

```
In [59]: plt.figure(figsize=(12,6))
         plt.title("Income Category vs Loan Condition")
         sns.countplot(x='income_category',data=Train_Data, hue='loan_cat',palette="husl")
```

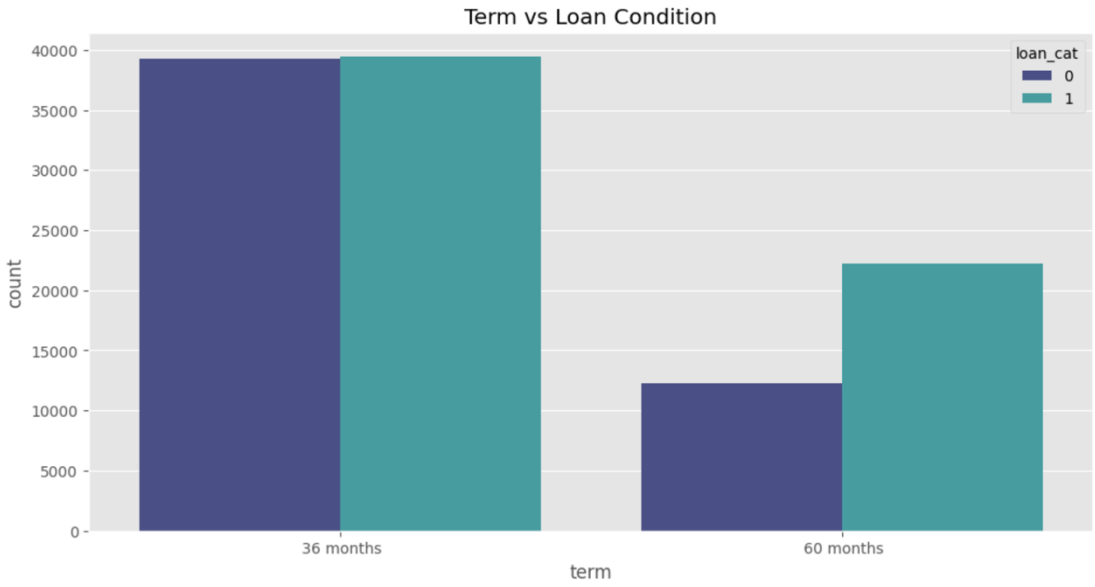
```
Out[59]: <AxesSubplot: title={'center': 'Income Category vs Loan Condition'}, xlabel='income_category', ylabel='count'>
```



- Term vs loan condition

```
In [79]: plt.figure(figsize=(12,6))
         plt.title("Term vs Loan Condition")
         sns.countplot(x='term',data=Train_Data, hue='loan_cat',palette="mako")
```

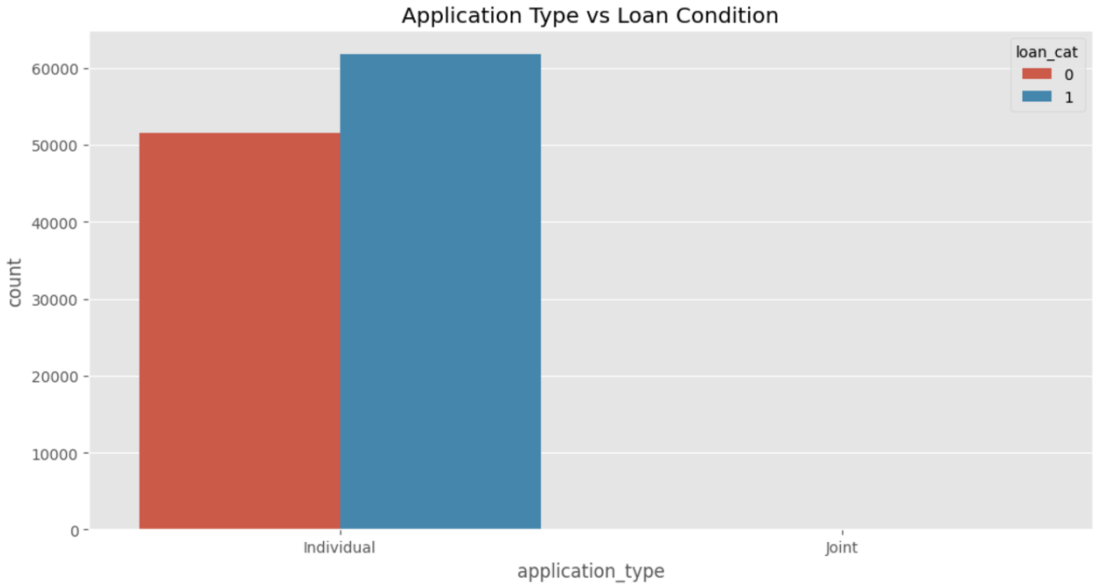
```
Out[79]: <AxesSubplot: title={'center': 'Term vs Loan Condition'}, xlabel='term', ylabel='count'>
```



- Application type vs loan condition

```
In [80]:
         plt.figure(figsize=(12,6))
         plt.title("Application Type vs Loan Condition")
         sns.countplot(x='application_type',data=Train_Data, hue='loan_cat')
```

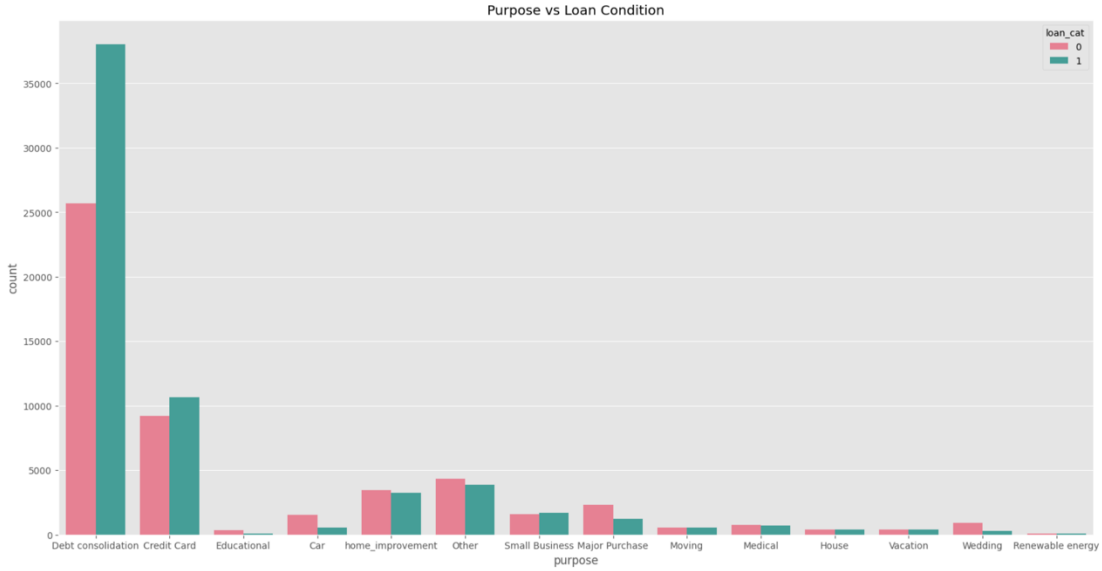
```
Out[80]: <AxesSubplot: title={'center': 'Application Type vs Loan Condition'}, xlabel='application_type', ylabel='count'>
```



- Purpose vs loan condition

```
In [81]: plt.figure(figsize=(20,10))
         plt.title("Purpose vs Loan Condition")
         sns.countplot(x='purpose',data=Train_Data, hue='loan_cat',palette="husl")
```

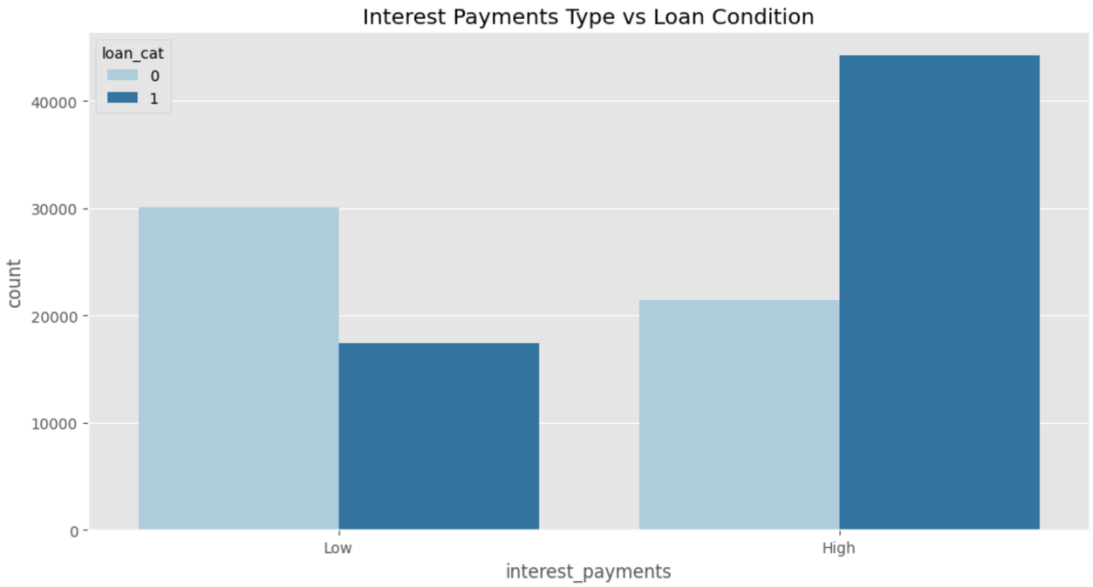
```
Out[81]: <AxesSubplot: title={'center': 'Purpose vs Loan Condition'}, xlabel='purpose', ylabel='count'>
```



- Interest payments type vs loan condition

```
In [22]: plt.figure(figsize=(12,6))
         plt.title("Interest Payments Type vs Loan Condition")
         sns.countplot(x='interest_payments',data=Train_Data, hue='loan_cat',palette="Paired")
```

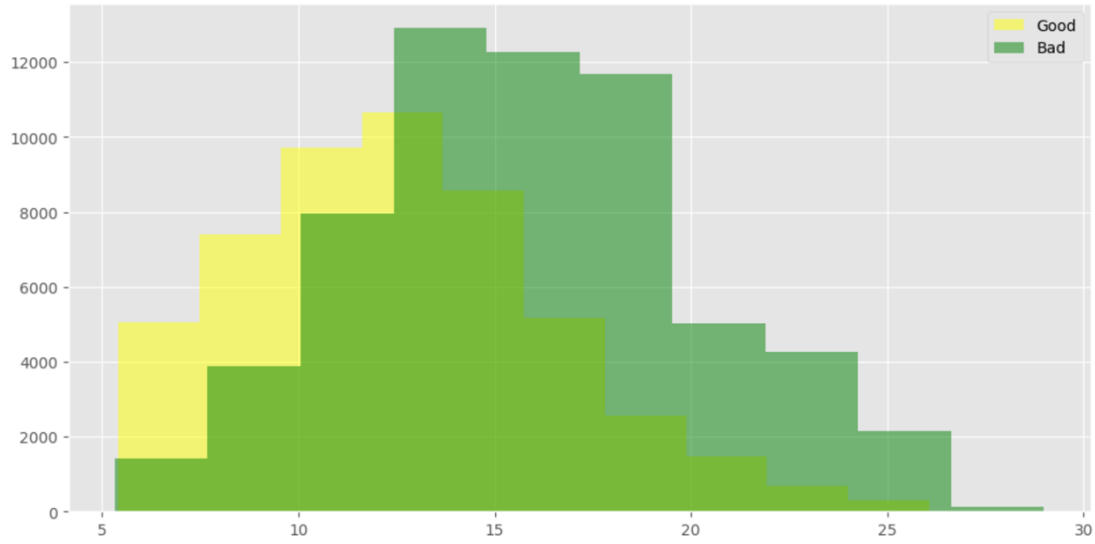
```
Out[22]: <AxesSubplot: title={'center': 'Interest Payments Type vs Loan Condition'}, xlabel='interest_payments', ylabel='coun
         t'>
```



Step 03: Check numerical data with loan condition

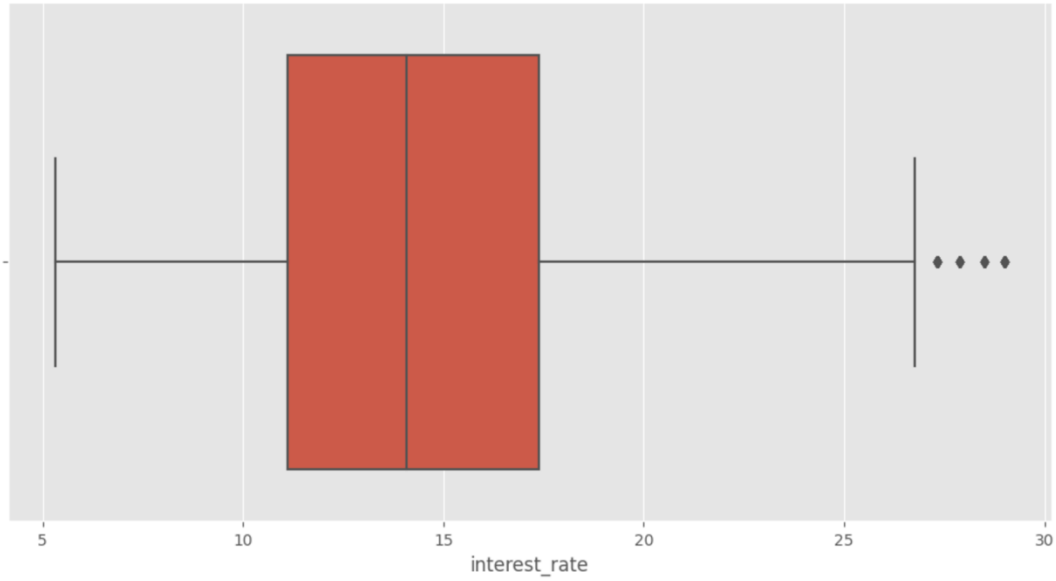
- Interest rate vs loan condition

```
In [82]: plt.figure(figsize=(12,6))
         plt.hist(Train_Data[Train_Data['loan_cat']==0]['interest_rate'],color='yellow',alpha=0.5,label='Good')
         plt.hist(Train_Data[Train_Data['loan_cat']==1]['interest_rate'],color='green',alpha=0.5,label='Bad')

         plt.legend()

Out[82]: <matplotlib.legend.Legend at 0x1a9ce9e86d0>
```



- Interest rate box plot

```
In [83]: plt.figure(figsize=(12,6))
         sns.boxplot(x=Train_Data['interest_rate'])

Out[83]: <AxesSubplot: xlabel='interest_rate'>
```

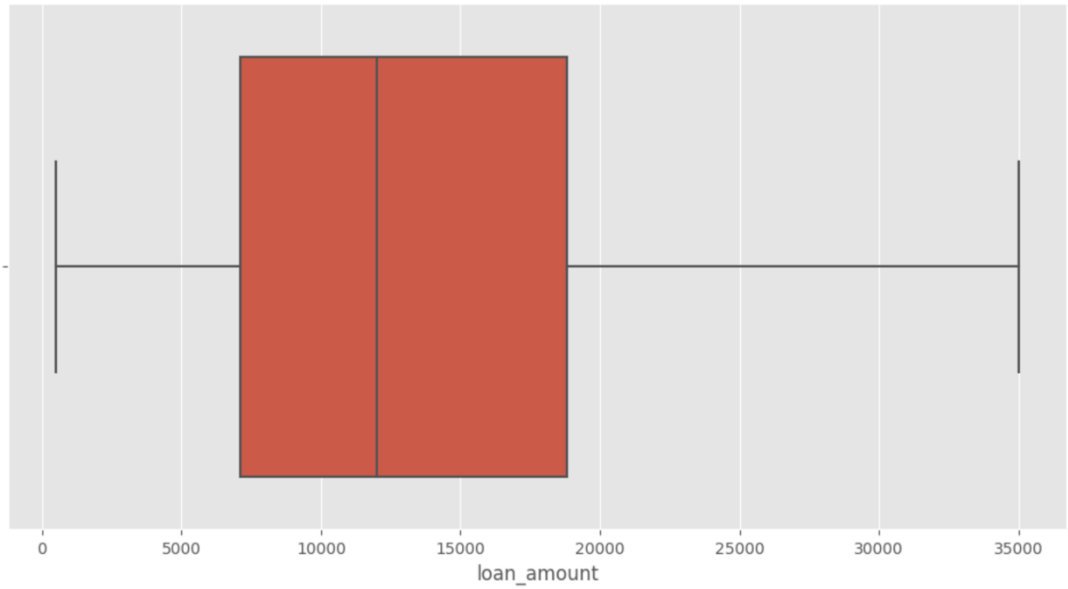


Interest rates have outliers. So, have to remove outliers to reduce the variance of the training data to improve the test accuracy.

```
In [84]: #removing outliers
         outliers = Train_Data[Train_Data['interest_rate'] > Train_Data['interest_rate'].quantile(.99)].index

In [85]: Train_Data.loc[outliers,'interest_rate'] = Train_Data['interest_rate'].quantile(.99)

In [86]: scaler = MinMaxScaler()
         Train_Data['interest_rate'] = scaler.fit_transform(Train_Data['interest_rate'].values.reshape(-1,1))
```

- Loan amount box plot

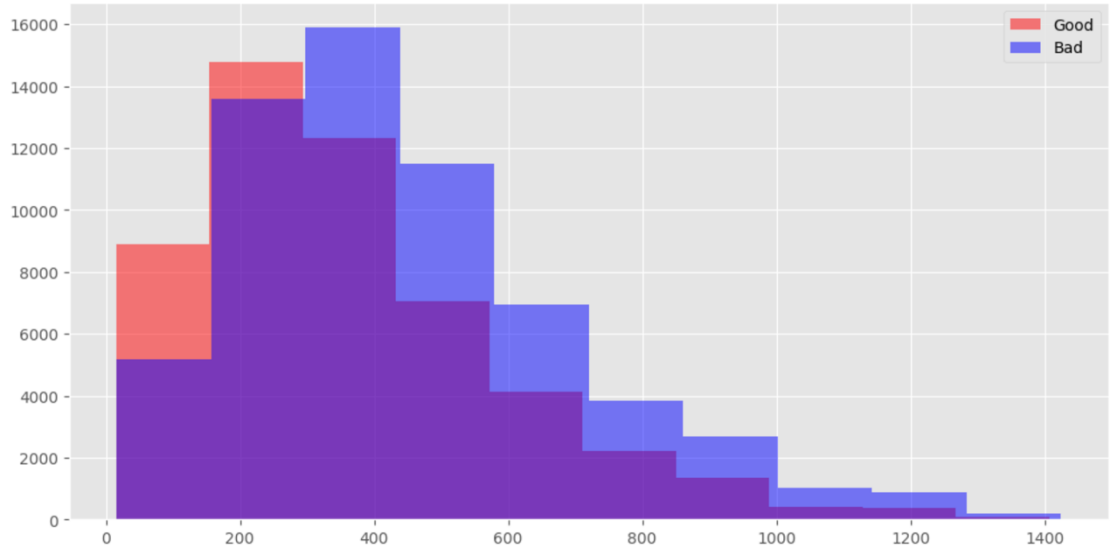
```
In [87]: plt.figure(figsize=(12,6))
         sns.boxplot(x=Train_Data['loan_amount'])
Out[87]: <AxesSubplot: xlabel='loan_amount'>
```



- Installment histogram

```
In [88]: plt.figure(figsize=(12,6))
         plt.hist(Train_Data[Train_Data['loan_cat']==0]['installment'],color='red',alpha=0.5,label='Good')
         plt.hist(Train_Data[Train_Data['loan_cat']==1]['installment'],color='blue',alpha=0.5,label='Bad')

         plt.legend()
```

```
Out[88]: <matplotlib.legend.Legend at 0x1a9d21cd630>
```

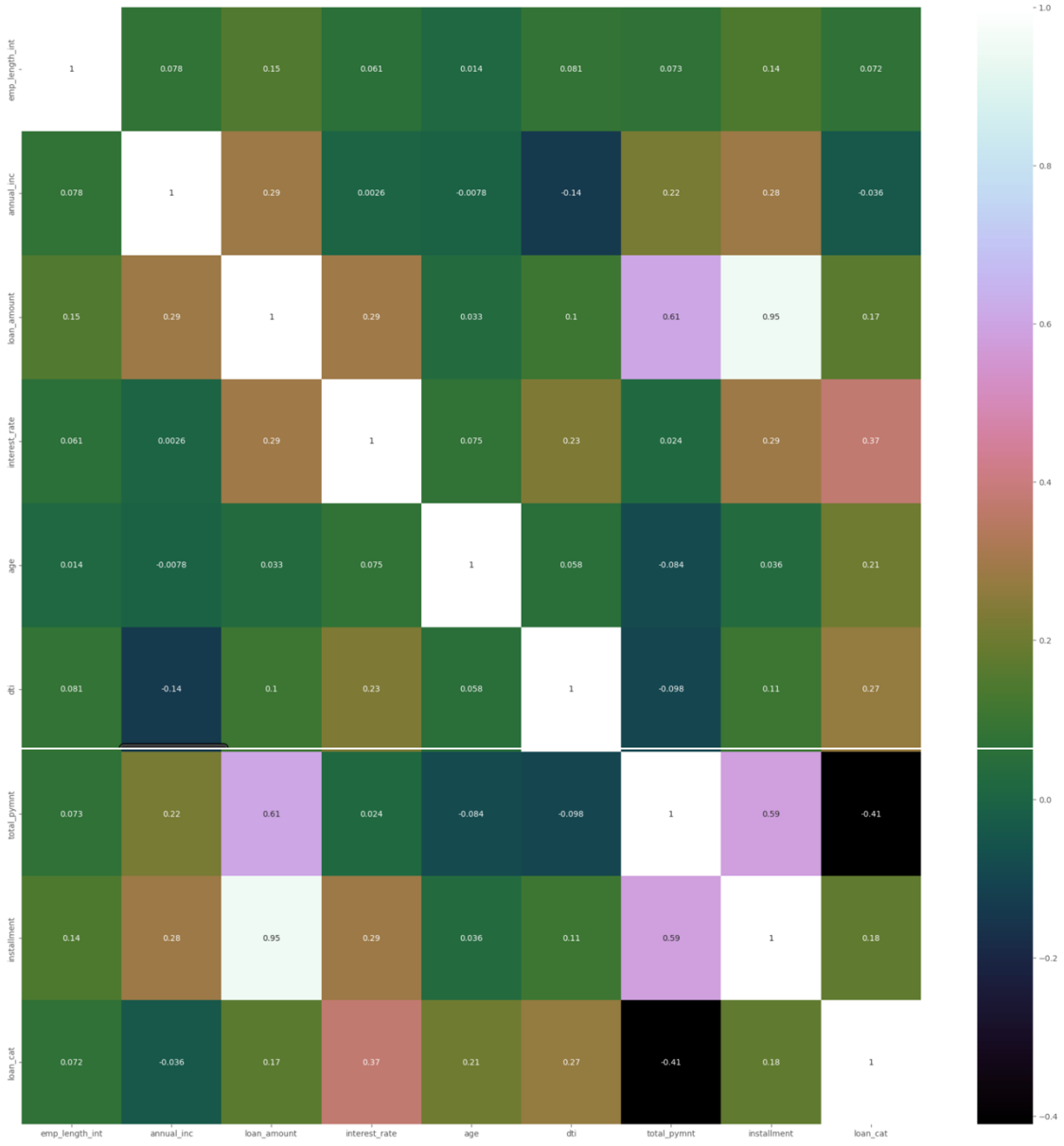


Step 04: Identify correlation between attributes after the initial preprocessing

```
In [89]: plt.figure(figsize=(25,25))
         sns.heatmap(Train_Data.corr(), annot=True, cmap = 'cubehelix')
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_16456\3284243467.py:2: FutureWarning: The default value of numeric_only in
DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(Train_Data.corr(), annot=True, cmap = 'cubehelix')
```

```
Out[89]: <AxesSubplot: >
```



Step 05: Handle the categorical data with OneHotEncoding

After identifying the categorical attributes have to perform one hot encoding to change them into a numerical attribute.

In [91]:
```
#one hot encoding data
from sklearn.preprocessing import OneHotEncoder

numerical_cols_train = Input_cols.select_dtypes(include=np.number).columns.tolist()
categorical_cols_train = Input_cols.select_dtypes('object').columns.tolist()
```

In [92]:
```
encoder = OneHotEncoder(sparse=False,handle_unknown='ignore').fit(Input_cols[categorical_cols_train])
```

In [93]:
```
encoded_cols = list(encoder.get_feature_names_out(categorical_cols_train))
```

In [94]:
```
Input_cols[encoded_cols] =encoder.transform(Input_cols[categorical_cols_train])
Train_Inputs=Input_cols[numerical_cols_train + encoded_cols]
```

In [95]:
```
Train_Inputs
```

Out[95]:

	emp_length_int	annual_inc	loan_amount	interest_rate	age	dti	total_pymnt	installment	loan_cat	home_ownership_Mortgage	...	purpose_home_im
0	0.5	85000	25000	0.324444	21	19.48	29324.32	829.10	0	0.0	...	
1	0.5	30000	1000	0.531358	43	23.84	1207.76	35.20	0	0.0	...	
2	0.5	65000	7000	0.266173	20	14.29	8215.45	228.22	0	0.0	...	
3	10.0	189500	7000	0.531358	55	22.47	1231.90	246.38	1	1.0	...	
4	1.0	70000	25000	0.576296	18	10.50	6073.10	891.20	1	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
113275	0.5	56000	15000	0.554568	58	25.11	0.00	531.44	1	0.0	...	
113276	7.0	101000	24000	0.576296	53	22.66	0.00	596.34	1	1.0	...	
113277	4.0	98000	2400	0.359012	27	17.32	0.00	80.40	1	1.0	...	
113278	3.0	61900	6000	0.498272	21	20.32	0.00	209.20	1	1.0	...	
113279	6.0	150000	28000	0.823210	87	19.21	0.00	773.18	1	0.0	...	

113280 rows × 44 columns

After one hot handling when we check the data type it shows numerical attributes

In [96]:
```
#cheking  data types of the coulmn
Train_Inputs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113280 entries, 0 to 113279
Data columns (total 44 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   emp_length_int              113280 non-null  float64
 1   annual_inc                  113280 non-null  int64
 2   loan_amount                 113280 non-null  int64
 3   interest rate               113280 non-null  float64
```

7 Proposed Solution

7.1 Tools and Technologies

Language	**Python** – has a rich technology stack, with extensive set of libraries for machine learning.
IDE	**Jupyter –** can view both code and the results, key libraries of python already installed and can execute the codes at better speed compared to local machines. **Visual Studio Code** – it's an open-source tool, which support multiple languages and extensions.
Framework	**Flask** – Used for developing the web application using python. It is a framework with built-in development server and a fast debugger
Server/Hosting	**Heroku** – it's a cloud platform as a service, supports several languages. Further components can be scaled up and out to ensure reliable and consistent performance

7.2 Model Development

7.2.1 Prediction Model

Step 01: Separating independent and dependent variables

In [98]:
```
# specify input and output attributes
x = Train_Inputs.drop(columns=['loan_cat'], axis=1)
y = Train_Inputs['loan_cat']
```

In [99]:
```
x
```

Out[99]:

	emp_length_int	annual_inc	loan_amount	interest_rate	age	dti	total_pymnt	installment	home_ownership_Mortgage	home_ownership_None	...	pur
0	0.5	85000	25000	0.324444	21	19.48	29324.32	829.10	0.0	0.0	...	
1	0.5	30000	1000	0.531358	43	23.84	1207.76	35.20	0.0	0.0	...	
2	0.5	65000	7000	0.266173	20	14.29	8215.45	228.22	0.0	0.0	...	
3	10.0	189500	7000	0.531358	55	22.47	1231.90	246.38	1.0	0.0	...	
4	1.0	70000	25000	0.576296	18	10.50	6073.10	891.20	0.0	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
113275	0.5	56000	15000	0.554568	58	25.11	0.00	531.44	0.0	0.0	...	
113276	7.0	101000	24000	0.576296	53	22.66	0.00	596.34	1.0	0.0	...	
113277	4.0	98000	2400	0.359012	27	17.32	0.00	80.40	1.0	0.0	...	
113278	3.0	61900	6000	0.498272	21	20.32	0.00	209.20	1.0	0.0	...	
113279	6.0	150000	28000	0.823210	87	19.21	0.00	773.18	0.0	0.0	...	

113280 rows × 43 columns

In [100]:
```
y
```

Out[100]:
```
0         0
1         0
2         0
3         1
4         1
         ..
113275    1
113276    1
113277    1
113278    1
113279    1
```

Step 02: Splitting training and testing data set

```
In [101]: from sklearn.model_selection import train_test_split
          x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=50)
```

```
In [102]: x_train.head()
```

Out[102]:

	emp_length_int	annual_inc	loan_amount	interest_rate	age	dti	total_pymnt	installment	home_ownership_Mortgage	home_ownership_None	...	pur
9619	3.0	60000	7000	0.126420	48	12.74	7842.770000	218.97	0.0	0.0	...	
22060	10.0	107500	10000	0.115556	32	28.40	11218.070000	311.80	0.0	0.0	...	
5039	1.0	225000	3000	0.197531	64	11.77	3448.690000	95.85	0.0	0.0	...	
35655	2.0	44004	5000	0.176790	19	12.71	5713.859502	158.77	0.0	0.0	...	
63024	10.0	60000	6000	0.734815	74	7.34	6037.200000	223.60	1.0	0.0	...	

5 rows × 43 columns

```
In [103]: y_test.head()
```

```
Out[103]: 103342    1
          63701     1
          65078     1
          59211     1
          110520    1
          Name: loan_cat, dtype: int64
```

7.2.2 Model Training

For the loan condition dataset, the team proposed testing three model decision trees, logistic regression, and random forests because it had already been classified as a binary classification problem.

Logistic Regression

Technique for supervised learning that is applied to classification issues in order to forecast the likelihood of a target variable The model can be used in a variety of real-world situations.

In [104]:
```
#training the data set
LR_Model=LogisticRegression()
LR_Model=LR_Model.fit(x_train, y_train)

pred=LR_Model.predict(x_test)
LR_probability =LR_Model.predict_proba(x_test)[:,1]


#getting the accuracy , reacall and precission of the model using test set
accuracyLr=accuracy_score(y_test,pred)
recallLr=recall_score(y_test,pred)
precisionLr=precision_score(y_test,pred)
f1scoreLr=f1_score(y_test,pred)
AUC_LR=roc_auc_score(pred,y_test)


#print accuracy and Auc values of model
print("Accuracy : ", accuracy_score(y_test,pred)*100)
print("Precision:",precision_score(y_test,pred))
print("Recall:",recall_score(y_test,pred))
print("F1-Score:",f1_score(y_test,pred))
print("ROC_AUC Score:",AUC_LR)
```

```
Accuracy :  87.33227401129943
Precision: 0.8782168025971969
Recall: 0.8927795218546245
F1-Score: 0.8854382883602108
ROC_AUC Score: 0.8726877370919564
```

In [46]:
```
#GETTING CLASSIFICATION REPORT
print(classification_report(pred,y_test))
```

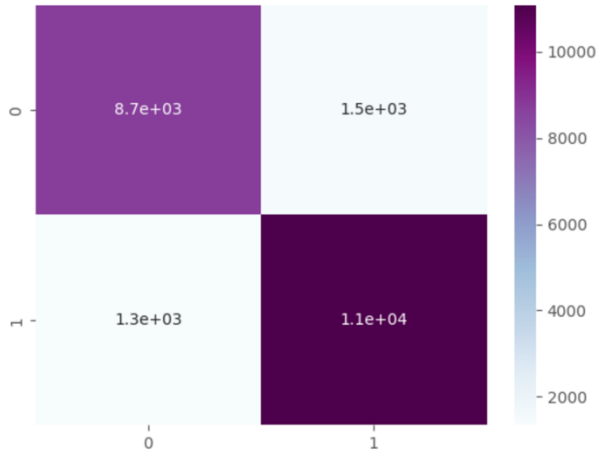
```
              precision    recall  f1-score   support

           0       0.85      0.87      0.86     10095
           1       0.89      0.88      0.89     12561

    accuracy                           0.88     22656
   macro avg       0.87      0.87      0.87     22656
weighted avg       0.88      0.88      0.88     22656
```

ROC Curve

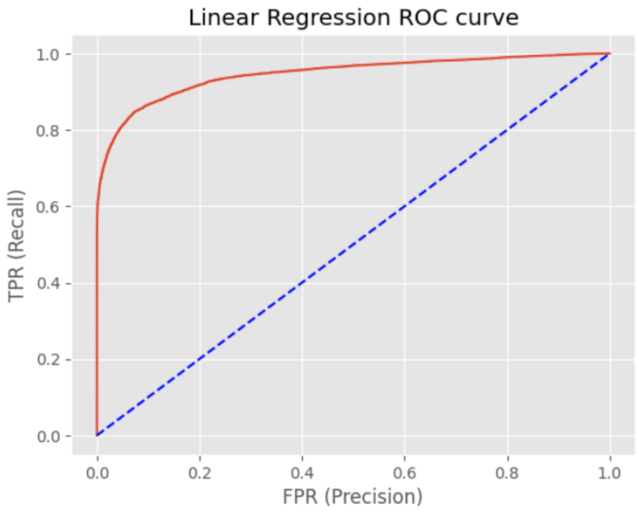
In [47]:
```
cm=confusion_matrix(y_test,pred)
print(cm)
sns.heatmap(cm,annot=True,cmap='BuPu')
```

```
[[ 8748  1485]
 [ 1347 11076]]
```

Out[47]: <AxesSubplot: >



In [48]:
```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(y_test, LR_probability)

plt.title('Linear Regression ROC curve')
plt.xlabel('FPR (Precision)')
plt.ylabel('TPR (Recall)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='blue')
plt.show()
```



Random Forest Classifier

The forest created using the supervised learning approach is an aggregation of decision trees that were typically trained using the bagging method. The bagging method combines various learning models to improve final outcomes.

In [49]:
```
from sklearn.ensemble import RandomForestClassifier
#Training the model
randomForest_Model = RandomForestClassifier()
randomForest_Model=randomForest_Model.fit(x_train, y_train)
y_pred = randomForest_Model.predict(x_test)
RF_probability = randomForest_Model.predict_proba(x_test)[:,1]


#getting the accuracy , reacall and precission of the model using test set
AUC_RF=roc_auc_score(y_pred,y_test)
acc_rf=accuracy_score(y_test,y_pred)
recall_rf=recall_score(y_test,y_pred)
precision_rf=precision_score(y_test,y_pred)
f1score_rf=f1_score(y_test,y_pred)

print("Accuracy : ", accuracy_score(y_test,y_pred)*100)
print("Precision:",precision_score(y_test,y_pred))
print("Recall:",recall_score(y_test,y_pred))
print("F1-Score:",f1_score(y_test,y_pred))
print("ROC_AUC Score:",AUC_RF)
```

```
Accuracy :  94.23993644067797
Precision: 0.9638684913217623
Recall: 0.9298076149078323
F1-Score: 0.9465317326996353
ROC_AUC Score: 0.9410796729472379
```

In [50]:
```
#PRINT CLASSIFIACTION REPORT
print(classification_report(pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.85      0.87      0.86     10095
           1       0.89      0.88      0.89     12561

    accuracy                           0.88     22656
   macro avg       0.87      0.87      0.87     22656
weighted avg       0.88      0.88      0.88     22656
```

Decision Tree Classifier

For classification and prediction, supervised learning with a pre-determined target variable is the best option.

In [51]:
```
# decision tree classifier
from sklearn.tree import DecisionTreeClassifier
DT_model = DecisionTreeClassifier()
#Training the model
DT_model.fit(x_train, y_train)

y_pred = DT_model.predict(x_test)
RF_probability = DT_model.predict_proba(x_test)[:,1]


#getting the accuracy , reacall and precission of the model using test set
AUC_DT=roc_auc_score(y_pred,y_test)
acc_DT=accuracy_score(y_test,y_pred)
recall_DT=recall_score(y_test,y_pred)
precision_DT=precision_score(y_test,y_pred)
f1score_DT=f1_score(y_test,y_pred)

print("Accuracy : ", accuracy_score(y_test,y_pred)*100)
print("Precision:",precision_score(y_test,y_pred))
print("Recall:",recall_score(y_test,y_pred))
print("F1-Score:",f1_score(y_test,y_pred))
print("ROC_AUC Score:",AUC_RF)
```

```
Accuracy :  93.71027542372882
Precision: 0.9433247339567882
Recall: 0.9418819930773565
F1-Score: 0.9426028114552706
ROC_AUC Score: 0.9410796729472379
```

In [52]:
```
#PRINT CLASSIFIACTION REPORT
print(classification_report(pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.85      0.87      0.86     10095
           1       0.89      0.88      0.89     12561

    accuracy                           0.88     22656
   macro avg       0.87      0.87      0.87     22656
weighted avg       0.88      0.88      0.88     22656
```

Comparison among the models

The team did take into consideration variables like prediction accuracy when choosing the optimal model for the given case.

In [56]:
```
ind=['Logistic regression','Randomforest','Decision Tree']
data={"Accuracy":[accuracyLr,acc_rf,acc_DT],"Recall":[recallLr,recall_rf,recall_DT],"Precision":[precisionLr,precision_
    'f1_score':[f1scoreLr,f1score_rf,f1score_DT],"ROC_AUC":[AUC_LR,AUC_RF,AUC_DT]}
result=pd.DataFrame(data=data,index=ind)
result
```

Out[56]:

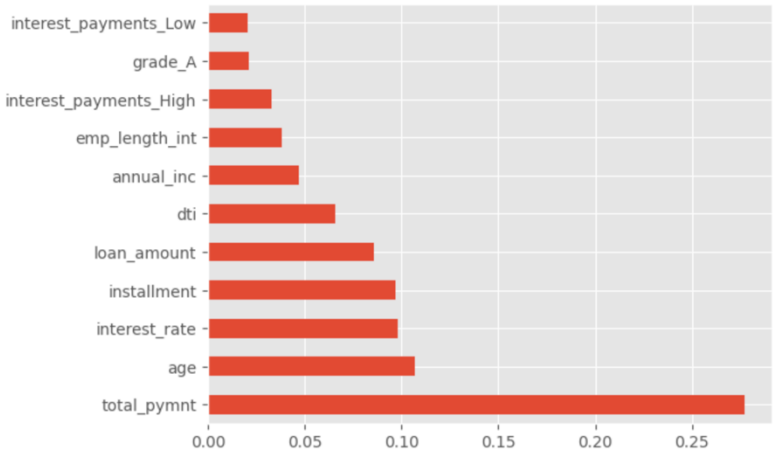
	Accuracy	Recall	Precision	f1_score	ROC_AUC
Logistic regression	0.875000	0.891572	0.881777	0.886647	0.874172
Randomforest	0.942399	0.929808	0.963868	0.946532	0.941080
Decision Tree	0.937103	0.941882	0.943325	0.942603	0.936450

The team decided to go with the Random Forest model. Additionally, the choice is made with the team's project in mind.

After building the model, to assess the relevance of the features selected, feature importance was performed.

```
In [57]: from sklearn.ensemble import ExtraTreesClassifier
Model = ExtraTreesClassifier()
Model.fit(x,y)
#use inbuilt class feature_importances of tree based class
print(Model.feature_importances_)

#plot of feature importances for best visualization
feat_importances = pd.Series(Model.feature_importances_, index=x.columns)
feat_importances.nlargest(11).plot(kind='barh')
plt.show()
```

```
[3.79514601e-02 4.67923333e-02 8.54303751e-02 9.79709267e-02
 1.06946266e-01 6.60078522e-02 2.77301633e-01 9.70890624e-02
 6.95102234e-03 3.18962791e-05 3.43795445e-04 4.80282931e-03
 7.02951437e-03 1.21639124e-03 3.82483961e-03 3.37640851e-03
 6.86440407e-03 5.70389362e-03 1.48337997e-06 1.38200671e-06
 2.62429776e-03 6.28540007e-03 8.57942179e-03 8.09473240e-04
 1.14319884e-03 2.85189682e-03 1.81135907e-03 1.17057200e-03
 4.35008492e-03 3.68512612e-04 2.71057600e-03 1.14392079e-03
 1.94905507e-03 3.52135622e-03 3.30628109e-02 2.03420465e-02
 2.09138983e-02 9.27578717e-03 6.75004961e-03 6.04149980e-03
 4.63697292e-03 2.85372499e-03 1.16631539e-03]
```

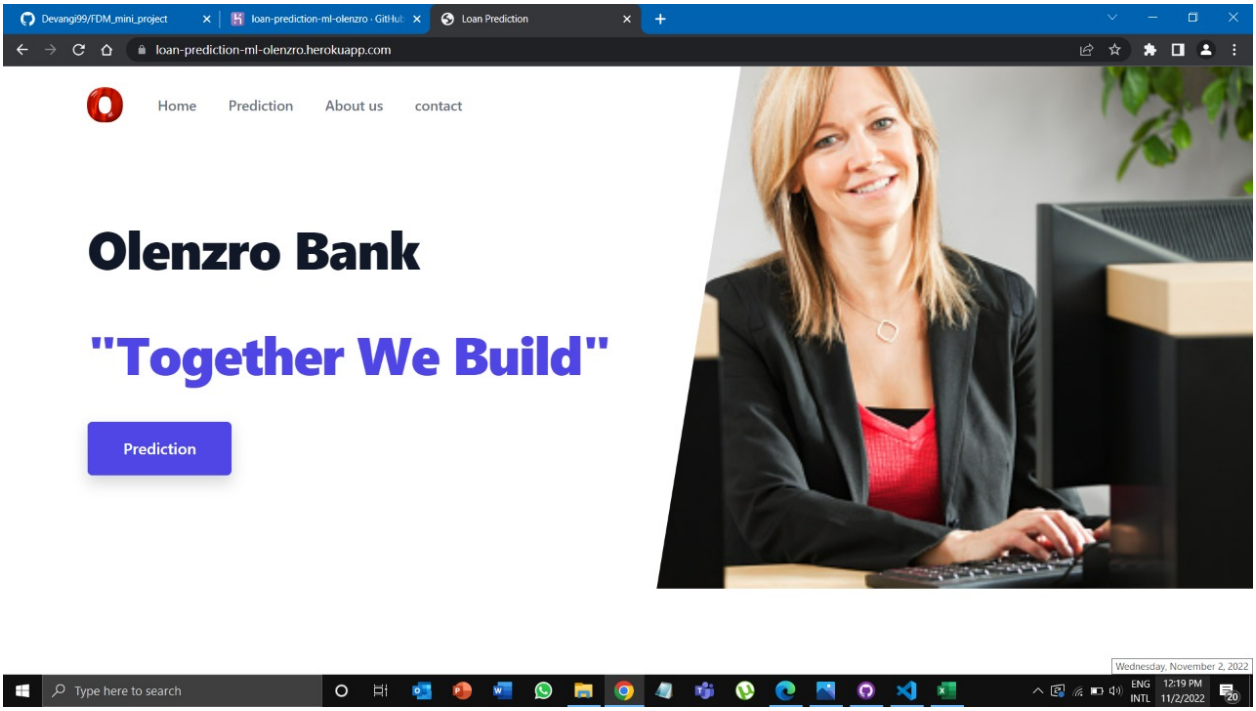


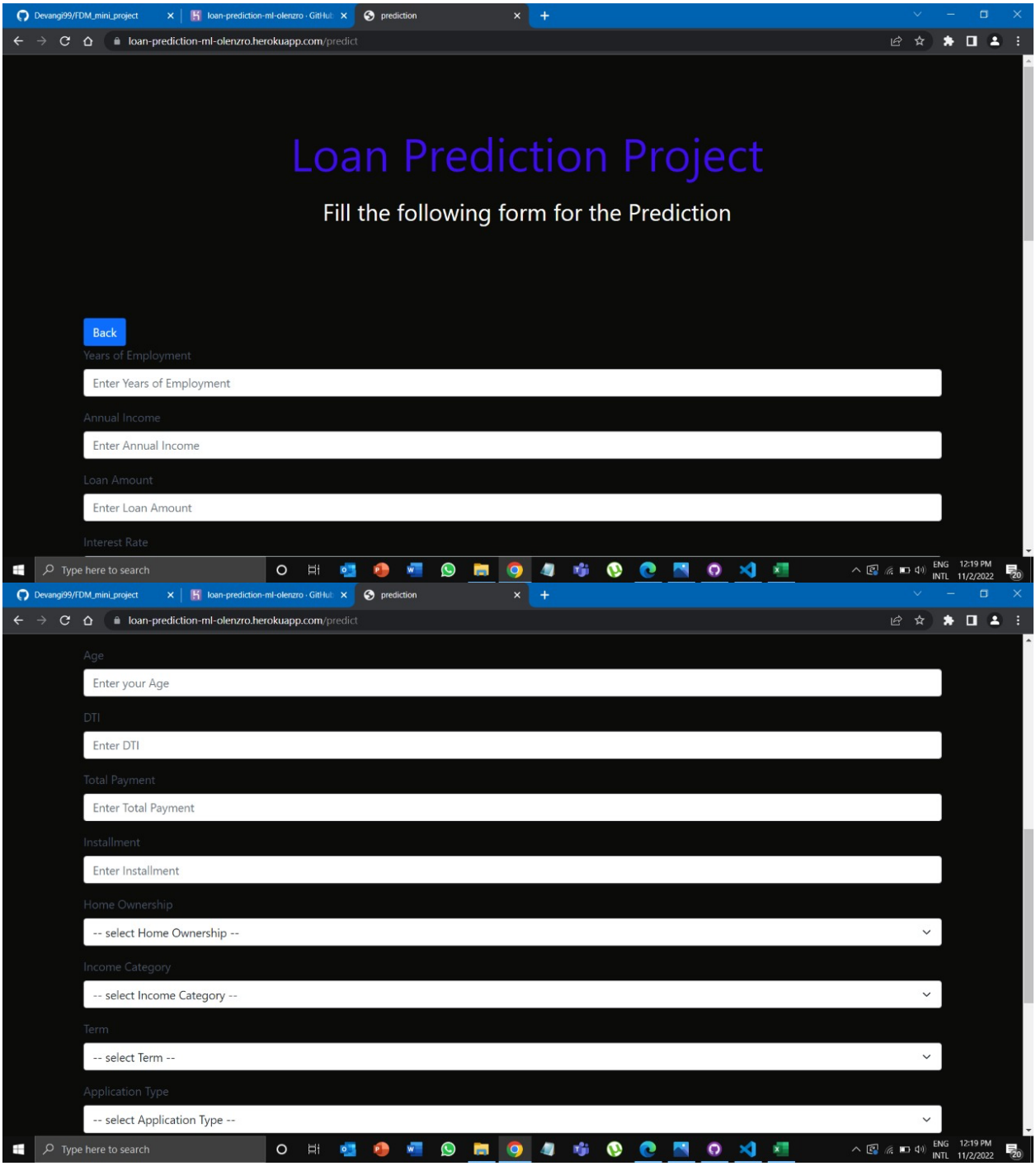
7.3 Application Development

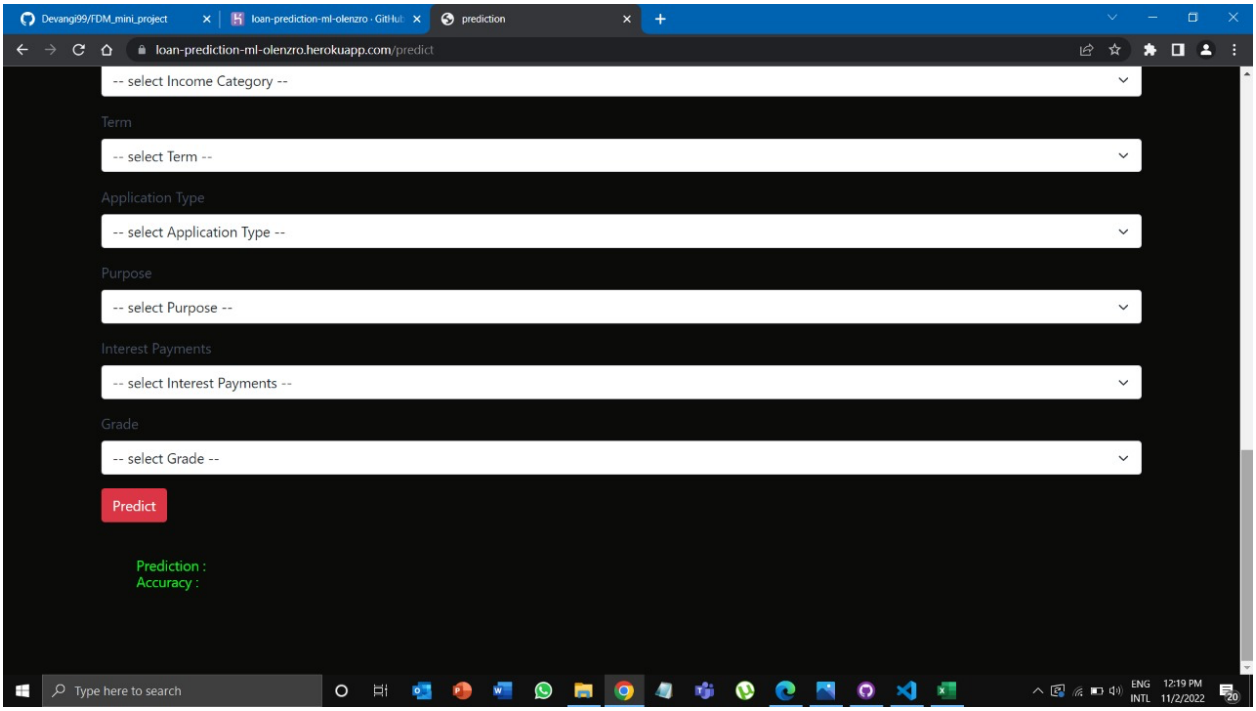
The application was developed differently than prior projects since it incorporates a machine learning model into a real-world scenario. The development focuses on elements like arranging the integration of models into the program and building user interfaces.

7.4 UIs of solution

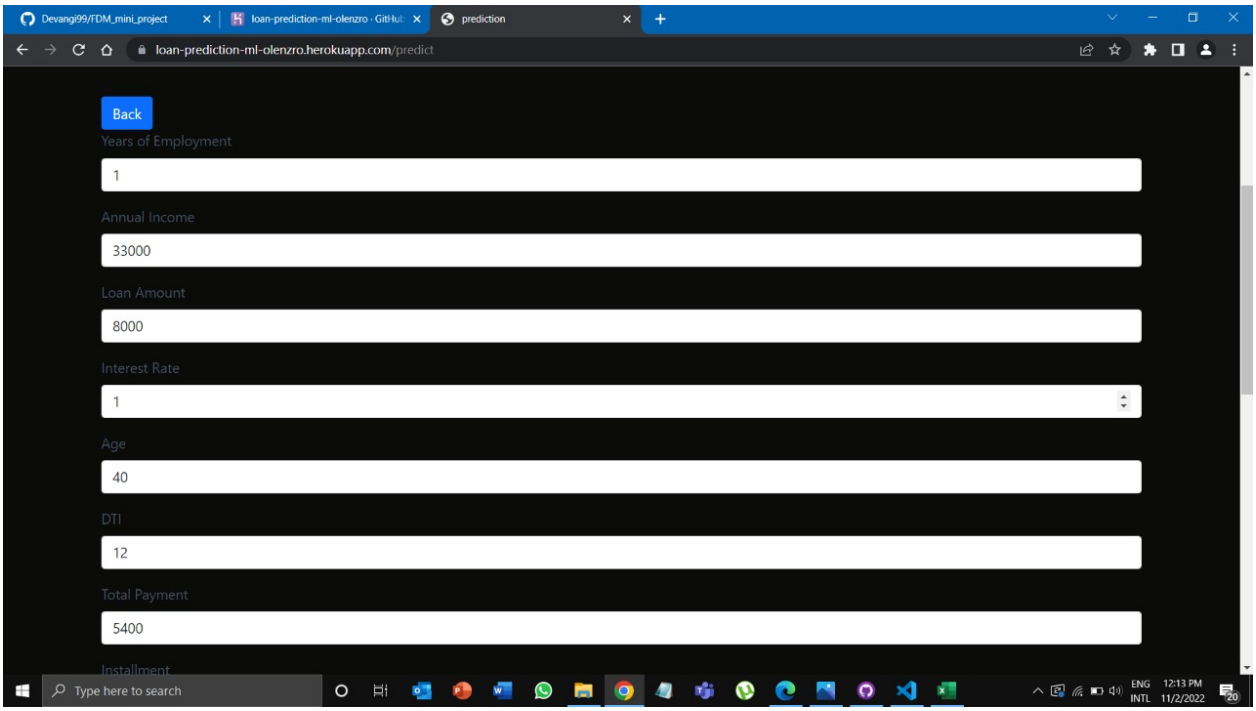
The following screenshots illustrates the user interfaces of the final application deployed to the server.



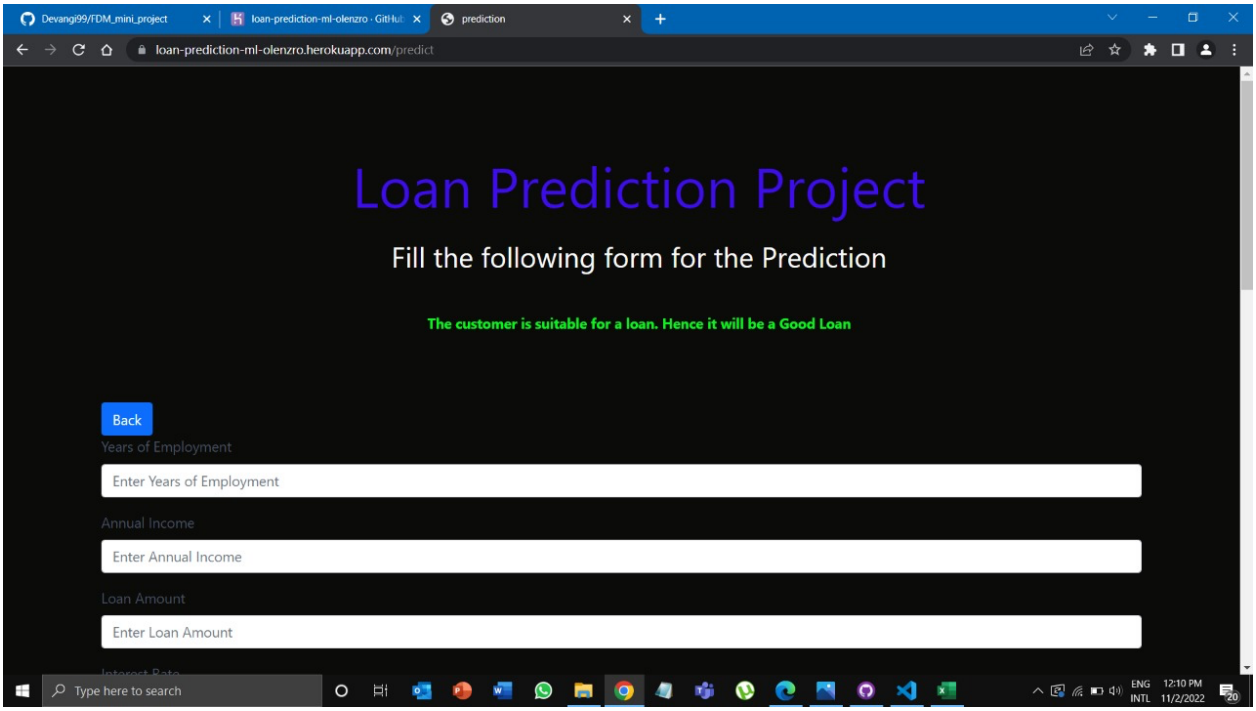

Devangi99/FDM_mini_project
loan-prediction-ml-olenzro · GitHub
prediction
loan-prediction-ml-olenzro.herokuapp.com/predict
Loan Prediction Project
Fill the following form for the Prediction
Back
Years of Employment
Enter Years of Employment
Annual Income
Enter Annual Income
Loan Amount
Enter Loan Amount
Interest Rate
Type here to search
ENG
INTL
12:19 PM
11/2/2022
Devangi99/FDM_mini_project
loan-prediction-ml-olenzro · GitHub
prediction
loan-prediction-ml-olenzro.herokuapp.com/predict
Age
Enter your Age
DTI
Enter DTI
Total Payment
Enter Total Payment
Installment
Enter Installment
Home Ownership
-- select Home Ownership --
Income Category
-- select Income Category --
Term
-- select Term --
Application Type
-- select Application Type --
Type here to search
ENG
INTL
12:19 PM
11/2/2022



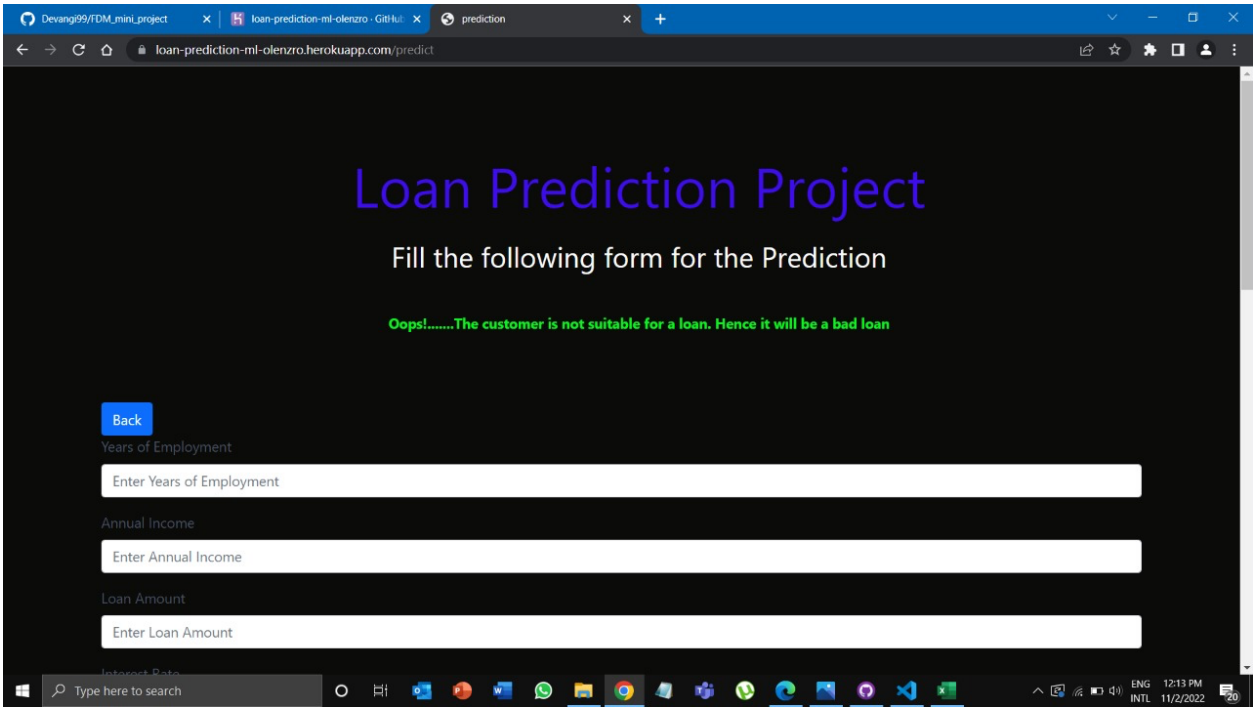
7.5 Solution Overview with demo



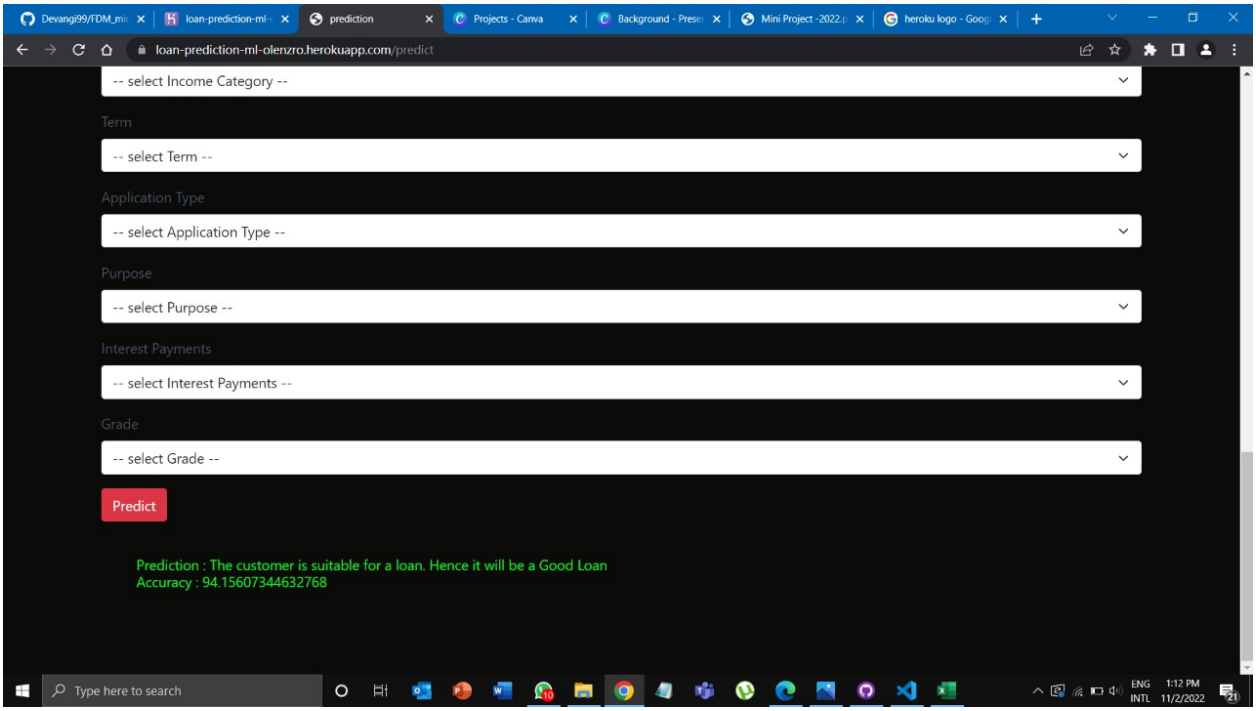
After filling the form if the customer suitable for a loan, this will be shown



After filling the form if the customer not suitable for a loan, this will be shown



Accuracy will be displayed shown as below



7.6 Constraints and limitations

Only two operational areas of the bank are the focus of the current application, and the dataset is only available for a limited time period. Furthermore, security features like user authentication are not given much attention in the program. Since the team is using a free form of hosting for the application, there are restrictions in terms of data loading and application loading time.

7.7 Future Enhancements

The security features of the program will treat better focus in the following iteration, with user access being generated and limited in accordance with operations. The team will then concentrate on finding alternative alternatives with the goal of improving application performance. Integration of additional bank processes would help the management make decisions more effectively. Finally, in the upcoming phase, advanced features like a loan recommendation system based on credit score will be implemented because at the moment, once a loan is identified as a bad loan, the bank is unable to offer an alternative loan that may meet their needs and because the bank does not want to lose any potential customers. These

feature implementations will only be feasible if the bank begins to consistently capture or is currently doing so for the necessary data.

8 Project Management & Methodology

8.1 Methodology

Agile has grown into a popular project management technique. The project has been divided into important phases such as plan, design, develop, test, release, and feedback in accordance with the approach that was chosen.

Plan:

- Identify requirements
- Define scope
- Identify and preprocess datasets
- Selection of tools and technologies

Design:

- Plan the web application UI
- Plan model integration

Develop:

- Develop machine learning models
- Develop web application
- Integrate models to application

Test:

- Test accuracy of data model

Release:

- Host application of different versions

Feedback:

- Obtain feedback from testing team
- Review feedback on model accuracy and web application

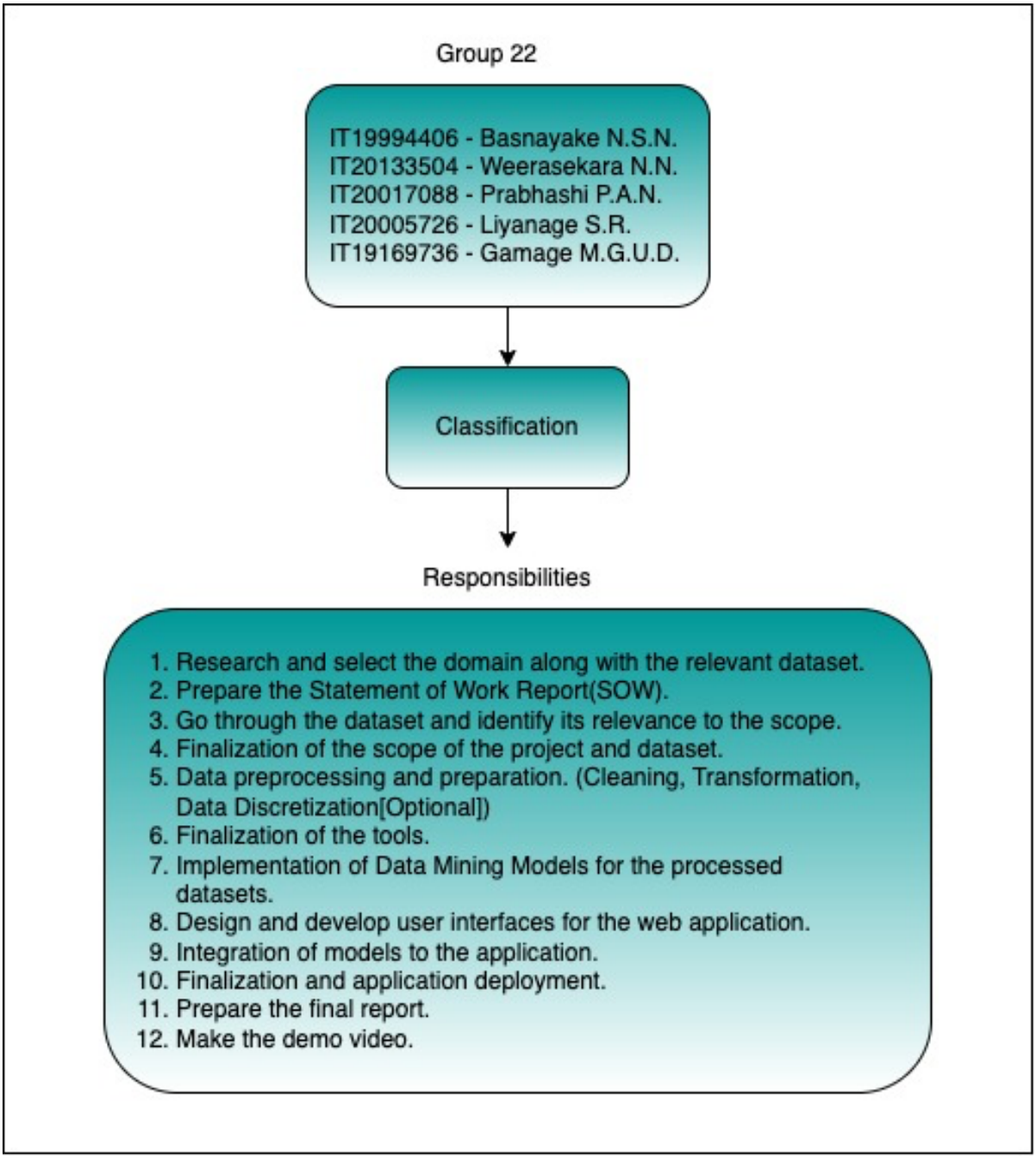
8.2 Deliverables

Delivery	Available at	Description
Statement of Work (SOW) Report	Already submitted on CourseWeb	This report includes important details about the project's history, problem definition, scope of work, tools and methodologies used, scheduled activities, timeframe, and deliverables, as well as the duties carried out by each group member.
Data Mining Models	-	The model for the chosen data sets that would carry out the tasks (predictive or segmentation). This might be a file that just contains the code necessary to run the produced models.
Web Application	https://loan-prediction-ml-olenzro.herokuapp.com/	Users may evaluate a hosted web application to carry out their tasks. The model or models are included in the web application.
Video	-	Presentation of the application's functionality.
Final Report	Submitted on CourseWeb	Information about the project domain, the produced solution, and test cases are included in the report.
Git Hub Repository	https://github.com/Devangi99/FDM_mini_project.git	Contain application source codes with version management

8.3 Team Management (Members & Responsibilities)

Member	Role	Contribution	Responsibilities
Basnayake N.S.N.	Team Lead	100%	• Share the project plan and timeline with group members and assign them. • Manage the project progress.
Weerasekara N.N.	Technical Expert	100%	• Find the appropriate technologies. • Implement the application by solving the technology problems.
Prabhashi P.A.N.	Independent Tester	100%	• Test in different environments. • Create test case scenarios.
Liyanage S.R.	Domain Expert	100%	• Find the best data mining models/algorithms. • Improve the accuracy of the model.
Gamage M.G.U.D.	Integrator	100%	• Integrate the built application with the data mining tools. • Check that every task is performed out according to plan.

Further, the diagram below illustrates main responsibilities of the assignment, where all the team members perform all the tasks specified in the responsibilities.



8.4 Project Plan and Timeline

Type	Title	Start date	End date	Duration	% Complete
Task	Forming a group	18th Sept	20th Sept	3	100
Task	Finalization of Scope of project	21st Sept	22nd Sept	2	100
Task	Finalization of Dataset	23rd Sept	23rd Sept	1	100
Milestone	Group Registration & Dataset Submission	24th Sept	24th Sept	-	100
Task	Preparing Statement of Work (SOW)	7th Oct	9th Oct	3	100
Task	Getting approval on SOW	9th Oct	10th Oct	2	100
Milestone	Submission of SOW	10th Oct	10th Oct	-	100
Task	Data Preparation & Preprocessing	11th Oct	15th Oct	5	100
Milestone	Finalizing Tools & Models	15th Oct	15th Oct	-	100
Task	Development of Model	16th Oct	2nd Oct	15	100
Task	Design & Development of UI	17th Oct	2nd Oct	14	100
Task	Finalization & Application Deployment	18th Oct	2nd Oct	13	100
Task	Preparing Final Report	23rd Oct	2nd Oct	8	100
Task	Making Demo Video	30th Oct	2nd Oct	4	100
Milestone	Submission of Final Report, Application & Video	2nd Oct	2nd Oct	-	100

ID	Name	Start Date	End Date	Duration
1	Forming a group	Sep 18, 2022	Sep 20, 2022	3 days
2	Finalization of Scope of project	Sep 21, 2022	Sep 22, 2022	2 days
3	Finalization of Dataset	Sep 23, 2022	Sep 23, 2022	1 day
4	Group Registration & Dataset Submission	Sep 24, 2022	Sep 24, 2022	0 days
5	Preparing Statement of Work (SOW)	Oct 07, 2022	Oct 09, 2022	3 days
6	Getting approval on SOW	Oct 09, 2022	Oct 10, 2022	2 days
7	Submission of SOW	Oct 10, 2022	Oct 10, 2022	0 days
8	Data Preparation & Preprocessing	Oct 11, 2022	Oct 15, 2022	5 days
9	Finalizing Tools & Models	Oct 15, 2022	Oct 15, 2022	0 days
10	Development of Model	Oct 16, 2022	Oct 30, 2022	15 days
11	Design & Development of UI	Oct 17, 2022	Oct 30, 2022	14 days
12	Finalization & Application Deployment	Oct 18, 2022	Oct 30, 2022	13 days
13	Preparing Final Report	Oct 23, 2022	Oct 30, 2022	8 days
14	Making Demo Video	Oct 27, 2022	Oct 30, 2022	4 days
15	Submission of Final Report, Application & Video	Oct 30, 2022	Oct 30, 2022	0 days

9 References

Carbajo, M. (2022, August 11). *7 Steps To Take When Getting a Business Loan From a Bank*. Retrieved from www.thebalancemoney.com: https://www.thebalancemoney.com/business-loan-steps-3953768

JiyoonLee. (2022, August 22). Do firms use credit lines to support investment opportunities?: Evidence from success in R&D. *Journal of Empirical Finance*, 1-14.

Team, C. (2022, January 27). *Major Risks for Banks*. Retrieved from corporatefinanceinstitute.com: https://corporatefinanceinstitute.com/resources/knowledge/finance/major-risks-for-banks/