Predicting Personal Loan Approval Using Machine Learning

1.INTRODUCTION

Now-a-days obtaining loans from banks have become a very common phenomenon. The banks gain profits from the loans lent to their customers in the form of interest. While approving a loan, the banks should consider many factors such as credit history and score, reputation of the person, the location of the property and the relationship with the bank.

Many people apply for loans in the name of home loan, car loan and many more. Everyone cannot be approved based on above mentioned conditions. There are so many cases where applicant's applications for loans are not approved by various finance companies. The right predictions whether to give a loan to a customer or not is very important for the banks to maximize the profits. The idea behind this project is to use Machine Learning to predict whether a customer can get a loan from a bank or not.

1.1Overview:

A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest. There are various types of loans available to individuals and businesses, such as personal loans, mortgages, auto loans, student loans, business loans and many more. They are offered by banks, credit unions, and other financial institutions, and the terms of the loan, such as interest rate, repayment period, and fees, vary depending on the lender and the type of loan.

A personal loan is a type of unsecured loan that can be used for a variety of expenses such as home repairs, medical expenses, debt consolidation, and more. The loan amount, interest rate, and repayment period vary depending on the lender and the borrower's credit worthiness. To qualify for a personal loan, borrowers typically need to provide proof of income and have a good credit score.

Predicting personal loan approval using machine learning analyses a borrower's financial data and credit
history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

1.2 Purpose:

Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

However, the benefits can only be reaped if the bank has a robust model to accurately predict which customer's loan it should approve and which to reject, in order to minimize the risk of loan default.

1.3. PREDICTIVE MODELLING

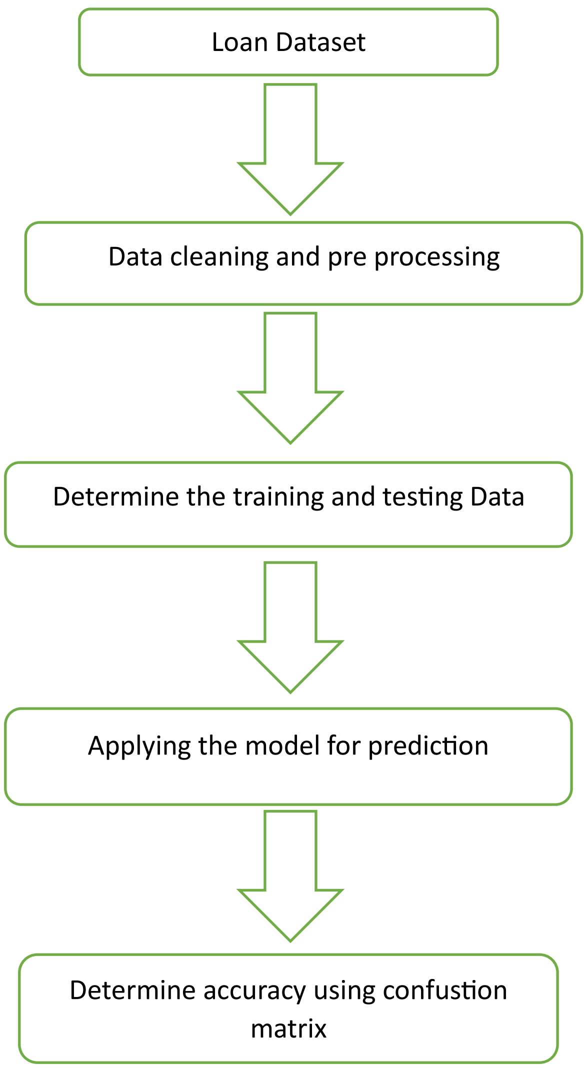
Predictive modeling is used to analyze the data and predict the outcome. Predictive modeling used to predict the unknown event which may occur in the future. In this process, we are going to create, test and validate the model.

There are different methods in predictive modeling. They are learning, artificial intelligence and statistics. Once we create a model, we can use many times, to determine the probability of outcomes.

So, predict model is reusable. Historical data is used to train an algorithm. The predictive modeling process is an iterative process and often involves training the model, using multiple models on the same dataset.

- Creating the model: To create a model to run one or more algorithms on the data set.
- Testing a model: The testing is done on past data to see how the best model predicts
- Validating a model: Using visualization tools to validate the model.
- Evaluating model: Evaluating the best fit model from the models used and choosing the model right fitted for the data.

1.4.PROCESS MODE USED:


Loan Dataset
Data cleaning and pre processing
Determine the training and testing Data
Applying the model for prediction
Determine accuracy using confustion matrix

When a customer demands credit from a bank, the bank should evaluate the credit demand as soon as possible to gain competitive advantage. Additionally, for each credit demand, the same process is repeated and constitutes a cost for the bank.

- Load the data.
- Determine the training and testing data.
- Data cleaning and preprocessing.
- Fill the missing values with mean values regarding numerical values.
- Fill the missing values with mode values regarding categorical variables.
- Outlier treatment.
- Apply the modeling for prediction.
- Removing the load identifier.
- Create the target variable (based on the requirement). In this approach,
- Target variable is loan-status.
- Create a dummy variable for categorical variable (if required) and split the
- Training and testing data for validation.
- Apply the model
- LR method
- RF method
- SVM method
- Determine the accuracy followed by confusion matrix

1.5.Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

✓ Define Problem / Problem Understanding

✓ Data Collection & Preparation

✓ Exploratory Data Analysis

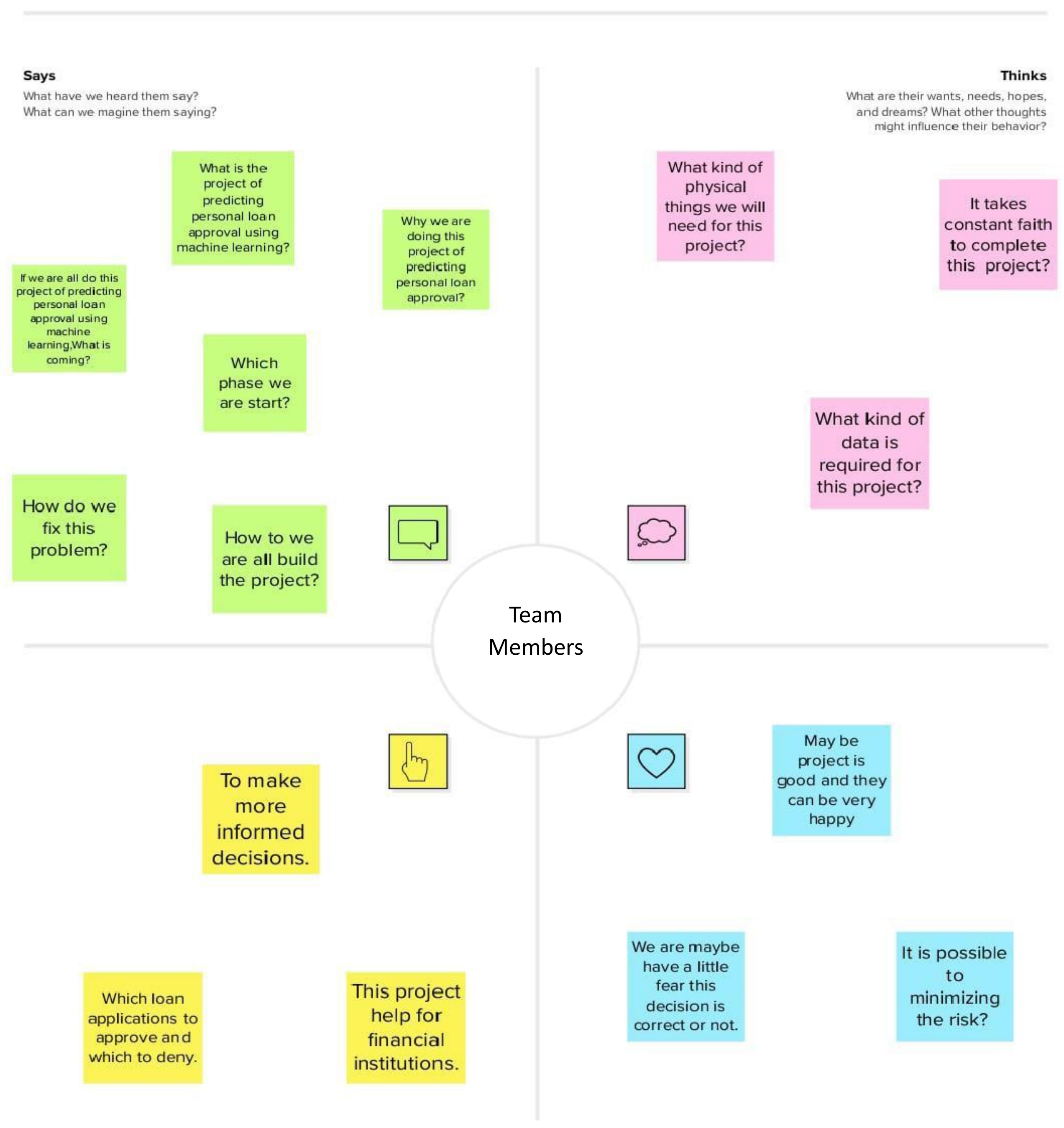
✓ Model Building

✓ Performance Testing & Hyperparameter Tuning

✓ Model Deployment

2.PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map

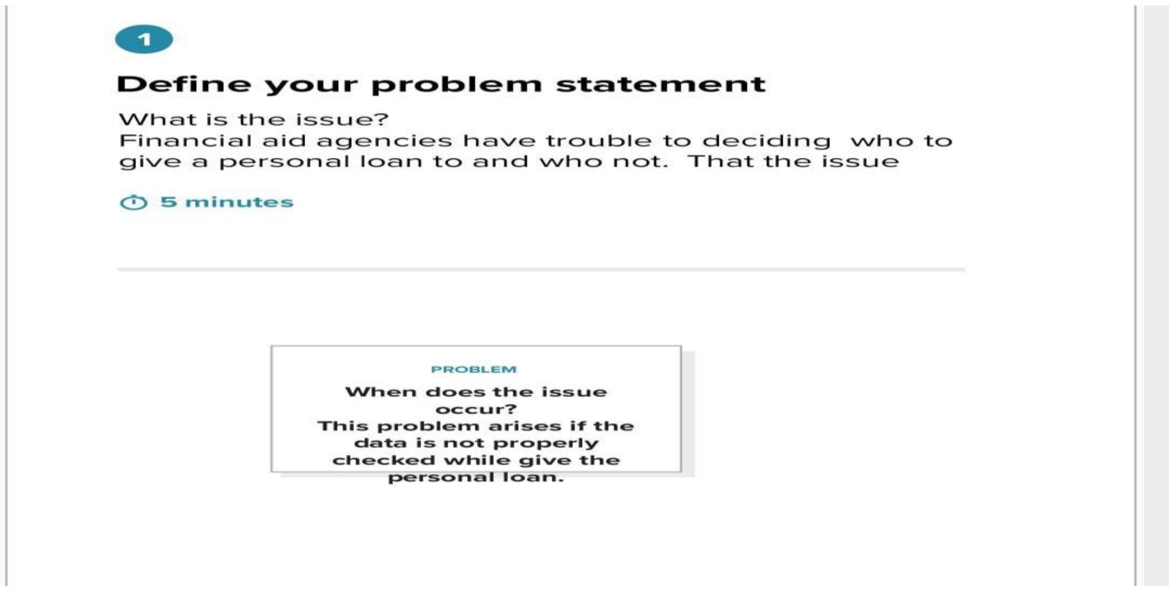


Says
What have we heard them say?
What can we magine them saying?
What is the project of predicting personal loan approval using machine learning?
Why we are doing this project of predicting personal loan approval?
If we are all do this project of predicting personal loan approval using machine learning,What is coming?
Which phase we are start?
How do we fix this problem?
How to we are all build the project?
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
What kind of physical things we will need for this project?
It takes constant faith to complete this project?
What kind of data is required for this project?
Team Members
To make more informed decisions.
Which loan applications to approve and which to deny.
This project help for financial institutions.
May be project is good and they can be very happy
We are maybe have a little fear this decision is correct or not.
It is possible to minimizing the risk?

In the above Mural picture Our Group done the map with our ideas about Optimizing Spam Filtering with Machine Learning. We talk about the topic from users side what they Says, Thinks, Feels and Does. We added the screenshot of our Mural map.

2.2 Ideation & Brainstroming

In the below we added Screenshot of our Brainstorm and Ideation. In this template we give our brainstorm ideas about our project.



1

Define your problem statement

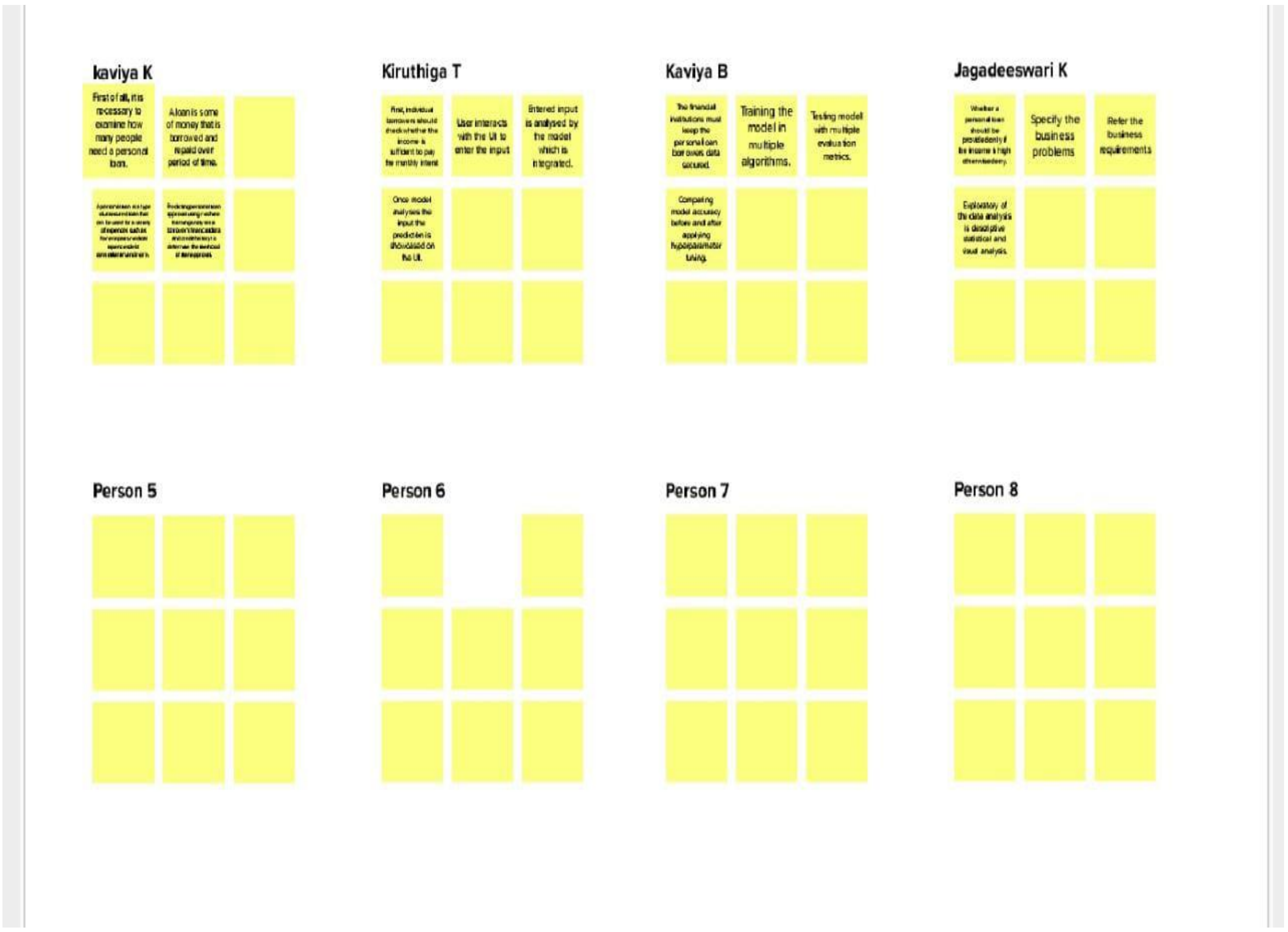
What is the issue?
Financial aid agencies have trouble to deciding who to give a personal loan to and who not. That the issue

5 minutes

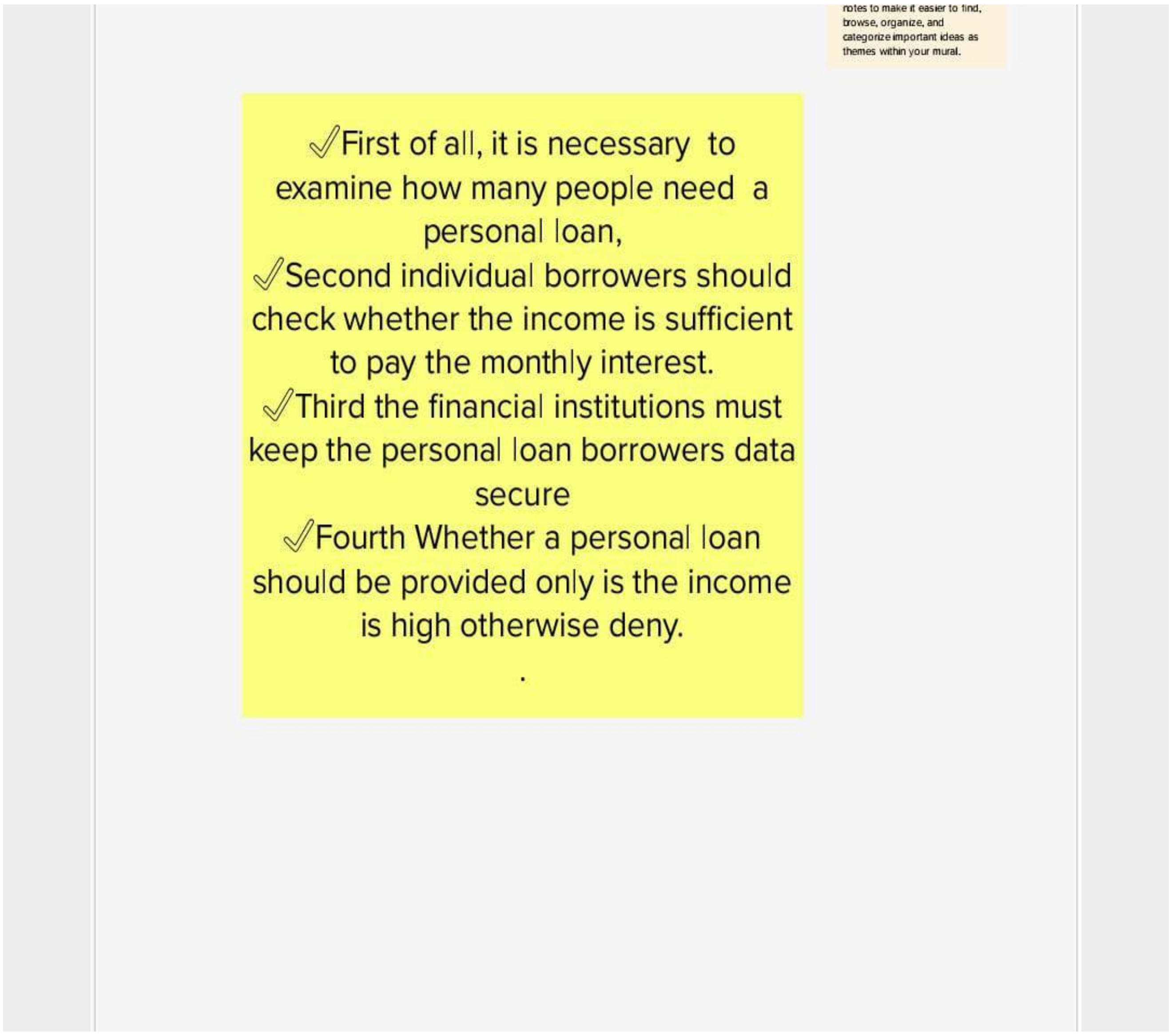
PROBLEM

When does the issue occur?
This problem arises if the data is not properly checked while give the personal loan.

BRAINSTROM

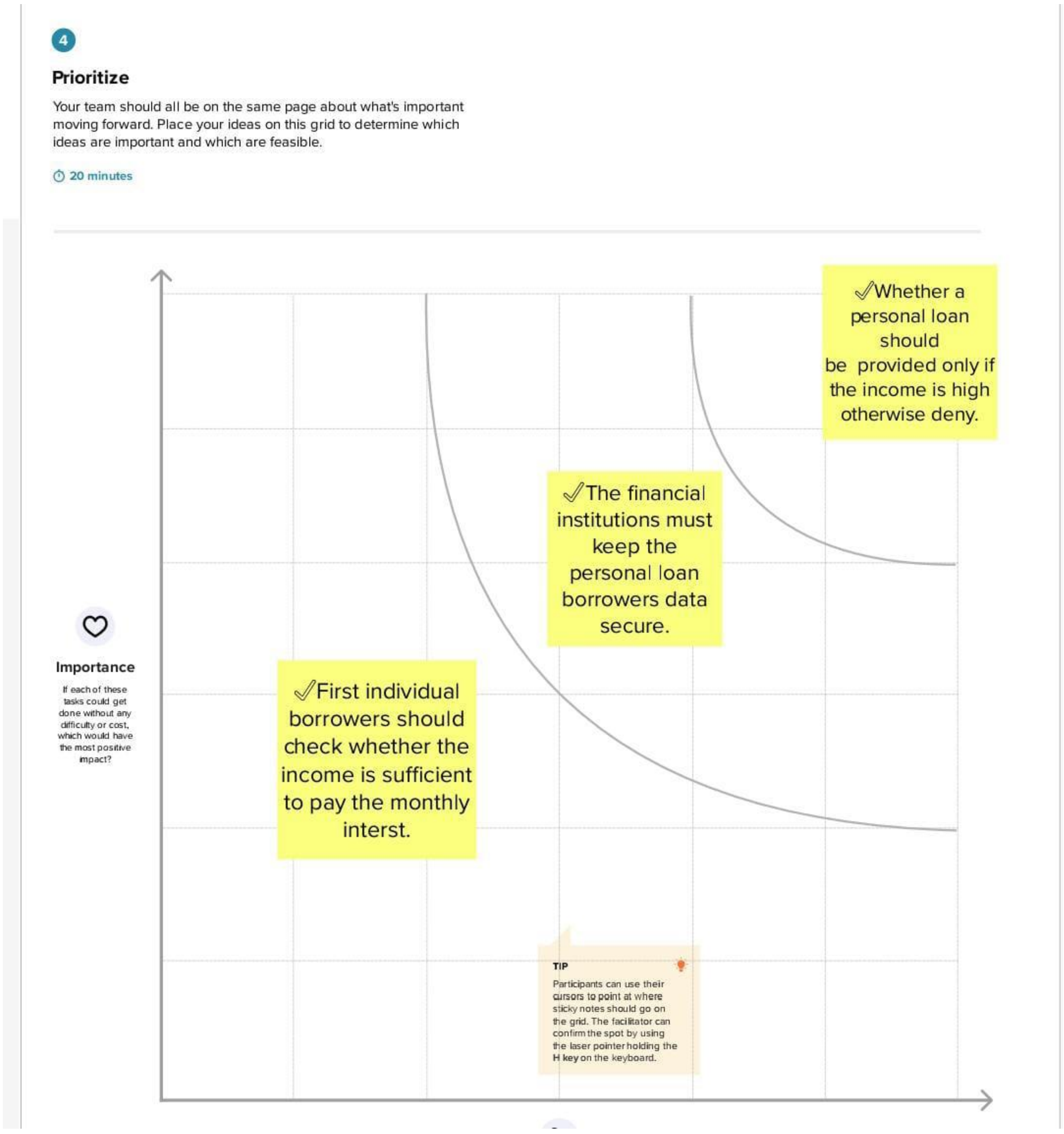

kaviya K
First of all, it is necessary to examine how many people need a personal loan.
A loan is some of money that is borrowed and repaid over period of time.
Kiruthiga T
First, individual borrowers should check whether the income is sufficient to pay the monthly interest
User interacts with the UI to enter the input
Entered input is analysed by the model which is integrated.
Once model analyses the input the prediction is showcased on the UI.
Kaviya B
The financial institutions must keep the personal loan borrowers data secured.
Training the model in multiple algorithms.
Testing model with multiple evaluation metrics.
Comparing model accuracy before and after applying hyperparameter tuning.
Jagadeeswari K
Specify the business problems
Refer the business requirements
Exploratory of the data analysis is descriptive statistical and visual analysis.
Person 5
Person 6
Person 7
Person 8

GROUP IDEAS


notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

✓First of all, it is necessary to examine how many people need a personal loan,
✓Second individual borrowers should check whether the income is sufficient to pay the monthly interest.
✓Third the financial institutions must keep the personal loan borrowers data secure
✓Fourth Whether a personal loan should be provided only is the income is high otherwise deny.
.

PRIORITIZE



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⏱ 20 minutes

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

✓Whether a personal loan should be provided only if the income is high otherwise deny.

✓The financial institutions must keep the personal loan borrowers data secure.

✓First individual borrowers should check whether the income is sufficient to pay the monthly interst.

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

2.3.SPECIFY THE BUSINESS PROBLEM

The problem of predicting personal loan approval involves using data analysis and machine learning algorithms to predict whether a borrower will be approved for a personal loan or not. This problem is important for both lenders and borrowers. Lenders need to accurately assess the risk of lending money to borrowers, while borrowers need to understand their chances of getting approved and prepare accordingly.

The problem of predicting personal loan approval involves analyzing various factors such as the borrower's credit history, income, employment status, debt-to-income ratio, loan amount, and loan purpose, among others. Machine learning algorithms such as logistic regression, decision trees, and neural networks can be used to analyze these factors and predict whether a borrower will be approved or not.

The accuracy of the prediction model depends on the quality and quantity of data used for training and testing the model. Therefore, it is important to have a large and diverse dataset to ensure that the model is robust and can generalize well to new data. Additionally, feature engineering, data preprocessing, and model selection are important steps in building an accurate prediction model.

Overall, the problem of predicting personal loan approval is an important and challenging task that requires careful analysis and modeling to ensure accurate predictions.

Business requirements

Personal loan approval prediction using machine learning can be a useful tool for businesses in the lending industry. By leveraging historical data and statistical algorithms, machine learning models can identify patterns and make accurate predictions about the likelihood of a loan being approved.

The business environment for predicting personal loan approval involves utilizing machine learning and data analysis techniques to assess the credit worthiness of loan applicants.

The goal is to accurately predict the likelihood of an applicant's loan being approved based on factors such as their credit score, income, employment history, and debt-to-income ratio. This technology can benefit financial institutions, such as banks and credit unions, by streamlining the loan approval process and reducing the risk of default on loans. By accurately assessing an applicant's creditworthiness, financial institutions can make more informed decisions about whether to approve a loan, how much to lend, and at what interest rate. Additionally, by leveraging data analytics and machine learning algorithms, financial institutions can improve their customer service by providing a faster and more convenient loan application process.

This can help attract more customers and increase customer satisfaction. Overall, the business environment for predicting personal loan approval involves using cutting-edge technology to improve the accuracy, speed, and convenience of the loan approval process, benefiting both financial institutions and their customers.

2.4.Literature Survey

"Loan Prediction using Decision Tree and Random Forest" Author-Kshitiz Gautam, Arun Pratap Singh, Keshav Tyagi, Mr. Suresh Kumar Year-2020. In India the number of people or organization applying for loan gets increased every year. The bank have to put in a lot of work to analyses or predict whether the customer can pay back the loan amount or not (defaulter or non-defaulter) in the given time.

The aim of this paper is to find the nature or background or credibility of client that is applying for the loan. We use exploratory data analysis technique to deal with problem of approving or rejecting the loan request or in short loan prediction.

The main focus of this paper is to determine whether the loan given to a particular person or an organization shall be approved or not. “A Comparative Study of Machine Learning Algorithms for Personal Loan Approval Prediction" by S. Chakraborty and S. Chakraborty (2020) - This study compares the performance of logistic regression, decision tree, random forest, and support vector machine algorithms for predicting personal loan approval. The authors found that random forest and support vector machine algorithms had the highest accuracy in predicting loan approval. "Predicting Personal Loan Approval Using Machine Learning Techniques" by R. Gupta and S. Bansal (2019) - This study uses a dataset of personal loan applications to predict loan approval using logistic regression and decision tree algorithms. The authors found that logistic regression had higher accuracy in predicting loan approval than decision trees.

In 2019, Jency, Sumathi and Shiva Sri [2] proposed an Exploratory Data Analysis(EDA) regarding the loan prediction procedure based on the client‘s nature and their requirements.

The major factors concentrated during the data analysis were annual income versus loan purpose, customer ‘s trust, loan tenure versus delinquent months, loan tenure versus credit category, loan tenure versus number of years in the current job, and chances for loan repayment versus the house ownership.

Finally, the outcome of the present work was to infer the constraints on the customer who are applying for the loan followed by the prediction regarding the repayment. Further, results showed that, the customers were interested more on availing short-tenure loans rather than long-tenure loans. “Exploring the Machine Learning Algorithm for Prediction the Loan Sanctioning Process” Author- E. Chandra Bessie, R. Rekha - Year- 2019 Extending credits to corporates and individuals for the smooth functioning of growing economies like India is inevitable. As increasing number of customers apply for loans in the banks and non- banking financial companies (NBFC), it is really challenging for banks and NBFCs with limited capital to device a standard resolution and safe procedure to lend money to its borrowers for their financial needs. In addition, in recent times NBFC inventories have suffered a significant downfall in terms of the stock price. It has contributed to a contagion that has also spread to other financial stocks, adversely affecting the benchmark in recent times. In this paper, an attempt is made to condense the risk involved in selecting the suitable person who could repay the loan on time thereby keeping the bank's nonperforming assets (NPA) on the hold.

This is achieved by feeding the past records of the customer who acquired loans from the bank into a trained machine learning model which could yield an accurate result. The prime focus of the paper is to determine whether or not it will be safe to allocate the loan to a particular person. This paper has the following sections

(i) Collection of Data,

(ii) Data Cleaning and

(iii) Performance Evaluation.

Experimental tests found that the Naïve Bayes model has better performance Evaluation. Experimental tests found that the Naïve Bayes model has better performance than other models in terms of loan forecasting. Manjeet et al (2018) [24] there are seven types of variables that may influence consumer loan default; consumer 'annual income, debt-income ratio, occupation, home ownership, work duration and whether or not consumer possesses saving/checking account.

"Loan Prediction using machine learning model" Year2019whether or not it will be safe to allocate the loan to a particular person. This paper has the following sections

Collection of Data,

(ii) Data Cleaning and

(iii) Performance Evaluation.

Experimental tests found that the Naïve Bayes model has better performance than other models in terms of loan forecasting. With the enhancement in the banking sector lots of people are applying for bank loans but the bank has its limited assets which it has to grant to limited people only, so finding out to whom the loan can be granted which will be a safer option for the bank is a typical process.

So in this project we try to reduce this risk factor behind selecting the safe person so as to save lots of bank efforts and assets. This is done by mining the Big Data of the previous records of the people to whom the loan was granted before and on the basis of these records/experiences the machine was trained using the machine learning model which give the most accurate result The main objective of this project is to predict whether assigning the loan to particular person will be safe or not.

This paper is divided into four sections

(i)Data Collection

(ii) Comparison of machine learning models on collected data

(iii) Training of system on most promising model

(iv) Testing.

In this paper we are predict the loan data by using some machine learning algorithms they are classification, logic regression, Decision Tree and gradient boosting. " Loan Approval Prediction based on Machine Learning Approach" Author- Kumar Arun, Garg Ishan, Kaur Sanmeet Year- 2018The main objective of this paper is to predict whether assigning the loan to particular person will be safe or not. This paper is divided into four sections

(i)Data Collection

(ii) Comparison of machine learning models on collected data

(iii) Training of system on most promising model

(iv) Testing

In 2017, Goyal and Kaur [4] presented a loan prediction model using several Machine Learning (ML) algorithms. The dataset with features, namely, gender, marital status, education, number of dependents, employment status, income, co applicant‘s income, loan amount, loan tenure, credit history, existing loan status, and property area, are used for determining the loan eligibility regarding the loan sanctioning process. Various ML models adopted in the present method includes, Linear model, Decision Tree (DT), Neural Network (NN), Random Forest (RF), SVM, Extreme learning machines, Model tree, Multivariate Adaptive Regression Splines, Bagged Cart Model, NB and TGA.

When evaluated these models using Environment in five runs, TGA resulted in better loan forecasting performance than the other methods. Bekhet and Eletter in 2012 attempted to develop a model with Artificial Neural Networks (ANN) as a decision support system for Jordanian commercial banks to assist in credit approval evaluation.

The system can be easily utilized by credit officers in taking excellent decisions before determining future loan applications and applicants. The results obtained with this technique showed that ANN was better in performance. The main challenge with the approach was in the small dataset used for the evaluation (Bekhet and Eletter, 2012)

2.5.Social or Business Impact

Prediction personal loan approval social or business impact As an AI language model, don't have access to specific information about individuals or their loan applications. However, in general, whether a personal loan application is approved or not can have both social and business impacts.

From a social perspective, being approved for a personal loan can provide financial relief and help individuals meet their immediate needs, such as paying for medical expenses, funding a home renovation project, or consolidating high-interest debt. This can lead to a better quality of life and reduce financial stress.

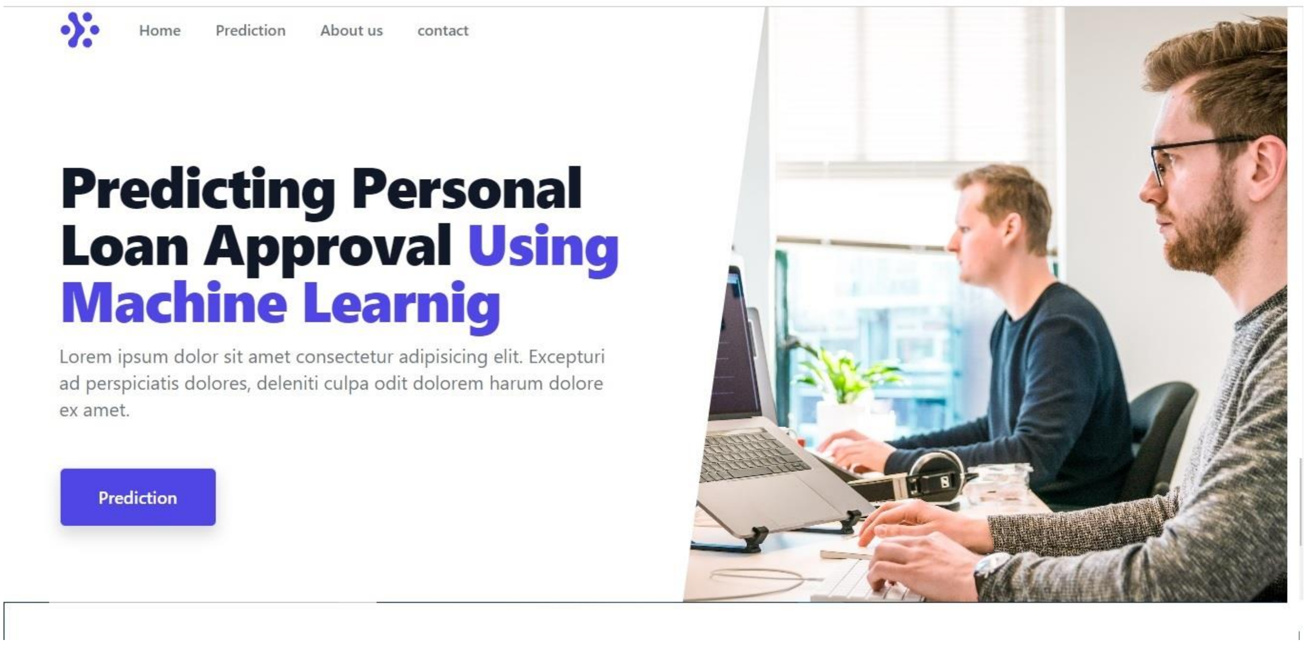
On the other hand, being denied a personal loan can have negative social impacts, such as increased financial strain and potentially damaging credit scores. This can make it harder to secure future loans, rent an apartment, or even find a job. From a business perspective, loan approval can also have significant impacts.

For lenders, loan approval decisions are based on factors such as credit score, income, and employment history. These factors can be used to assess the risk of loan repayment and determine whether a loan application should be approved or denied. Approving loans that are unlikely to be repaid can lead to financial losses for the lender and potentially impact their ability to offer loans to other borrowers in the future.

Therefore, lenders often have strict criteria for loan approval and take into account a wide range of factors before making a decision. In conclusion, personal loan approval or denial can have both social and business impacts, depending on the individual's circumstances and the lender's criteria.

It's important to carefully consider your financial situation and your ability to repay a loan before applying and to choose a reputable lender with fair loan terms.

3.RESULT

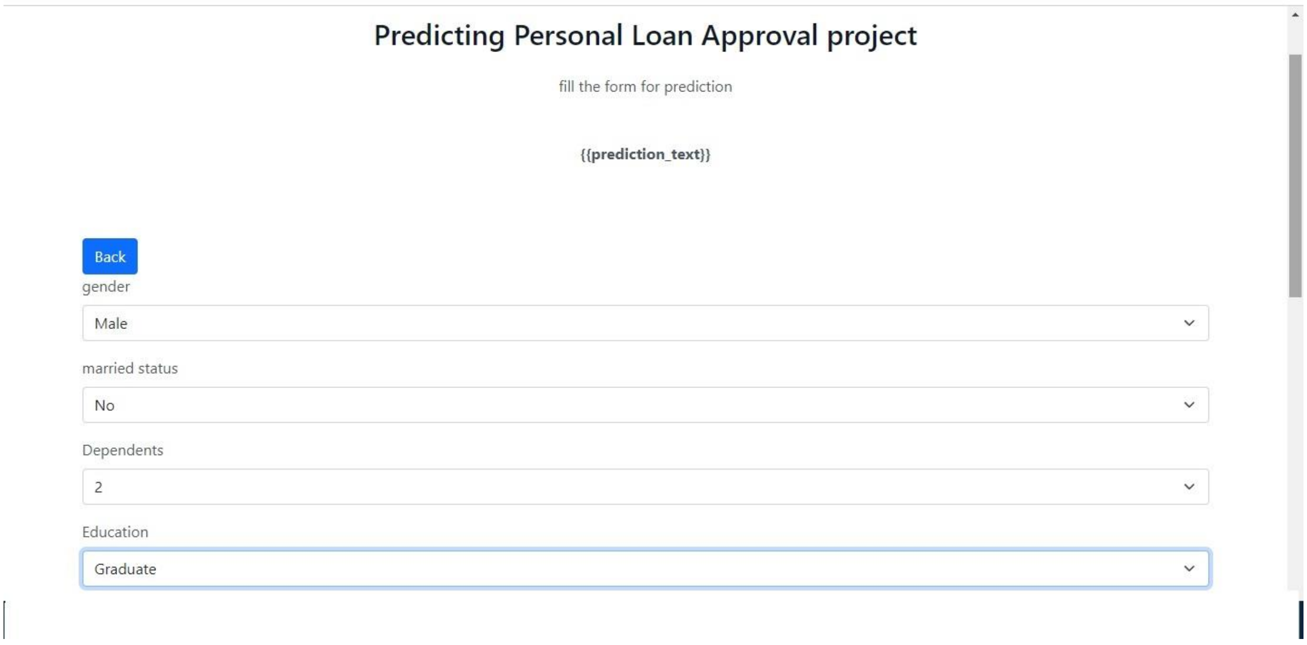


Predicting Personal Loan Approval Using Machine Learnig

Lorem ipsum dolor sit amet consectetur adipisicing elit. Excepturi ad perspiciatis dolores, deleniti culpa odit dolorem harum dolore ex amet.

Web page of Loan Approval!!!

Loan Approval prediction form


Predicting Personal Loan Approval project

fill the form for prediction

{{prediction_text}}

Back

gender

Male

married status

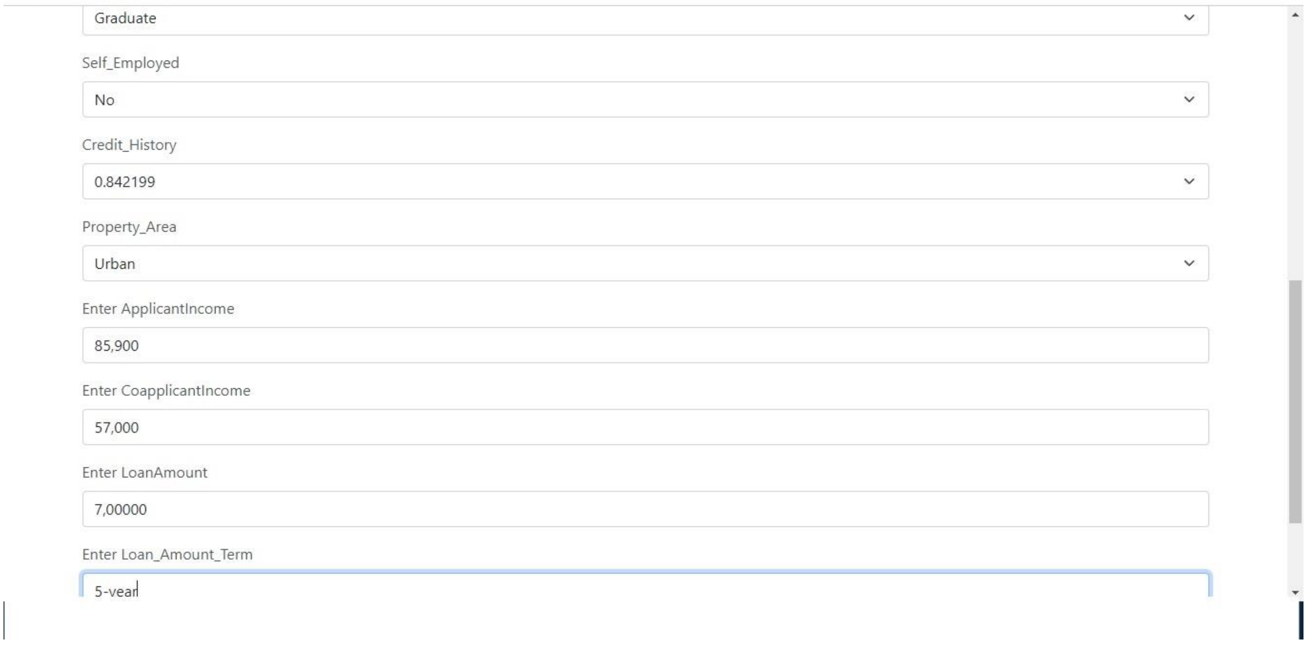
No

Dependents

2

Education

Graduate


Graduate

Self_Employed

No

Credit_History

0.842199

Property_Area

Urban

Enter ApplicantIncome

85,900

Enter CoapplicantIncome

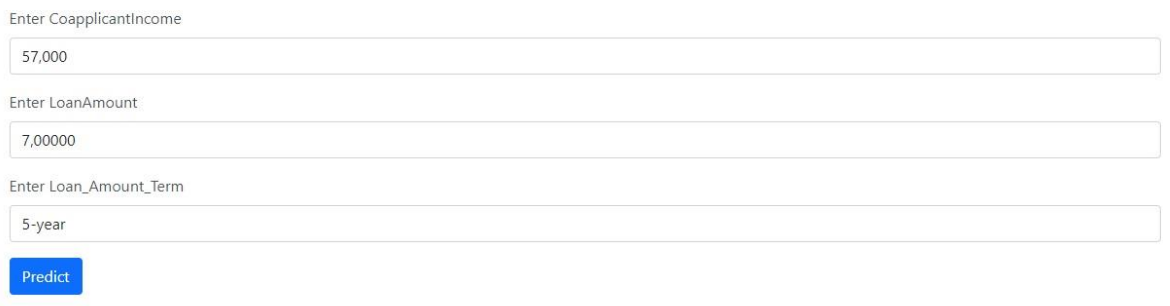
57,000

Enter LoanAmount

7,00000

Enter Loan_Amount_Term

5-year



Enter CoapplicantIncome

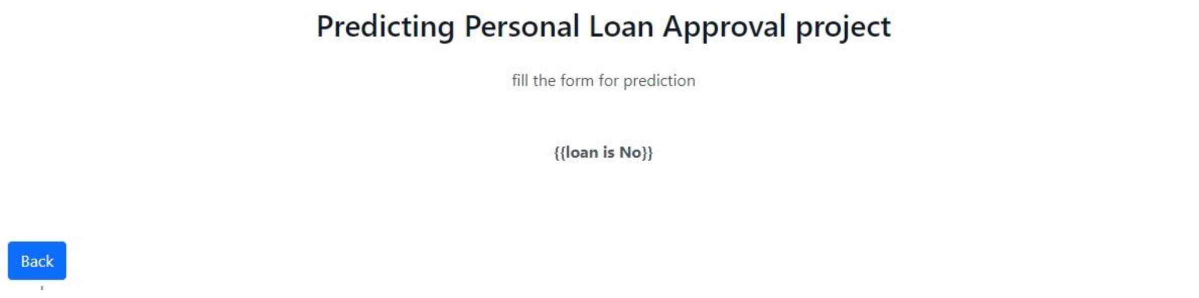
57,000

Enter LoanAmount

7,00000

Enter Loan_Amount_Term

5-year



Predicting Personal Loan Approval project

fill the form for prediction

{{loan is No}}

FINAL OUTPUT PREDICTED!!!

4.ADVANTAGES

- ✓ The nonfunctional requirements ensure the software system follow legal and compliance rules.
- ✓ They ensure the reliability, availability, and performance of the software system
- ✓ They ensure good user experience and ease of operating the software.
- ✓ They help in formulating security policy of the software system.

4.1 DISADVANTAGES

- ✓ None functional requirement may affect the various high-level software subsystem
- ✓ Technically you must be have a hardware requirements and that's ram almost 8 GB RAM.

5.APPLICATIONS

Here we will be using a declared constructor to route to the HTML page which we have createdearlier.

In, '/' URL is bound with the home.html function. Hence, when the home pageof the web server is opened in the browser, the html page will be rendered. Whenever you enterthe values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will berendered to the text that we have mentioned in the submit.html page earlier.

6.CONCLUSION

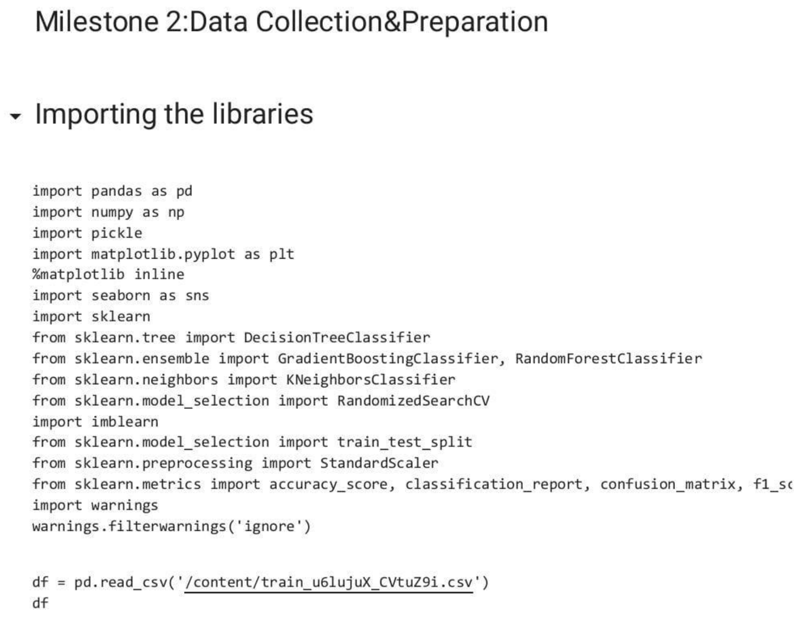
Random Forest Classifier is giving the best accuracy with an accuracy score of 82% for the testing dataset. And to get much better results ensemble Learning techniques like Bagging and Boosting can also be used.

7.FUTURE SCOPE

Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

However, the future scope is benefits can only be reaped if the bank has a robust model to accurately predict which customer's loan it should approve and which to reject, in order to minimize the risk of loan default.

8.APPENDIX



Milestone 2:Data Collection&Preparation

Importing the libraries

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_sc
import warnings
warnings.filterwarnings('ignore')


df = pd.read_csv('/content/train_u6lujuX_CVtuZ9i.csv')
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
0	LP001002	Male	No	0	Graduate	No	584

```
df.drop(['Loan_ID'],axis=1,inplace=True)
```

2	LP001005	Male	Yes	0	Graduate	Yes	300

```
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	Male	No	0	Graduate	No	5849	
1	Male	Yes	1	Graduate	No	4583	
2	Male	Yes	0	Graduate	Yes	3000	
3	Male	Yes	0	Not Graduate	No	2583	
4	Male	No	0	Graduate	No	6000	

```
df['Gender']=df['Gender'].map({'Female':1,'Male':0})
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	0.0	No	0	Graduate	No	5849	
1	0.0	Yes	1	Graduate	No	4583	
2	0.0	Yes	0	Graduate	Yes	3000	
3	0.0	Yes	0	Not Graduate	No	2583	
4	0.0	No	0	Graduate	No	6000	

```
df['Property_Area']=df['Property_Area'].map({'Urban':2,'Semiurban':1, 'Rural':0})
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	0.0	No	0	Graduate	No	5849	
1	0.0	Yes	1	Graduate	No	4583	
2	0.0	Yes	0	Graduate	Yes	3000	

```
df['Married']=df['Married'].map({'Yes':1,'No':0})
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	0.0	0.0	0	Graduate	No	5849	
1	0.0	1.0	1	Graduate	No	4583	
2	0.0	1.0	0	Graduate	Yes	3000	
3	0.0	1.0	0	Not Graduate	No	2583	
4	0.0	0.0	0	Graduate	No	6000	

```
df['Education']=df['Education'].map({'Graduate':1,'Not Graduate':0})
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	0.0	0.0	0	1	No	5849	
1	0.0	1.0	1	1	No	4583	
2	0.0	1.0	0	1	Yes	3000	
3	0.0	1.0	0	0	No	2583	
4	0.0	0.0	0	1	No	6000	

```
df['Self_Employed']=df['Self_Employed'].map({'Yes':1,'No':0})
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	0.0	0.0	0	1	0.0	5849	
1	0.0	1.0	1	1	0.0	4583	
2	0.0	1.0	0	1	1.0	3000	

```
df['Loan_Status']=df['Loan_Status'].map({'Y':1,'N':0})
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	0.0	0.0	0	1	0.0	5849	
1	0.0	1.0	1	1	0.0	4583	
2	0.0	1.0	0	1	1.0	3000	
3	0.0	1.0	0	0	0.0	2583	
4	0.0	0.0	0	1	0.0	6000	

```
df.isnull().sum()
```

```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
```

```
df['Married'] = df['Married'].fillna(df['Married'].mode()[0])
```

```
df['Dependents']=df['Dependents'].str.replace('+','')
```

```
df['Dependents']=df['Dependents'].fillna(df['Dependents'].mode()[0])
```

```
df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
```

```
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mode()[0])
```

```
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0])
```

```
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mode()[0])
```

```
df.isnull().sum()
```

```
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             614 non-null    float64
 1   Married            614 non-null    float64
 2   Dependents         614 non-null    object 
 3   Education          614 non-null    int64  
 4   Self_Employed      614 non-null    float64
 5   ApplicantIncome    614 non-null    int64  
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         614 non-null    float64
 8   Loan_Amount_Term   614 non-null    float64
 9   Credit_History     614 non-null    float64
 10  Property_Area      614 non-null    int64  
 11  Loan_Status        614 non-null    int64  
dtypes: float64(7), int64(4), object(1)
memory usage: 57.7+ KB
```

```
df['Gender']=df['Gender'].astype('int64')
df['Married']=df['Married'].astype('int64')
df['Dependents']=df['Dependents'].astype('int64')
df['Self_Employed']=df['Self_Employed'].astype('int64')
df['CoapplicantIncome']=df['CoapplicantIncome'].astype('int64')
df['LoanAmount']=df['LoanAmount'].astype('int64')
df['Loan_Amount_Term']=df['Loan_Amount_Term'].astype('int64')
df['Credit_History']=df['Credit_History'].astype('int64')
```

Handling Imbalance data

```
from imblearn.combine import SMOTETomek
```

```
smote = SMOTETomek()
```

```
y = df['Loan_Status']
x = df.drop(columns=['Loan_Status'],axis=1)
```

```
x.shape
```

```
    (614, 11)
```

```
y.shape
```

```
    (614,)
```

```
x_bal,y_bal = smote.fit_resample(x,y)
```

```
print(y.value_counts())
print(y_bal.value_counts())
```

```
    1    422
    0    192
    Name: Loan_Status, dtype: int64
    1    345
    0    345
    Name: Loan_Status, dtype: int64
```

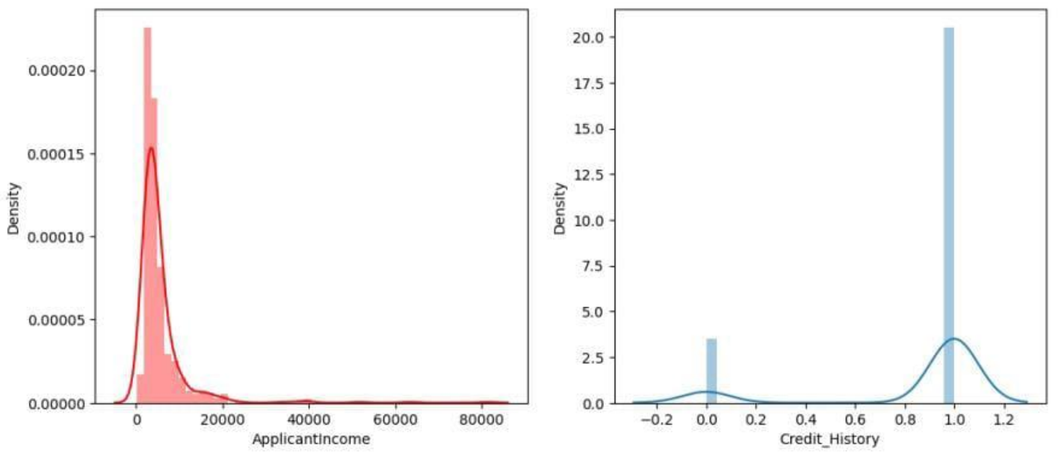
Milestone 3: Exploratory Data Analysis

```
df.describe()
```

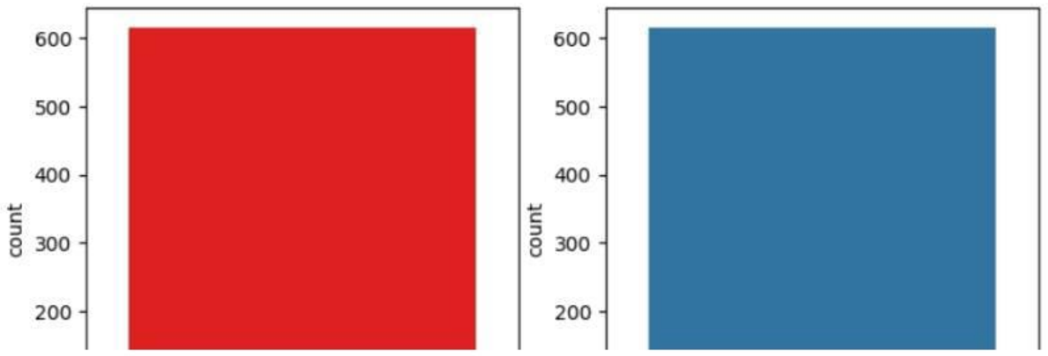
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000
mean	0.182410	0.653094	0.744300	0.781759	0.133550	5403.459283
std	0.386497	0.476373	1.009623	0.413389	0.340446	6109.041673

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000
mean	0.182410	0.653094	0.744300	0.781759	0.133550	5403.459283
std	0.386497	0.476373	1.009623	0.413389	0.340446	6109.041673

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['ApplicantIncome'], color='r')
plt.subplot(122)
sns.distplot(df['Credit_History'])
plt.show()
```

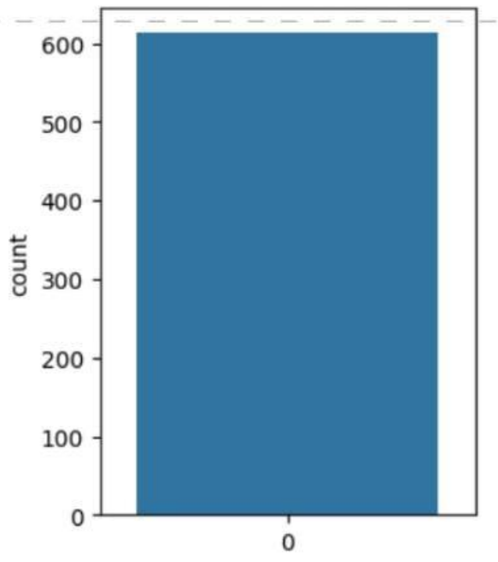


```
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(df ['Gender'],color='r')
plt.xlabel('Gender')
plt.subplot(1,4,2)
sns.countplot(df['Education'])
plt.xlabel('Education')
plt.show()
```



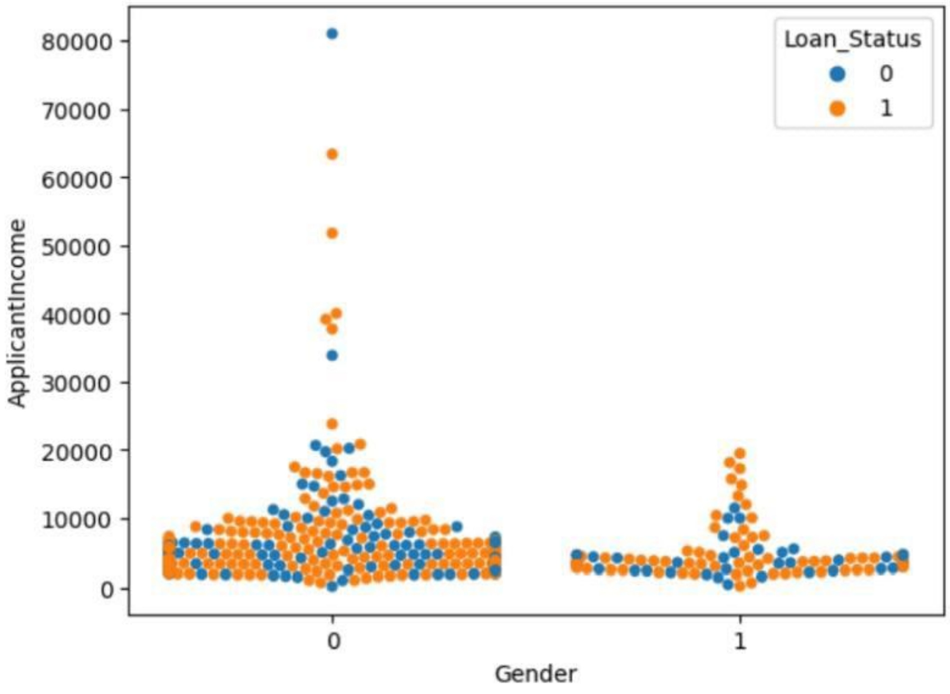
```
plt.figure(figsize=(10,4))
plt.subplot(131)
sns.countplot(df['Married'])
```

```
<Axes: ylabel='count'>
```



```
plt.figure(figsize=(20,5))
plt.subplot(1,3,1)
sns.countplot(x = 'Married', hue = "Gender", data = df)
plt.subplot(1,3,2)
sns.countplot(x = 'Self_Employed', hue = "Education", data = df)
plt.subplot(1,3,3)
sns.countplot(x = 'Property_Area', hue = "Loan_Amount_Term", data = df)
sns.swarmplot(x = 'Gender', y = 'ApplicantIncome', hue = "Loan_Status", data = df)
```

```
<Axes: xlabel='Gender', ylabel='ApplicantIncome'>
```



```
names = x_bal.columns
```

```
x_bal.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	0	0	0	1	0	5849	
1	0	1	0	1	1	3000	
2	0	1	0	0	0	2583	

```
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
```

```
x_bal = pd.DataFrame(x_bal,columns=names)
x_bal.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coappl
0	-0.421545	-1.139877	-0.694022	0.641306	-0.312094	0.134727	
1	-0.421545	0.877288	-0.694022	0.641306	3.204164	-0.377325	
2	-0.421545	0.877288	-0.694022	-1.559318	-0.312094	-0.452272	
3	-0.421545	-1.139877	-0.694022	0.641306	-0.312094	0.161866	
4	-0.421545	0.877288	1.453159	0.641306	3.204164	0.057083	

```
x_train,x_test,y_train,y_test = train_test_split(x_bal,y_bal, test_size=0.33, random_state
```

```
x_train.shape
```

```
(452, 11)
```

```
x_test.shape
```

```
(224, 11)
```

```
y_train.shape, y_test.shape
```

```
((452,), (224,))
```

Milestone 4: Model Building

```
#decision tree model
def RandomForest(x_train,x_test,y_train,y_test):
    model = RandomForestClassifier()
    model.fit(x_train,y_train)
```

```
    y_tr = model.predict(x_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(x_test)
    print(accuracy_score(yPred,y_test))
    y_tr = model.predict(x_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(x_test)
    print(accuracy_score(yPred,y_test))
```

```
RandomForest(x_train,x_test,y_train,y_test)
```

```
1.0
0.8080357142857143
```

```
def decisionTree(x_train,x_test,y_train,y_test):
    model = DecisionTreeClassifier()
    model.fit(x_train,y_train)
    y_tr = model.predict(x_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(x_test)
    print(accuracy_score(yPred,y_test))
```

```
decisionTree(x_train,x_test,y_train,y_test)
```

```
1.0
0.7991071428571429
```

```
def KNN(x_train,x_test,y_train,y_test):
    model = KNeighborsClassifier()
    model.fit(x_train,y_train)
    y_tr = model.predict(x_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(x_test)
    print(accuracy_score(yPred,y_test))
```

```
KNN(x_train,x_test,y_train,y_test)
```

```
0.8584070796460177
0.7410714285714286
```

```
def XGB(x_train,x_test,y_train,y_test):
    model = GradientBoostingClassifier()
    model.fit(x_train,y_train)
    y_tr = model.predict(x_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(x_test)
    print(accuracy_score(yPred,y_test))
```

```
XGB(x_train,x_test,y_train,y_test)
```

```
0.9358407079646017
0.7946428571428571
```

```
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

classifier = Sequential()
classifier.add(Dense(units=100, activation='relu', input_dim=11))

classifier.add(Dense(units=50, activation='relu'))

classifier.add(Dense(units=1, activation='sigmoid'))

classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

classifier.fit(x_train,y_train,batch_size=100, validation_split=0.2, epochs=100)
```

```
Epoch 93/100
4/4 [==============================] - 0s 21ms/step - loss: 0.2191 - accuracy: 0.9
Epoch 94/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2162 - accuracy: 0.9
Epoch 95/100
4/4 [==============================] - 0s 19ms/step - loss: 0.2147 - accuracy: 0.9
Epoch 96/100
4/4 [==============================] - 0s 15ms/step - loss: 0.2127 - accuracy: 0.9
Epoch 97/100
4/4 [==============================] - 0s 22ms/step - loss: 0.2109 - accuracy: 0.9
Epoch 98/100
4/4 [==============================] - 0s 22ms/step - loss: 0.2094 - accuracy: 0.9
Epoch 99/100
4/4 [==============================] - 0s 22ms/step - loss: 0.2089 - accuracy: 0.9
Epoch 100/100
4/4 [==============================] - 0s 20ms/step - loss: 0.2065 - accuracy: 0.9
[illegible]
```

```
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
print(classification_report(y_test,dt.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       118
           1       0.75      0.87      0.80       106

    accuracy                           0.80       224
   macro avg       0.80      0.80      0.80       224
weighted avg       0.81      0.80      0.80       224
```

```
confusion_matrix(y_test,dt.predict(x_test))
```

```
array([[87, 31],
       [14, 92]])
```

```
dt.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])
```

```
array([0])
```

```
rfr = RandomForestClassifier()
rfr.fit(x_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
print(classification_report(y_test,dt.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       118
           1       0.75      0.87      0.80       106

    accuracy                           0.80       224
   macro avg       0.80      0.80      0.80       224
weighted avg       0.81      0.80      0.80       224
```

```
rfr.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])
```

```
array([1])
```

```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
print(classification_report(y_test,dt.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       118
           1       0.75      0.87      0.80       106

    accuracy                           0.80       224
   macro avg       0.80      0.80      0.80       224
weighted avg       0.81      0.80      0.80       224
```

```
knn.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])
```

```
array([1])
```

```
xgb = GradientBoostingClassifier()
xgb.fit(x_train,y_train)
```

```
▾ GradientBoostingClassifier
GradientBoostingClassifier()
```

```
print(classification_report(y_test,dt.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       118
           1       0.75      0.87      0.80       106
```

```
    accuracy                           0.80       224
   macro avg       0.80      0.80      0.80       224
weighted avg       0.81      0.80      0.80       224
```

```
xgb.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])

     array([0])

classifier.save("loan.h5")

y_pred = classifier.predict(x_test)

     7/7 [==============================] - 0s 2ms/step

y_pred
```

```
            [2.01783798e-04],
            [4.60799366e-01],
            [8.92553091e-01],
            [4.05183015e-03],
            [9.93230700e-01],
            [1.15584977e-01],
            [9.30823982e-01],
            [1.99571135e-03],
            [9.67529535e-01],
            [5.40495336e-01],
sample_value = [[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]]
if predict_exit(sample_value)>0.5:
    print('Prediction: High Chance of Loan Approval!')
else:
    print('Prediction: Low Chance of Loan Approval!')
```

```
1/1 [==============================] - 0s 61ms/step
Prediction: Low Chance of Loan Approval!
```

```
sample_value = [[1,0, 1, 1, 1, 45, 14,45, 240, 1,1]]
if predict_exit(sample_value)>0.5:
    print('Prediction: High Chance of Loan Approval!')
else:
    print('Prediction: Low Chance of Loan Approval!')
```

```
1/1 [==============================] - 0s 25ms/step
Prediction: Low Chance of Loan Approval!
```

Milestone 5: Performance Testing & Hyperparameter Tuning

```
def compareModel(x_train,x_test,y_train,y_test):
    decisionTree(x_train,x_test,y_train,y_test)
```

```
    print('-'*100)
    RandomForest(x_train,x_test,y_train,y_test)
    print('-'*100)
    XGB(x_train,x_test,y_train,y_test)
    print('-'*100)
    KNN(x_train,x_test,y_train,y_test)
    print('-'*100)
```

```
compareModel(x_train,x_test,y_train,y_test)
decisionTree(x_train,x_test,y_train,y_test)
print(classification_report(y_test,dt.predict(x_test)))
```

```
1.0
0.7723214285714286
----------------------------------------------------------------------------------------------------
1.0
0.8035714285714286
----------------------------------------------------------------------------------------------------
0.9358407079646017
0.7946428571428571
----------------------------------------------------------------------------------------------------
0.8584070796460177
0.7410714285714286
----------------------------------------------------------------------------------------------------
1.0
0.7678571428571429
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       118
           1       0.75      0.87      0.80       106

    accuracy                           0.80       224
   macro avg       0.80      0.80      0.80       224
weighted avg       0.81      0.80      0.80       224
```

```
RandomForest(x_train,x_test,y_train,y_test)
print(classification_report(y_test,dt.predict(x_test)))
```

```
1.0
0.7991071428571429
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       118
           1       0.75      0.87      0.80       106

    accuracy                           0.80       224
   macro avg       0.80      0.80      0.80       224
weighted avg       0.81      0.80      0.80       224
```

```
XGB(x_train,x_test,y_train,y_test)
print(classification_report(y_test,dt.predict(x_test)))
```

```
0.9358407079646017
0.7946428571428571
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       118
           1       0.75      0.87      0.80       106

    accuracy                           0.80       224
   macro avg       0.80      0.80      0.80       224
weighted avg       0.81      0.80      0.80       224
```

```
KNN(x_train,x_test,y_train,y_test)
print(classification_report(y_test,dt.predict(x_test)))
```

```
0.8584070796460177
0.7410714285714286
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       118
           1       0.75      0.87      0.80       106

    accuracy                           0.80       224
   macro avg       0.80      0.80      0.80       224
weighted avg       0.81      0.80      0.80       224
```

```
y_pred = y_pred.astype(int)
y_pred
```

```
       [1],
       [1],
       [1],
       [0],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [0],
       [0],
       [1],
       [0],
       [1],
       [0],
       [1],
       [0],
       [1],
       [1],
       [1],
       [1],
       [1],
       [0],
       [1],
       [1],
       [1],
       [0],
       [1]])
```

```
ypred = classifier.predict(x_test)
print(accuracy_score(y_test,y_pred))
print("ANN Model")
print("Confusion_Matrix")
print(confusion_matrix(y_test,y_pred))
print("Classification Report")
print(classification_report(y_test,y_pred))
```

```
7/7 [==============================] - 0s 2ms/step
0.7633928571428571
ANN Model
Confusion_Matrix
[[74 44]
 [ 9 97]]
Classification Report
              precision    recall  f1-score   support

           0       0.89      0.63      0.74       118
           1       0.69      0.92      0.79       106

    accuracy                           0.76       224
   macro avg       0.79      0.77      0.76       224
weighted avg       0.80      0.76      0.76       224
```

```
from sklearn.model_selection import cross_val_score
```

```
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
ypred = rf.predict(x_test)
```

```
f1_score(ypred,y_test,average='weighted')
```

```
0.8050020008003201
```

```
np.mean(cv)

     0.793202718912435


ypred = classifier.predict(x_test)
print(accuracy_score(y_test,y_pred))
print("Random Froest")
print("Confusion_Matrix")
print(confusion_matrix(y_test,y_pred))
print("Classification Report")
print(classification_report(y_test,y_pred))

     7/7 [==============================] - 0s 2ms/step
     0.7633928571428571
     Random Froest
     Confusion_Matrix
     [[74 44]
      [ 9 97]]
     Classification Report
                   precision    recall  f1-score   support

                0       0.89      0.63      0.74       118
                1       0.69      0.92      0.79       106

         accuracy                           0.76       224
        macro avg       0.79      0.77      0.76       224
     weighted avg       0.80      0.76      0.76       224


ypred = classifier.predict(x_test)
print(accuracy_score(y_test,y_pred))
print("XGB")
print("Confusion_Matrix")
print(confusion_matrix(y_test,y_pred))
print("Classification Report")
print(classification_report(y_test,y_pred))

     7/7 [==============================] - 0s 3ms/step
     0.7633928571428571
     XGB
     Confusion_Matrix
     [[74 44]
      [ 9 97]]
```

```
     Classification Report
                   precision    recall  f1-score   support

                0       0.89      0.63      0.74       118
                1       0.69      0.92      0.79       106

         accuracy                           0.76       224
        macro avg       0.79      0.77      0.76       224
     weighted avg       0.80      0.76      0.76       224
```

Hyper Parameter Tuning

```
rf = RandomForestClassifier()


parameters = {
                'n_estimators' :[1,20,30,55,68,74,90,120,115],
                 'criterion':['gini','entropy'],
                  'max_features' : ["auto", "sqrt", "log2"],
              'max_depth' : [2,5,8,10], 'verbose' : [1,2,3,4,6,8,9,10]
}


RCV = RandomizedSearchCV(estimator=rf,param_distributions=parameters,cv=10,n_iter=4)


RCV.fit(x_train,y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
building tree 1 of 74
building tree 2 of 74
building tree 3 of 74
building tree 4 of 74
building tree 5 of 74
building tree 6 of 74
building tree 7 of 74
building tree 8 of 74
building tree 9 of 74
building tree 10 of 74
building tree 11 of 74
building tree 12 of 74
building tree 13 of 74
building tree 14 of 74
```

```
    building tree 57 of 74
```

```
bt_params = RCV.best_params_
bt_score = RCV.best_score_
```

```
    building tree 61 of 74
```

```
bt_params
```

```
    {'verbose': 4,
     'n_estimators': 74,
     'max_features': 'log2',
     'max_depth': 10,
     'criterion': 'gini'}
```

```
    building tree 70 of 74
```

```
bt_score
```

```
    0.8276811594202899
```

```
    building tree 1 of 74
```

```
#training and test the xgboost model on the best parameters gor forme randomized cv
def RandomForest(x_train,x_test,y_train,y_test):
    model = RandomForestClassifier(verbose=4, n_estimators=74,max_features='log2',max_dept
    model.fit(x_train,y_train)
    y_tr = model.predict(x_train)
    print("Training Accuracy")
    print(accuracy_score(y_tr,y_train))
    ypred = model.predict(x_test)
    print('Testing Accuracy')
    print(accuracy_score(ypred,y_test))
```

```
    building tree 14 of 74
```

```
model = RandomForestClassifier(verbose=4, n_estimators=74,max_features='log2',max_depth=1
model.fit(x_train,y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
building tree 1 of 74
building tree 2 of 74
building tree 3 of 74
building tree 4 of 74
building tree 5 of 74
building tree 6 of 74
building tree 7 of 74
building tree 8 of 74
building tree 9 of 74
building tree 10 of 74
building tree 11 of 74
building tree 12 of 74
building tree 13 of 74
building tree 14 of 74
building tree 15 of 74
building tree 16 of 74
building tree 17 of 74
building tree 18 of 74
building tree 19 of 74
building tree 20 of 74
building tree 21 of 74
building tree 22 of 74
building tree 23 of 74
building tree 24 of 74
building tree 25 of 74
building tree 26 of 74
building tree 27 of 74
building tree 28 of 74
building tree 29 of 74
building tree 30 of 74
building tree 31 of 74
building tree 32 of 74
building tree 33 of 74
building tree 34 of 74
building tree 35 of 74
building tree 36 of 74
building tree 37 of 74
building tree 38 of 74
building tree 39 of 74
building tree 40 of 74
building tree 41 of 74
building tree 42 of 74
building tree 43 of 74
building tree 44 of 74
building tree 45 of 74
building tree 46 of 74
building tree 47 of 74
building tree 48 of 74
building tree 49 of 74
building tree 50 of 74
building tree 51 of 74
building tree 52 of 74
```

Saving the Model

```
import pickle
pickle.dump(model,open('rdf.pkl','wb'))
```