Smart Lender - Applicant Creadibility Prediction For Loan Approval

NALAIYA THIRAN PROJECT BASED LEARNING

On

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

A PROJECT REPORT

CHIRUMAMILLA ARJUN	720819106019
DABBUGUNTA YASWANTH	720819106020
DEVANATHAN R	720819106022
DEVARALA PRAVEEN KUMAR	720819106023

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHNOLOGY

Approved by AICTE, New Delhi, Accredited with ‘A’ Grade by NAAC

(An Autonomous Institution, Affiliated to Anna University, Chennai)

COIMBATORE – 641 032

November 2022

INDEX

CHAPTER-1
INTRODUCTION

1.1 Project Overview

Despite the fact that our banking system has many products to sell, the main source of income for a bank is its credit line. So, they can earn from interest on the loans they credit. Commercial loans have always been a big part of the banking industry, and lenders are always aiming to reduce their credit risk. Nowadays in the market economy banks play a very crucial role. The profit or loss of a bank is largely influenced by loans, i.e., whether the customers repay the loans or default on them. The banks need to decide whether he/she is a good(non-defaulter) or bad(defaulter) before giving the loans to the borrowers. Among the most important problems to be addressed in commercial loan lending is the borrowers' creditworthiness. The credit risk is defined as the likelihood that borrowers will fail to meet their loan obligations To predict whether the borrower will be good or bad is a very difficult task for any bank or organization. The banking system uses a manual process for checking whether a borrower is a defaulter or not. No doubt the manual process will be more accurate and effective, but this process cannot work when there are a large number of loan applications at the same time. If there occurs a time like this, then the decision-making process will take a very long time and also lots of manpower will be required. If we are able to do the loan prediction it will be very helpful for applicants and also for the employees of banks. So, the task is to classify the borrower as good or bad i.e., whether the borrower will be able to pay the debts back or not. This can be done with the help of machine learning algorithms.

1.2 Purpose

A lender is a financial institution that repaid at a lends money to a corporate or an individual borrower with the expectation that the money will be later date. Lenders require borrowers to pay interest on the amount borrowed, usually charged at a specific percentage of the total amount of loan.

CHAPTER-2

LITERATURE SURVEY

In they have used only one algorithm; there is no comparison of different algorithms. The algorithm used was Logistic Regression and the best accuracy

they got was 81.11%. The final conclusion reached was only those who have a good credit score, high income and low loan amount requirement will get their loan approved. Comparison of two machine learning algorithms was made in . The two algorithms used were two class decision jungle and two class decision and their accuracy were 77.00% and 81.00% respectively. Along with these they also calculated parameters such as Precision, recall, F1 score and AUC. The [3] shows a comparison of four algorithms. The algorithms used were Gradient Boosting,
Logistic Regression, Random Forest and CatBoost
Classifier. Logistic Regression gave a very low accuracy of 14.96%. Random forest gave a good accuracy of 83.51%. The best accuracy we got was from CatBoost Classifier of 84.04%. There was not much difference between Gradient Boosting and CatBoost Classifier in terms of accuracy. Accuracy of Gradient Boosting was 84.03%. Logistic Regression, Support Vector Machine, Random Forest and Extreme Gradient Boosting algorithms are used in [4]. The accuracy percentage didn't vary a lot between all the algorithms. But the support vector Machine gave the lowest variance.

The less the variance, the less is the fluctuation of scores and the model will be more precise and stable. Only the K Nearest Neighbor Classifier is used in [5]. The process of Min-Max Normalization is used. It is a process of decomposing the attributes values. The highest accuracy they got was 75.08% when the percentage of dataset split was 50-50% with k to be set as 30. In [6] Logistic Regression is the only algorithm used. They didn't calculate the accuracy of the algorithm.

2.1 Existing Problem

Genetic algorithms (Holland, 1975, 1992) provide a method to perform randomized global search in a solution space. They operate on a population of potential solutions applying the principle of survival of the fittest to produce

(hopefully) better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than

the individuals that they were created from.
Usually, the algorithm starts with a random population of N candidate solutions, which are internally encoded as chromosomes (in the form of a string). Next the quality of each chromosome x in the population is evaluated by a fitness function $f(x)$, and the best two are selected to crossover and form a new solution (offspring). A further genetic operator, called mutation, may be then applied to the new offspring, which causes the individual genetic representation to be changed according to some probabilistic rule. After recombination and mutation, the process continues through subsequent generations and it terminates either after a predefined number of iterations or if the best member of the latest populations has not improved during a certain number of iterations.

2.2 References

[1] M. A. Sheikh, A. K. Goel and T. Kumar, "An Approach for Prediction of Loan Approval using Machine Learning Algorithm," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 490-494,doi: 10.1109/ICESC48915.2020.9155614.

[2] K. Alshouiliy, A. AlGhamdi and D. P. Agrawal, "AzureML Based Analysis and Prediction Loan Borrowers Creditworthy," 2020 3rd International Conference on Information and Computer Technologies (ICICT), 2020, pp. 302-306, doi: 10.1109/ICICT50521.2020.00053.

[3] B. Patel, H. Patil, J. Hembram and S. Jaswal, "Loan Default Forecasting using Data Mining," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-4, doi: 10.1109/INCET49848.2020.9154100.

[4] S. Z. H. Shoumo, M. I. M. Dhruba, S. Hossain, N. H. Ghani, H. Arif and S. Islam, "Application of Machine Learning in Credit Risk Assessment: A Prelude to Smart Banking," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 2023-2028, doi: 10.1109/TENCON.2019.8929527.
[5] G. Arutjothi, C. Senthamarai," Prediction of loan status in commercial bank using machine learning classifier" 2018 International Conference Sustainable Systems (ICISS)
[6] Ashlesha Vaidya, "Predictive and Probabilistic approach using Logistic Regression" 2017 8th International Conference on Computing, Communication and Networking Technologies.

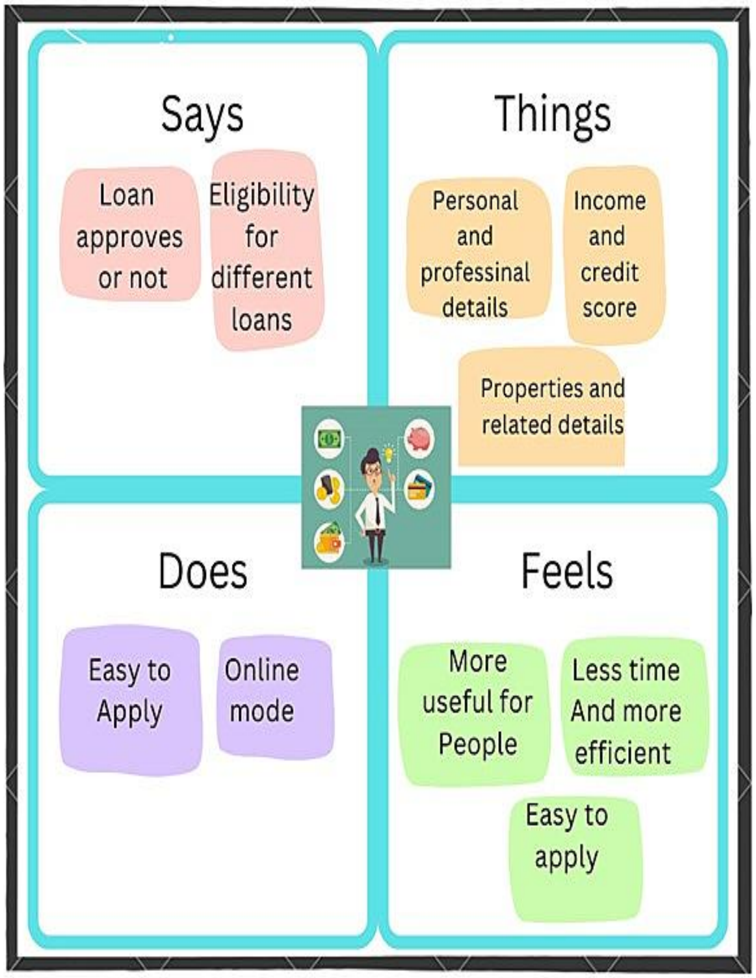
2.3 Problem Statement Defination

- Company wants to automate the loan eligibility process(real time) based oncustomer detail providedwhile filling onlineapplication form.
- These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others.
- Toautomate this process, they have given a problem to identify the customers segments, thoseare eligible for loan amount so that they can specifically target these customers
- It is a classification problem where we have to predictwhether a loan would be approved or not.

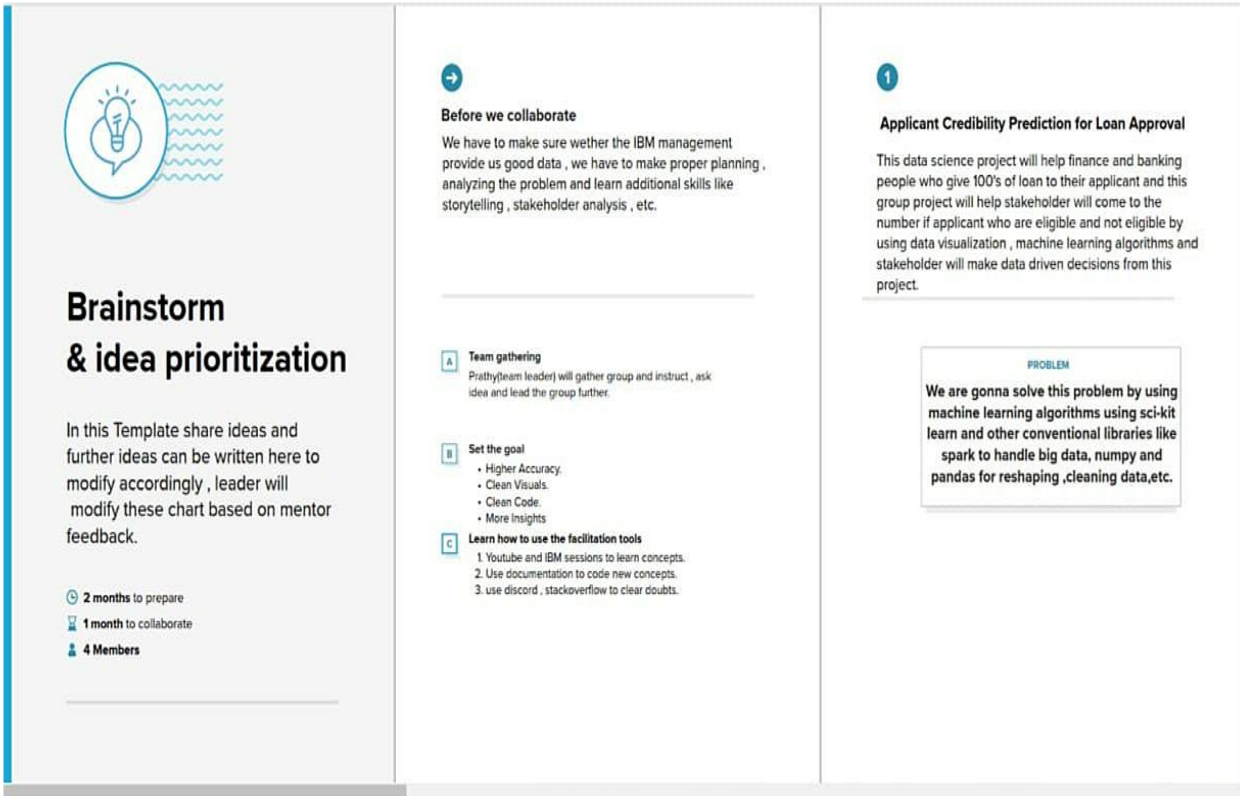
CHAPTER-3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstroming



3.3 Proposed Solution

These solution template relates the current situation to a desired result of this project andalsodescribe the benefits acquire when desired result is achieved.

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	1. Tracking or checking the status is difficult. 2. Prone to human errors. 3. Time consumption is high. 4. Lot of paper works.

		5. Poor customer service due to lack of manpower.
2.	Idea / Solution description	1. Tracking or checking the status becomes easy. •Reducethe potential for human error. 2. Time consumption of the process will be reduced. 3. Reduces the paperwork to paperless. 4. Improve the effectiveness of customer serviceteams. 5. Fair eligibility prediction. 6. Highly scalable and provide data driven decisionsto stakeholder and higherauthority. We will be using classification algorithms such as Decision tree, Random Forest, KNN, and xgboost to achieve higher accuracy in predicting the model. We will train and test the data with these algorithms, tune by hyperparameter tunning. From thisthe above ideas are implemented.
3.	Novelty / Uniqueness	As soon as the essential data are provided, the model will predict whether to approve the loan ornot - By use of transfer learning.

4.	Social Impact / Customer Satisfaction	One of the most important factors which affectour country's economy and financial condition is the credit systemgoverned by the banks. As we know credit risk evaluation is very crucial, thereis a variety of techniques are used for risk level calculation. In addition, credit riskis one of the mainfunctions of the banking community.
5.	Business Model (Revenue Model)	This model can be developed by minimum cost at the same time it will provide the peak performance, higher accuracy and the result willbe more effective than traditional techniques.

3.4 Problem Solution Fit

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

I. Bank higher authority.

II. Bank decision makers.

III. Stakeholders and customers.

IV. Persons who are giving and applying for loans.

6. CUSTOMER CONSTRAINTS CC

I. Loan approval prediction model predicts well by ml Algorithms . Training maybe slightly tricky.
II. Security issue maybe a concern and in rare case It may be hard to recover the bank details.

5. AVAILABLE SOLUTIONS AS

I. It reduces the workforce of the bank Employees.
II. Easy to predict and highly scalable.
III. It gives more insight and leads to more profit by data driven decision.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

I. Enter the details given by customers.

II. By ML algorithms predict the loan Approval.

III. By getting results employees and companies can provide loans.

9. PROBLEM ROOT CAUSE RC

I. Faster loan approval .

II. Profit for stakeholders.

III. Maintain standards in company.

IV. Scalability.

7. BEHAVIOUR BE

I. Collecting user data and attributes of personal details of user.

II. Perform EDA and provide Insight for stakeholder

III. At end Model will predict for loan eligibility.

Focus on J&P, tap into BE, understand RC

3. TRIGGERS TR

A. Scope of ML and data science increasesd ay byday.
B. Financial and Banks arein need of fasterloan approval model.

10. YOUR SOLUTION SL

1. Providing cleaner visuals to stakeholders.
2. Helping higherlevel and employees to takedatadriven decision.
3. More accuracy ML model forpredicting customerdata.

1. **CHANNELS of BEHAVIOUR** CH
    a. ONLINE

Online loan approval system - By online services of company customers can know their loan eligibility.

    b. OFFLINE

4.EMOTIONS: BEFORE / AFTER EM Before : Lots of workload and pressure to check and provide loaneligibility , It needs lots of humanor labor force. After : Easy , scalable and rapid approval in predicting andproviding loans to customers.	4. Highly scalable - Transfer learning allows highscalability and can be used across different leveland locations of particular bank orfinance company.	Bank and finance - Employees can work easily in offline and provide customer satisfaction in least effort

CHAPTER-4
REQUIREMENTS ANALYSIS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	User Registration	Registration through Bank WebsiteRegistration through Gmail

		Registration throughmobile Application
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Loan type	Personal LoanEducation Loan
FR-4	User Details	Name, Address, Income, Occupation.
FR-5	Assets Proof	Agricultural land, Gold
FR-6	Verification	Verification of user Details which are provided above

Non-functional Requirements:

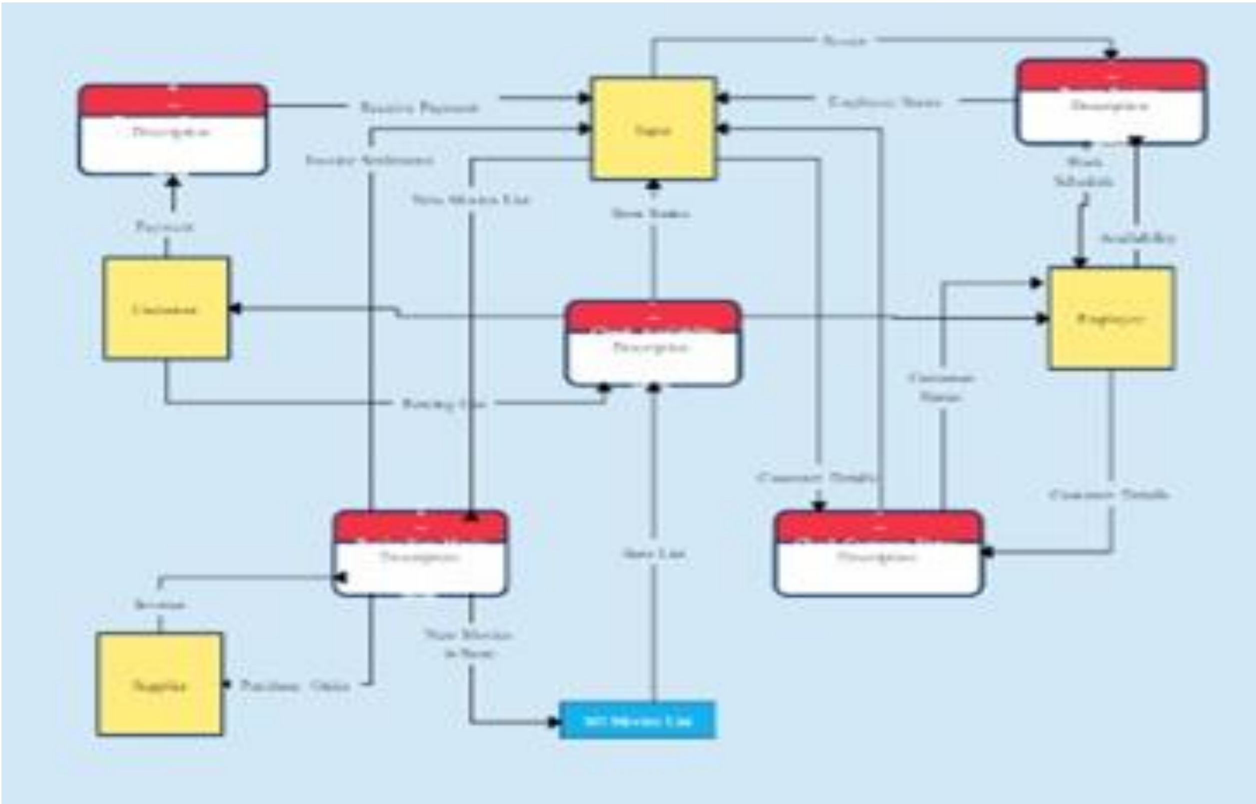
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Easy to access
NFR-2	**Security**	User proofs
NFR-3	**Reliability**	Based on the customer Income

NFR-4	**Performance**	Previous history of the userbank account
NFR-5	**Availability**	Based on the customer Address
NFR-6	**Scalability**	Based on the customer Assets proofs

CHAPTER-5
PROJECT DESIGN

5.1 Data Flow Diagrams



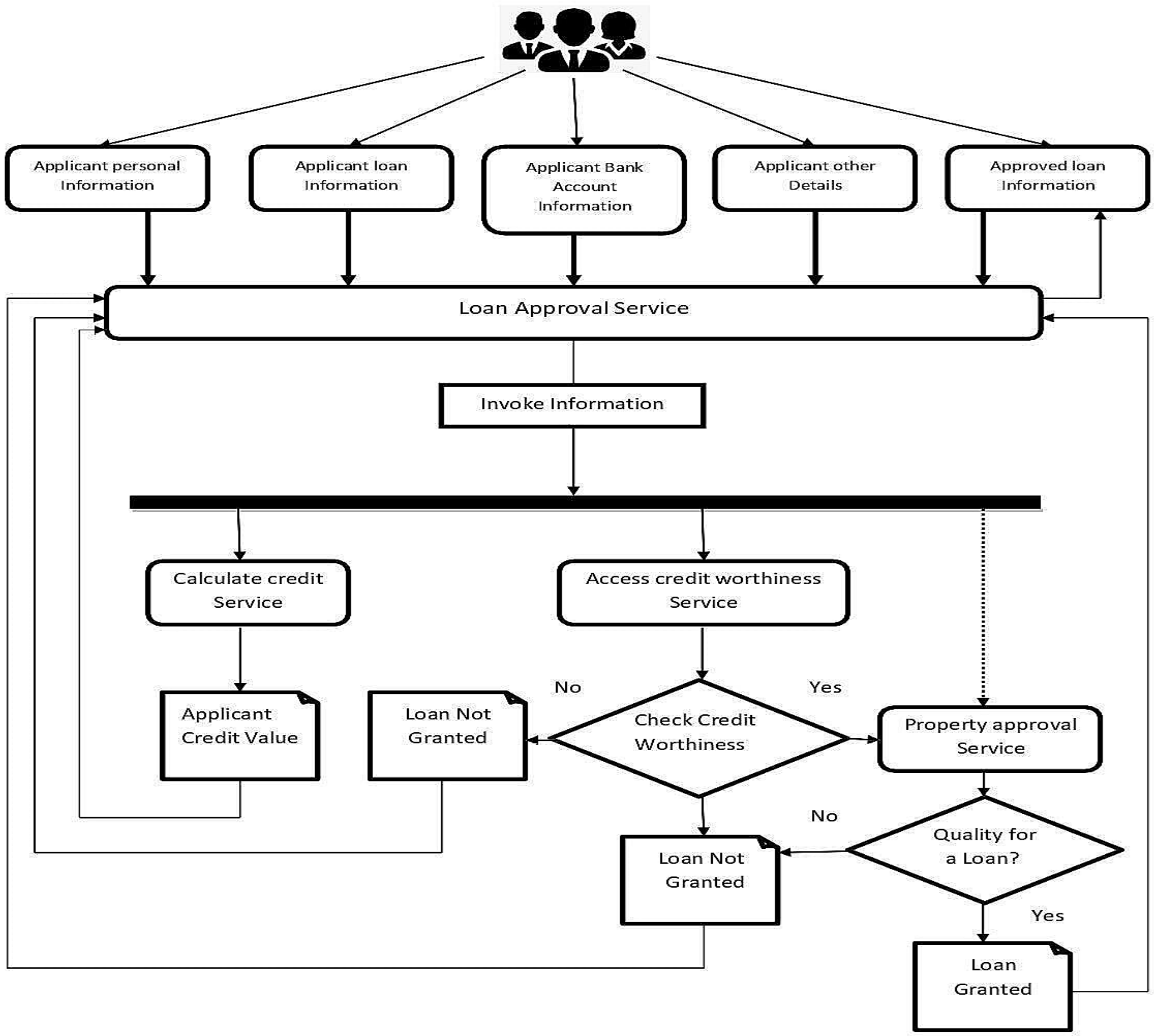
5.2 Solution & Technical Architecture

Solution Architecture

The primary goal in the banking industry is to place their funds in safe hands. So,the system needs to verify the documents effectively and should ensure thatonly capable people get the loan.

1. The model should be trained to produce results with satisfactory accuracy, afterwhich it produces accurate results as to whether a borrower should be lentmoney or not without any tedious manual work.
2. The userscan get the results in the comfort of their home.
3. The systemshould reduce risk to both the bank and the customer

Solution Architecture diagram:



5.3 User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release

Customer (Mobile user)	Registration	USN-1	As a user, I can register for the loan application by entering my email/user number, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the loan application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the loan application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can receive the mail that you are registered in loan application.	Medium	Sprint-1

CHAPTER-6
PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user,I can register for the application by	3	High	Arjun Yaswanth Devanathan Praveen

			entering my email, password, and confirming my password.			
Sprint-1		USN-2	As a user, I will receive confirmation email once I haveregistered for the application	3	High	Arjun Yaswanth Devanathan Praveen

Sprint-1		USN-3	As a user, I can register for the application through Facebook	1	Low	Arjun Yaswanth Devanathan Praveen
Sprint-1		USN-4	As a user, I can register for the	2	Medium	Arjun Yaswanth Devanathan

			application through Gmail			Praveen
Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-1	Login	USN-5	As a user, I canlog into the application by entering email & password	3	High	Arjun Yaswanth Devanathan Praveen
Sprint-1	Dashboard	USN-6	As a user, I should be able to access the dashboard with everything I am allowed touse.	2	Medium	Arjun Yaswanth Devanathan Praveen

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on PlannedEnd Date)	Sprint ReleaseDate(Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	28	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	10	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	25	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	6	19 Nov 2022

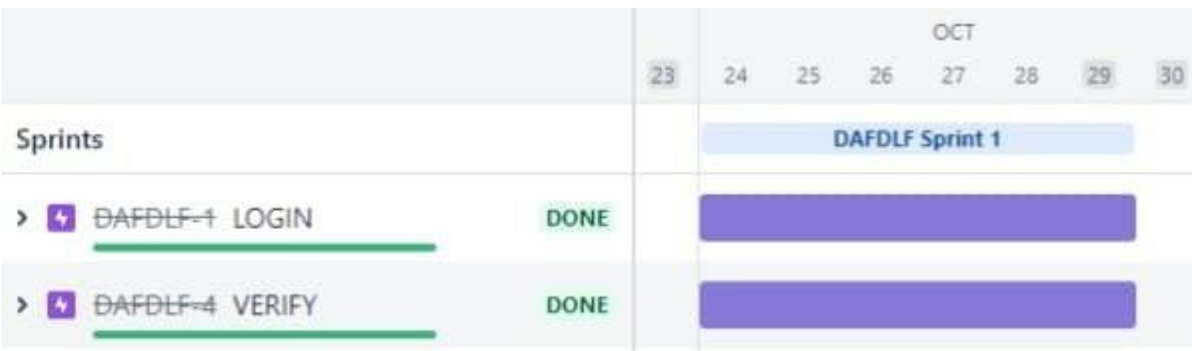
Velocity

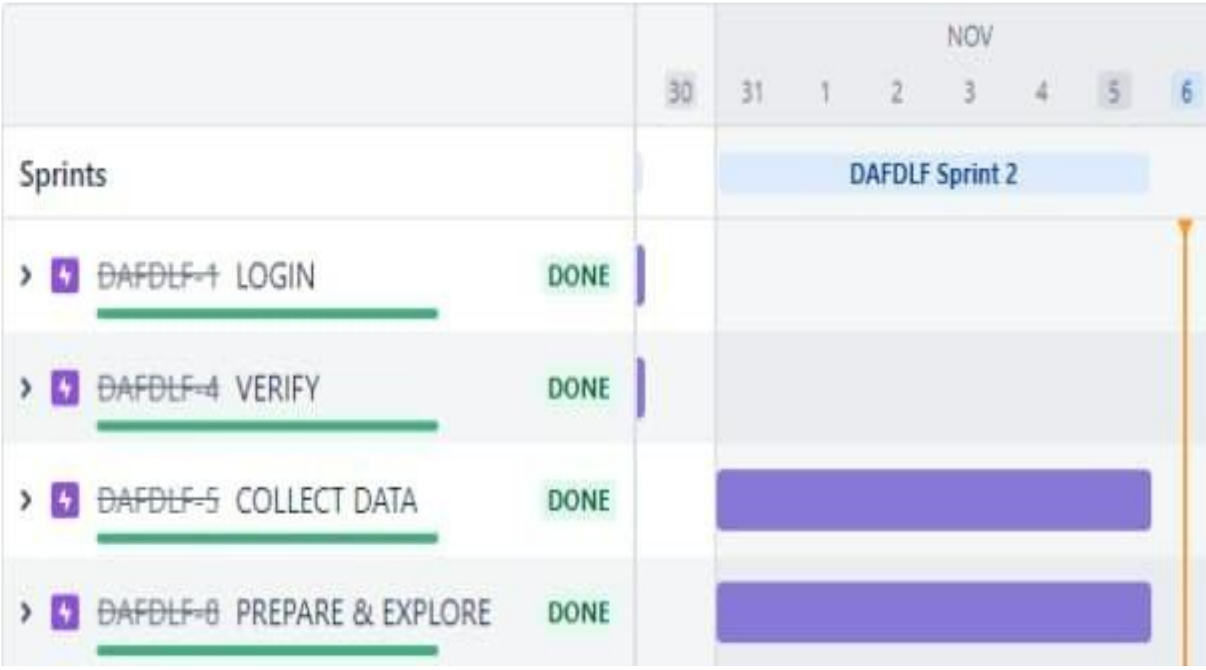
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) periteration unit (storypoints per day)

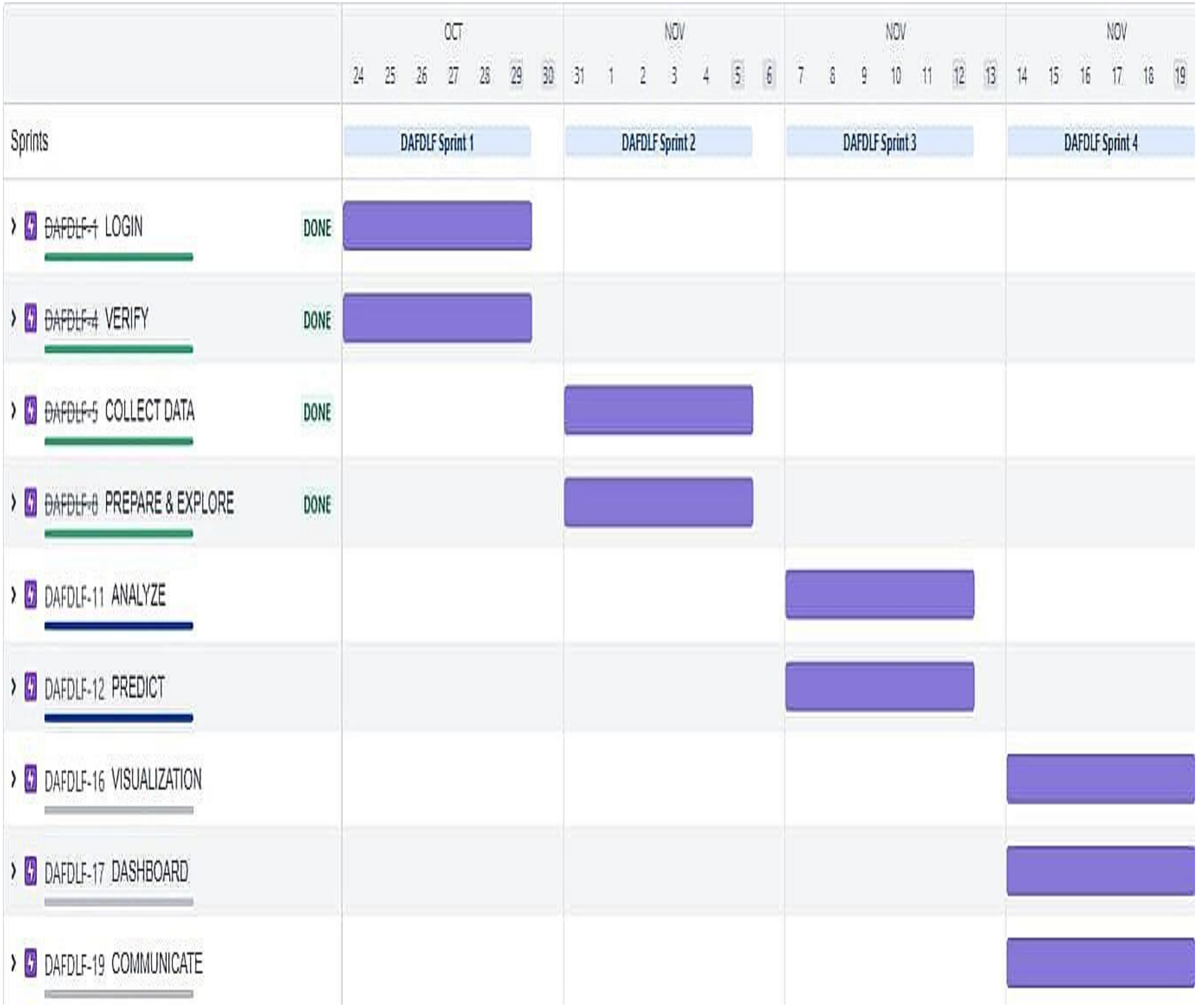
Burndown Chart

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum.However, burn down charts can be applied to any projectcontaining measurable progress over time.

6.3 Reports From JIRA




NOV
30
31
1
2
3
4
5
6
Sprints
DAFDLF Sprint 2
~~DAFDLF-1~~ LOGIN
DONE
~~DAFDLF-4~~ VERIFY
DONE
~~DAFDLF-5~~ COLLECT DATA
DONE
~~DAFDLF-8~~ PREPARE & EXPLORE
DONE



CHAPTER-7
CODING AND SOLUTIONING

7.1 Feature 1:

In this Chapter we will see about the coding part used for the project.

Importing the Libraries

In [1]:
```
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 ,MaxAbsScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

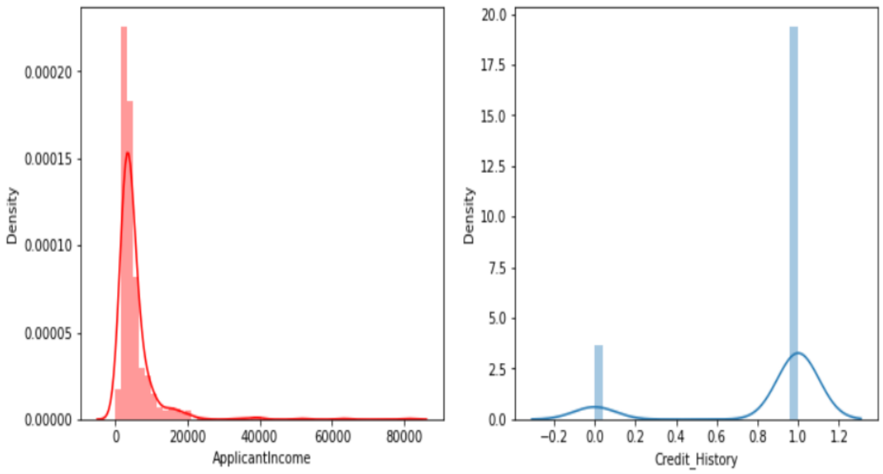
Reading the dataSet

In [2]:
```
data=pd.read_csv(r"..\data_sets\loan_data.csv")
data
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Ar
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urt
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Ru
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urt
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urt
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urt
...	...	...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Ru
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Ru
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urt
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urt
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurt

Visualizations

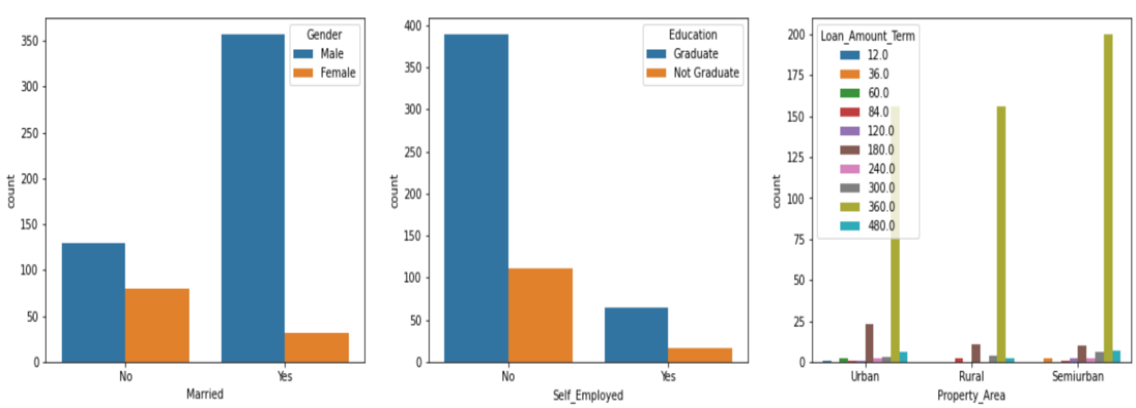
In [3]:
```
#plotting the using distplot
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'], color='r')
plt.subplot(122)
sns.distplot(data['Credit_History'])
plt.show()
```



In [4]:
```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(data['Married'], hue=data['Gender'])
plt.subplot(132)
sns.countplot(data['Self_Employed'], hue=data['Education'])
plt.subplot(133)
sns.countplot(data['Property_Area'], hue=data['Loan_Amount_Term'])
```

```
keyword will result in an error or misinterpretation.
  warnings.warn(
```

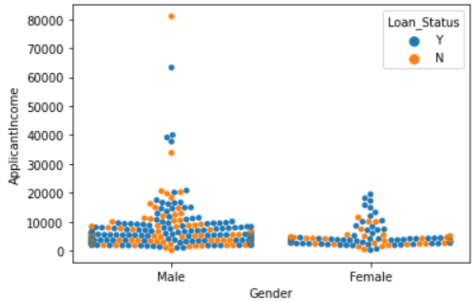
Out[4]: <AxesSubplot:xlabel='Property_Area', ylabel='count'>



In [5]:
```
sns.swarmplot(data['Gender'], data['ApplicantIncome'], hue = data['Loan_Status'])
```

```
u may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[5]: `<AxesSubplot:xlabel='Gender', ylabel='ApplicantIncome'>`



Data Pre-processing

In [6]:
```
data.describe()
```

Out[6]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Handling the Null Values

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
Loan_ID               0
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
```

In [9]:
```
data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])
data['Married'] = data['Married'].fillna(data['Married'].mode()[0])
#replacing + with space for filling the nan values
data['Dependents']=data['Dependents'].replace('3+',3)
data['Dependents'] = data['Dependents'].fillna(data['Dependents'].mode()[0])
data['Self_Employed'] = data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])
data['LoanAmount'] = data['LoanAmount'].fillna(data['LoanAmount']. mode()[0])
data['Loan_Amount_Term'] = data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0])
data['Credit_History'] = data['Credit_History'].fillna(data['Credit_History'].mode()[0])
```

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
Loan_ID              0
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
```

Handling the categorical columns

In [11]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data.Gender=le.fit_transform(data.Gender)
data.Loan_Status=le.fit_transform(data.Loan_Status)
data.Married=le.fit_transform(data.Married)
data.Education=le.fit_transform(data.Education)
data.Self_Employed=le.fit_transform(data.Self_Employed)
data.Property_Area=le.fit_transform(data.Property_Area)
```

In [12]:
```
data
```

Out[12]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	1	0	0	0	0	5849	0.0	120.0	360.0	1.0
1	LP001003	1	1	1	0	0	4583	1508.0	128.0	360.0	1.0
2	LP001005	1	1	0	0	1	3000	0.0	66.0	360.0	1.0
3	LP001006	1	1	0	1	0	2583	2358.0	120.0	360.0	1.0
4	LP001008	1	0	0	0	0	6000	0.0	141.0	360.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	0	0	0	0	0	2900	0.0	71.0	360.0	1.0
610	LP002979	1	1	3	0	0	4106	0.0	40.0	180.0	1.0
611	LP002983	1	1	1	0	0	8072	240.0	253.0	360.0	1.0
612	LP002984	1	1	2	0	0	7583	0.0	187.0	360.0	1.0
613	LP002990	0	0	0	0	1	4583	0.0	133.0	360.0	0.0

614 rows × 13 columns

In [13]:
```
#changing the datype of each float column to int
data['Gender']=data['Gender'].astype('int64')
data['Married']=data['Married'].astype('int64')
data['Dependents']=data['Dependents'].astype('int64')
data['Self_Employed']=data['Self_Employed'].astype('int64')
data['CoapplicantIncome']=data['CoapplicantIncome'].astype('int64')
data['LoanAmount']=data['LoanAmount'].astype('int64')
data['Loan_Amount_Term']=data['Loan_Amount_Term'].astype('int64')
data['Credit_History']=data['Credit_History'].astype('int64')
```

Balancing the Dataset

In [14]:
```
#Balancing the dataset by using smote
from imblearn.combine import SMOTETomek
smote = SMOTETomek (0.95)
y = data['Loan_Status']
x = data.drop(columns=["Loan_ID",'Loan_Status'], axis=1)
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    353
0    331
Name: Loan_Status, dtype: int64
```

```
C:\Users\Arjun\AppData\Roaming\Python\Python39\site-packages\imblearn\utils\_validation.py:586: FutureWarning: Pass sampling_st
rategy=0.95 as keyword args. From version 0.9 passing these as positional arguments will result in an error
  warnings.warn(
```

Scaling the Data

In [15]:
```
sc=MaxAbsScaler()
x_bal_scaled=sc.fit_transform(x_bal)
x_bal_scaled = pd.DataFrame(x_bal,columns=x.columns)
```

In [16]:
```
x_bal_scaled
```

Out[16]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_
0	1	0	0	0	0	5849	0	120	360	1	
1	1	1	1	0	0	4583	1508	128	360	1	
2	1	1	0	0	1	3000	0	66	360	1	
3	1	1	0	1	0	2583	2358	120	360	1	
4	1	0	0	0	0	6000	0	141	360	1	
...	...	...	...	...	...	...	...	...	...	...	
679	1	1	0	1	0	3068	1793	120	351	0	
680	0	0	0	0	0	4585	0	126	360	0	
681	0	0	0	0	0	2442	1851	135	360	0	
682	0	0	0	0	0	2548	0	127	421	1	
683	1	1	0	1	0	2854	3257	137	290	0	

684 rows × 11 columns

Processed Data

In [17]:
```
final_df=pd.concat([x_bal_scaled,y_bal],axis=1)
```

In [18]:
```
final_df.to_csv("loan_data.csv")
```

In [36]:
```
final_df
```

Out[36]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_
0	1	0	0	0	0	5849	0	120	360	1	
1	1	1	1	0	0	4583	1508	128	360	1	
2	1	1	0	0	1	3000	0	66	360	1	
3	1	1	0	1	0	2583	2358	120	360	1	
4	1	0	0	0	0	6000	0	141	360	1	
...	...	...	...	...	...	...	...	...	...	...	
679	1	1	0	1	0	3068	1793	120	351	0	
680	0	0	0	0	0	4585	0	126	360	0	
681	0	0	0	0	0	2442	1851	135	360	0	
682	0	0	0	0	0	2548	0	127	421	1	
683	1	1	0	1	0	2854	3257	137	290	0	

684 rows × 12 columns

Saving into train test datasets

In [19]:
```
train,test = train_test_split(final_df, test_size=0.33, random_state=42)
```

In [20]:
```
train.to_csv('train.csv',encoding='utf-8',index=False)
test.to_csv('test.csv',encoding='utf-8',index=False)
```

Splitting the data ¶

In [21]:
```
x=final_df.drop(["Loan_Status"],axis=1)
```

In [22]:
```
x
```

Out[22]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Prope
0	1	0	0	0	0	5849	0	120	360	1	
1	1	1	1	0	0	4583	1508	128	360	1	
2	1	1	0	0	1	3000	0	66	360	1	
3	1	1	0	1	0	2583	2358	120	360	1	
4	1	0	0	0	0	6000	0	141	360	1	
...	...	...	...	...	...	...	...	...	...	...	
679	1	1	0	1	0	3068	1793	120	351	0	
680	0	0	0	0	0	4585	0	126	360	0	
681	0	0	0	0	0	2442	1851	135	360	0	
682	0	0	0	0	0	2548	0	127	421	1	
683	1	1	0	1	0	2854	3257	137	290	0	

In [23]:
```
y=final_df.Loan_Status
y
```

Out[23]:
```
0      1
1      0
2      1
3      1
4      1
      ..
679    0
680    0
681    0
682    0
683    0
Name: Loan_Status, Length: 684, dtype: int32
```

In [24]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

Building the Models

Descision tree

In [25]:
```
def decisionTree(x_train, x_test, y_train, y_test):
    dt=DecisionTreeClassifier()
    dt.fit(x_train,y_train)
    yPred = dt.predict(x_test)
    print('***DecisionTreeClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report (y_test,yPred))
    print("score")
    print(dt.score(x_test,y_test))
```

Random Forest

In [26]:
```
def randomForest(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
    print("score")
    print(rf.score(x_test,y_test))
```

KNN

In [27]:
```
def KNN(x_train, x_test, y_train, y_test):
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    yPred = knn.predict(x_test)
    print('***KNeighborsClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
    print("score")
    print(knn.score(x_test,y_test))
```

XGboost

In [28]:
```
def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print('***Gradient BoostingClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
    print("score")
    print(xg.score(x_test,y_test))
```

Comapring Models ¶

In [29]:
```
decisionTree(x_train, x_test, y_train, y_test)
```

```
***DecisionTreeClassifier***
Confusion matrix
[[52 10]
 [16 59]]
Classification report
              precision    recall  f1-score   support

           0       0.76      0.84      0.80        62
           1       0.86      0.79      0.82        75

    accuracy                           0.81       137
   macro avg       0.81      0.81      0.81       137
weighted avg       0.81      0.81      0.81       137

score
0.8102189781021898
```

In [30]:
```
randomForest(x_train, x_test, y_train, y_test)
```

```
***RandomForestClassifier***
Confusion matrix
[[47 15]
 [ 9 66]]
Classification report
              precision    recall  f1-score   support

           0       0.84      0.76      0.80        62
           1       0.81      0.88      0.85        75

    accuracy                           0.82       137
   macro avg       0.83      0.82      0.82       137
weighted avg       0.83      0.82      0.82       137

score
0.8248175182481752
```

In [31]:
```
KNN(x_train, x_test, y_train, y_test)
```

```
***KNeighborsClassifier***
Confusion matrix
[[38 24]
 [22 53]]
Classification report
              precision    recall  f1-score   support

           0       0.63      0.61      0.62        62
           1       0.69      0.71      0.70        75

    accuracy                           0.66       137
   macro avg       0.66      0.66      0.66       137
weighted avg       0.66      0.66      0.66       137

score
0.6642335766423357
```

In [32]:
```
xgboost(x_train, x_test, y_train, y_test)
```

```
***Gradient BoostingClassifier***
Confusion matrix
[[43 19]
 [ 5 70]]
Classification report
              precision    recall  f1-score   support

           0       0.90      0.69      0.78        62
           1       0.79      0.93      0.85        75

    accuracy                           0.82       137
   macro avg       0.84      0.81      0.82       137
weighted avg       0.84      0.82      0.82       137

score
0.8248175182481752
```

In [ ]:

Evaluating Performance Of The Model And Saving The Model

In [33]:
```
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')
cv = cross_val_score(rf,x,y,cv=5)
np.mean(cv)
```

Out[33]: 0.8217582653499356

In [34]:
```
pickle.dump(rf,open('rdf.pkl','wb'))
```

In [35]:
```
pickle.dump(sc,open("scalar.pkl","wb"))
```

Deployment

In [34]:
```
!pip install -U ibm-watson-machine-learning
```

```
Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.25
7)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-lear
ning) (0.3.3)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-wats
on-machine-learning) (1.3.4)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-l
earning) (21.3)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-lea
rning) (1.26.7)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-le
arning) (2.26.0)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-lea
rning) (2022.9.24)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-le
arning) (0.8.9)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson
-machine-learning) (2.11.0)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-
```

In [35]:
```
from ibm_watson_machine_learning import APIClient
import json
```

In [36]:
```
wml_credentials = {
    "apikey":"WNOYbQ3_-Vz-1DZg4sfdB_I9RU2ki-1BDilaXGFq3_P0",
    "url":"https://us-south.ml.cloud.ibm.com"
}
```

In [37]:
```
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
```

```
Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
------------------------------------  ------------  ------------------------
ID                                    NAME          CREATED
84db7564-cced-459a-a809-a1473f4d6a33  smart lender  2022-11-13T09:20:52.711Z
------------------------------------  ------------  ------------------------
```

In [38]:
```
SPACE_ID= "84db7564-cced-459a-a809-a1473f4d6a33"
```

In [39]:
```
wml_client.set.default_space(SPACE_ID)
```

Out[39]: 'SUCCESS'

In [40]:
```
wml_client.software_specifications.list(500)
```

```
-----------------------------  ------------------------------------  ----
NAME                           ASSET_ID                              TYPE
default_py3.6                  0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12      020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt     069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6        09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12     09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9      0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6          0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                     0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod   1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6              10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl      111e41b3-de2d-5422-a4d6-bf776828c4b7  base
autoai-kb_rt22.2-py3.10        125b6d9a-5b1f-5e8d-972a-b251688ccf40  base
runtime-22.1-py3.9             12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6        154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                   1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6         1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
kernel-spark3.3-r3.6           1c9e5454-f216-59dd-a20e-474a5cdf5988  base
pytorch-onnx_rt22.1-py3.9-edt  1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6           1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
```

In [41]:
```
import sklearn
sklearn.__version__
```

Out[41]: '1.0.2'

In [43]:
```
MODEL_NAME = 'samrt lender'
DEPLOYMENT_NAME = 'smart lender'
DEMO_MODEL = rf
```

In [44]:
```
# Set Python Version
software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
```

In [45]:
```
# Setup model meta
model_props = {
    wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
```

In [46]:
```
#Save model
model_details = wml_client.repository.store_model(
    model=DEMO_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)
```

```
In [47]: model_details
```

```
Out[47]: {'entity': {'hybrid_pipeline_software_specs': [],
  'label_column': 'Loan_Status',
  'schemas': {'input': [{'fields': [{'name': 'Unnamed: 0', 'type': 'int64'},
      {'name': 'Gender', 'type': 'int64'},
      {'name': 'Married', 'type': 'int64'},
      {'name': 'Dependents', 'type': 'int64'},
      {'name': 'Education', 'type': 'int64'},
      {'name': 'Self_Employed', 'type': 'int64'},
      {'name': 'ApplicantIncome', 'type': 'int64'},
      {'name': 'CoapplicantIncome', 'type': 'int64'},
      {'name': 'LoanAmount', 'type': 'int64'},
      {'name': 'Loan_Amount_Term', 'type': 'int64'},
      {'name': 'Credit_History', 'type': 'int64'},
      {'name': 'Property_Area', 'type': 'int64'}],
     'id': '1',
     'type': 'struct'}],
   'output': []},
  'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',
   'name': 'runtime-22.1-py3.9'},
  'type': 'scikit-learn_1.0'},
 'metadata': {'created_at': '2022-11-13T09:25:42.802Z',
  'id': '5d05ed33-c7f9-4bd5-a6d4-2d93ac85ec99',
  'modified_at': '2022-11-13T09:25:46.864Z',
  'name': 'samrt lender',
  'owner': 'IBMid-6610045N94',
  'resource_key': '5301de43-d983-4fd5-9b5f-c297d409e520',
  'space_id': '84db7564-cced-459a-a809-a1473f4d6a33'},
 'system': {'warnings': []}}
```

```
In [48]: model_id = wml_client.repository.get_model_id(model_details)
         model_id
```

```
Out[48]: '5d05ed33-c7f9-4bd5-a6d4-2d93ac85ec99'
```

```
In [49]: # Set meta
         deployment_props = {
             wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
             wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
         }
```

```
In [50]: # Deploy
         deployment = wml_client.deployments.create(
             artifact_uid=model_id,
             meta_props=deployment_props
         )
```

```
#######################################################################################

Synchronous deployment creation for uid: '5d05ed33-c7f9-4bd5-a6d4-2d93ac85ec99' started

#######################################################################################


initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='3ef47624-fc52-4a05-862e-23f7291f08ad'
------------------------------------------------------------------------------------------------
```

7.2 Feature 2

Developed a IBM_app.py file with integrated deployment and scoring points of IBM cloud.

```python
from flask import Flask, request, render_template
import joblib
import requests
#import jsonify
from flask import jsonify
import pickle


app = Flask(__name__)  # initialising flask app

with open("rdf.pkl","rb") as f:
    model=pickle.load(f)

@app.route('/', methods=['GET'])

def home():
    return render_template('index1.html')
@app.route('/predict1.html')
def formpg():
    return render_template('predict1.html')


```

```python

@app.route('/predict', methods=['POST', 'GET'])
def predict():
    if request.method == 'POST':
        GENDER = request.form['Gender']
        MARRIED=request.form['Married']
        DEPENDENTS=request.form['Dependents']
        EDUCATION = request.form['Education']
        SELF_EMPLOYES=request.form['Self_Employes']
        APPLICANTINCOME=request.form['ApplicantIncome']
        COAAPLICANTINCOME=request.form['CoaaplicantIncome']
        LOANAMOUNT= request.form['LoanAmount']
        LOAN_AMOUNT_TERM=request.form['Loan_Amount_Term']
        CREDIT_HISTORY=request.form['Credit_History']
        PROPERTY_AREA=request.form['Property_Area']
        if GENDER == 'Male':
            GENDER = 1
        else:
            GENDER = 0
        if MARRIED == 'yes':
            MARRIED = 1
        else:
            MARRIED = 0
```

app.py

```python
        else:
            MARRIED = 0
        if DEPENDENTS == '3+':
            DEPENDENTS = 3
        elif DEPENDENTS==1:
            DEPENDENTS=1
        elif DEPENDENTS==2:
            DEPENDENTS=2
        else:
            DEPENDENTS=0
        if EDUCATION == 'Graduate':
            EDUCATION = 0
        else:
            EDUCATION = 1
        if SELF_EMPLOYES == 'yes':
            SELF_EMPLOYES = 1
        else:
            SELF_EMPLOYES = 0
        if CREDIT_HISTORY == 'yes':
            CREDIT_HISTORY = 1
        else:
            CREDIT_HISTORY = 0
```

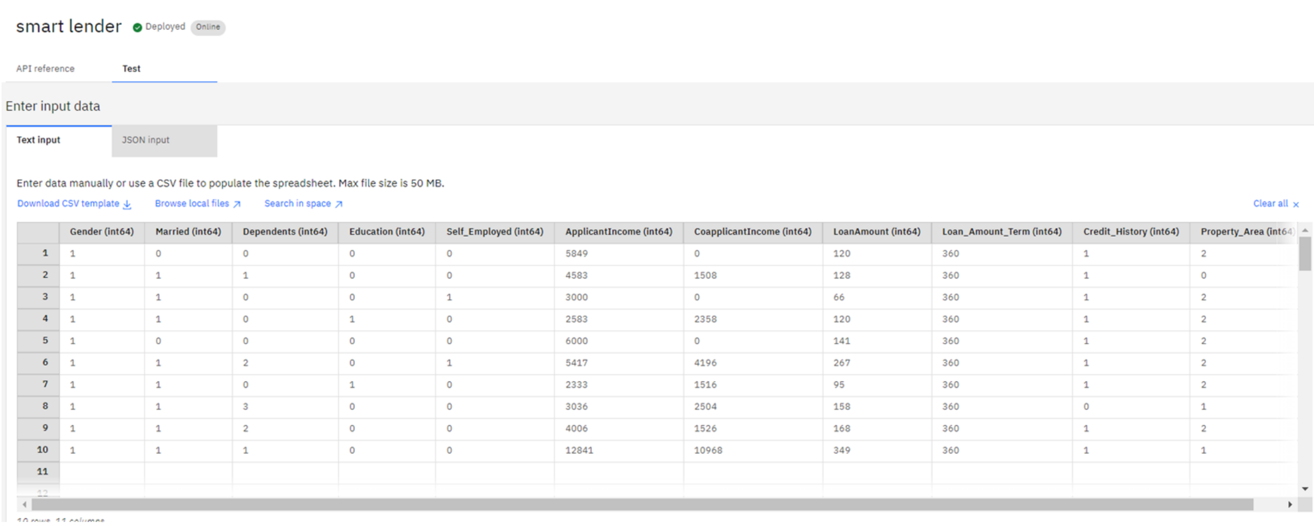
app.py

```python
            SELF_EMPLOYES = 0
        if CREDIT_HISTORY == 'yes':
            CREDIT_HISTORY = 1
        else:
            CREDIT_HISTORY = 0
        if  PROPERTY_AREA == 'Urban':
            PROPERTY_AREA = 2
        elif PROPERTY_AREA == 'Semiurban':
            PROPERTY_AREA = 1
        else:
            PROPERTY_AREA = 0
        prediction = model.predict([[GENDER, MARRIED, int(DEPENDENTS), EDUCATION, SELF_EMPLOYES, int(APPLICANTINCOME,
        output=prediction[0]
        if(output==1):
            return render_template('submit.html', prediction_text="Congratulations Your are Eligible for LOAN")
        else:
            return render_template('submit.html', prediction_text="Sorry, Your are Not Eligible for LOAN")
    else:
        return render_template('predict1.html')

if __name__ == '__main__':
    app.run(debug=True)
```

CHAPTER-8

TESTING

8.1 Test Cases



smart lender Deployed Online

API reference Test

Enter input data

Text input JSON input

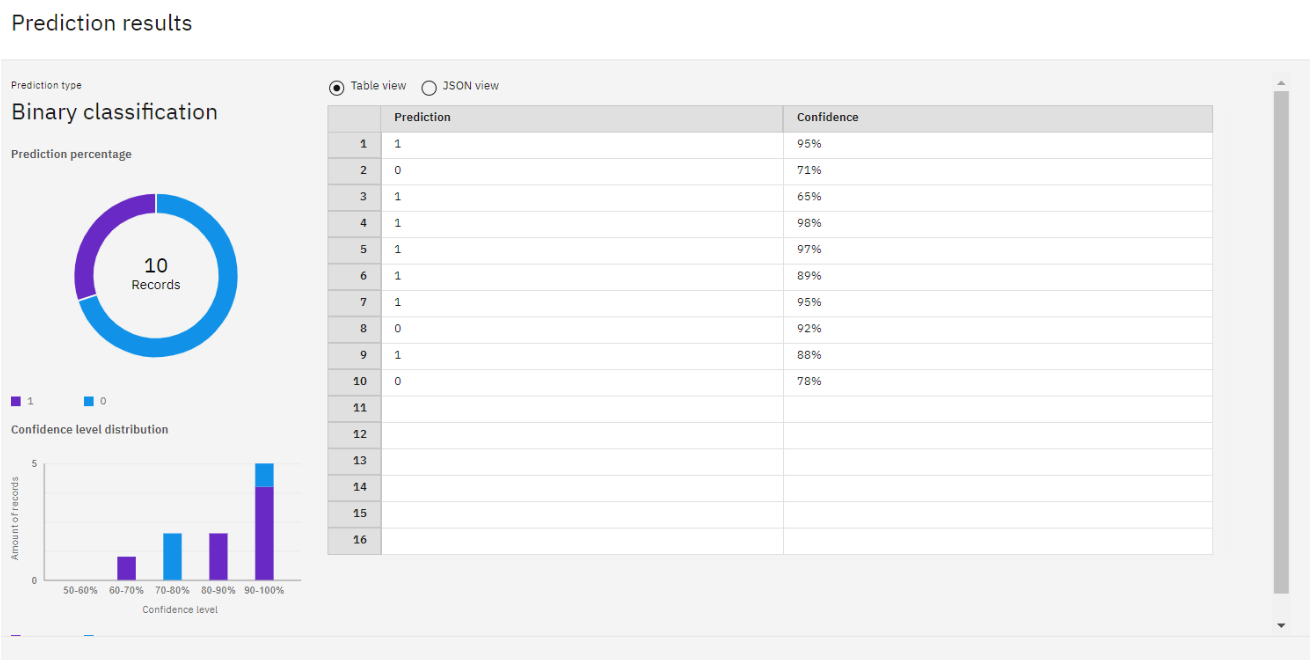
Enter data manually or use a CSV file to populate the spreadsheet. Max file size is 50 MB.

Download CSV template Browse local files Search in space Clear all

	Gender (int64)	Married (int64)	Dependents (int64)	Education (int64)	Self_Employed (int64)	ApplicantIncome (int64)	CoapplicantIncome (int64)	LoanAmount (int64)	Loan_Amount_Term (int64)	Credit_History (int64)	Property_Area (int64)
1	1	0	0	0	0	5849	0	120	360	1	2
2	1	1	1	0	0	4583	1508	128	360	1	0
3	1	1	0	0	1	3000	0	66	360	1	2
4	1	1	0	1	0	2583	2358	120	360	1	2
5	1	0	0	0	0	6000	0	141	360	1	2
6	1	1	2	0	1	5417	4196	267	360	1	2
7	1	1	0	1	0	2333	1516	95	360	1	2
8	1	1	3	0	0	3036	2504	158	360	0	1
9	1	1	2	0	0	4006	1526	168	360	1	2
10	1	1	1	0	0	12841	10968	349	360	1	1
11											

In the above we have given a different type of inputs to test our model.

8.2 User Acceptance Testing



	Prediction	Confidence
1	1	95%
2	0	71%
3	1	65%
4	1	98%
5	1	97%
6	1	89%
7	1	95%
8	0	92%
9	1	88%
10	0	78%

CHAPTER-9

9.RESULTS

9.1 Performance Metrics

There are various metrics which we can use to evaluate the performance of ML algorithms, classification as well as regression algorithms. We must carefully choose the metrics for evaluating ML performance because –

- How the performance of ML algorithms is measured and compared will be dependent entirely on the metric you choose.
- How you weight the importance of various characteristics in the result will be influenced completely by the metric you choose.

Importing the Libraries

In [1]:
```
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 ,MaxAbsScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

In [26]:
```
randomForest(x_train, x_test, y_train, y_test)
```

```
***RandomForestClassifier***
Confusion matrix
[[45 17]
 [ 3 72]]
Classification report
              precision    recall  f1-score   support

           0       0.94      0.73      0.82        62
           1       0.81      0.96      0.88        75

    accuracy                           0.85       137
   macro avg       0.87      0.84      0.85       137
weighted avg       0.87      0.85      0.85       137

score
0.8540145985401459
```

In [ ]:

CHAPTER-10
ADVANTAGES & DISADVANTAGES

Advantages

- The loan is not repayable on demand and so available for the term of the loan - generally three to ten years - unless you breach the loan conditions.
- Loans can be tied to the lifetime of the equipment or other assets you're borrowing the money to pay for.
- At the beginning of the term of the loan you may be able to negotiate a repayment holiday, meaning that you only pay interest for a certain amount of time while repayments on the capital are frozen.
- While you must pay interest on your loan, you do not have to give the lender a percentage of your profits or a share in your company.
- Interest rates may be fixed for the term so you will know the level of repayments throughout the life of the loan.
- There may be an arrangement fee that is paid at the start of the loan but not throughout its life. If it is an on-demand loan, an annual renewal fee may be payable.

Disadvantages

- Larger loans will have certain terms and conditions or covenants that you must adhere to, such as the provision of quarterly management information.

- Loans are not very flexible - you could be paying interest on funds you're not using.
- You could have trouble making monthly repayments if your customers don't pay you promptly, causing cashflow problems.
- In some cases, loans are secured against the assets of the business or your personal possessions, eg your home. The interest rates for secured loans may be lower than for unsecured ones, but your assets or home could be at risk if you cannot make the repayments.
- There may be a charge if you want to repay the loan before the end of the loan term, particularly if the interest rate on the loan is fixed.

CHAPTER-11
CONCLUSION

For the purpose of predicting the loan approval status of the applied customer,we have chosen the machinelearning approach to study the bank dataset.We have applied various machinelearning algorithms to decide which one will be the best for applying on the dataset to get the result with the highestaccuracy. Following this approach, we found that apart from the logistic regression, the rest of the algorithms performed satisfactory in terms of giving out the accuracy.The accuracy range of the rest of the algorithms were from 75% to 85%. Whereas the logisticregression gave us the best possible accuracy(88.70%) after the comparative study of all the algorithms.

We also determined the most importantfeatures that influence the loan approval status. These most important features are then used on some selected algorithms and their performance accuracy is compared with the instance of using all the features. This model can help the banks in figuringout which factorsare important for the loan approval procedure. The comparative study makes us clear about which algorithm will be the best and ignores the rest, based on their accuracy.

CHAPTER-12
FUTURE SCOPE

1.In future this project is going to be useful for making the loan prediction this will help people to check the loan eligibility before they are going to for any type of loans

2. In future updates these project will make a bigger change in the society for loan predictig.
3. and the model will imporve its accuracy with the help of new data inputs that the user is giving
4. This application can be used for online lon purposes.
5. Now a days the paylater and emis are increasing in every online retailing platform
6. So this application will help the lender to give the loan to the user those who are eligible.

CHAPTER-13
APPENDIX

Source Code GitHub & Project Demo Link

Source Code for Flask Application

ibm_app.py

```python
from flask import render_template, Flask, request
import numpy as np
import pickle
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "hmIOFhnjuvRGrJaKtFnyvNKEQTINuL4eRrcnbp6K7c8R"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
                               data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

app = Flask(__name__, template_folder='templates')

scale = pickle.load(open('scale.pkl', 'rb'))


@app.route('/')
def home():
    return render_template('index.html')


@app.route('/predict.html')
def formpg():
    return render_template('predict.html')

```

ibm_app.py

```python

@app.route('/submit', methods=['POST'])
def predict():
    loan_num, gender, married, depend, education, self_emp, applicant_income, co_income, loan_amount, loan_term, crec
        x for x in request.form.values()]
    if gender == 'Male':
        gender = 1
    else:
        gender = 0

    if married == 'Yes':
        married = 1
    else:
        married = 0

    if education == 'Graduate':
        education = 0
    else:
        education = 1

    if self_emp == 'Yes':
        self_emp = 1
    else:
        self_emp = 0

    if depend == '3+':
        depend = 3

```

if __name__ == "__main__"

```python
    loan_amount = int(loan_amount)
    loan_amount = np.log(loan_amount)

    if credit_history == 'Yes':
        credit_history = 1
    else:
        credit_history = 0

    if property_area == 'Urban':
        property_area = 2

    elif property_area == 'Rural':
        property_area = 0
    else:
        property_area = 1

    features = [[gender, married, depend, education, self_emp, applicant_income, co_income, loan_amount, loan_term,
                 credit_history, property_area]]
    # con_features = [np.array(features)]
    scale_features = scale.fit_transform(features)
    sf = scale_features.tolist()

    payload_scoring = {"input_data": [{"fields": ['gender', 'married', 'depend', 'education', 'self_emp',
                                                  'applicant_income', 'co_income', 'loan_amount', 'loan_term',
                                                  'credit_history', 'property_area'], "values": sf}]}

    response_scoring = requests.post(
        'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/5108313c-f101-4c06-8f87-151aa0d1c3ff/predictions?version
```

```python
    scale_features = scale.fit_transform(features)
    sf = scale_features.tolist()

    payload_scoring = {"input_data": [{"fields": ['gender', 'married', 'depend', 'education', 'self_emp',
                                                  'applicant_income', 'co_income', 'loan_amount', 'loan_term',
                                                  'credit_history', 'property_area'], "values": sf}]}

    response_scoring = requests.post(
        'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/5108313c-f101-4c06-8f87-151aa0d1c3ff/predictions?version
        json=payload_scoring, headers={'Authorization': 'Bearer ' + mltoken})
    print("response_scoring")
    prediction = response_scoring.json()
    predict = prediction['predictions'][0]['values'][0][0]

    # prediction = model.predict(scale_features)
    if predict == 0:
        return render_template('submit.html', prediction_text='You are eligible for loan')
    else:
        return render_template('submit.html', prediction_text='Sorry you are not eligible for loan')


if __name__ == "__main__":
    app.run(debug=True)
```

#Flask App using API_KEY:

```python
ibm_app.py × app.py ×
from flask import Flask, request, render_template
import requests
from flask import jsonify
import json

API_KEY = "WNOYbQ3_-Vz-1DZg4sfdB_I9RU2ki-1BDilaXGFq3_P0"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":API_KEY, "grant_type": 'ur
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}


app = Flask(__name__)  # initialising flask app



@app.route('/', methods=['GET'])

def home():
    return render_template('index1.html')
@app.route('/predict1.html')
def formpg():
    return render_template('predict1.html')


@app.route('/predict', methods=['POST', 'GET'])
def predict():
    if request.method == 'POST':
```

```python
ibm_app.py × app.py ×
    if request.method == 'POST':
        GENDER = request.form['Gender']
        MARRIED=request.form['Married']
        DEPENDENTS=request.form['Dependents']
        EDUCATION = request.form['Education']
        SELF_EMPLOYES=request.form['Self_Employes']
        APPLICANTINCOME=request.form['ApplicantIncome']
        COAAPLICANTINCOME=request.form['CoaaplicantIncome']
        LOANAMOUNT= request.form['LoanAmount']
        LOAN_AMOUNT_TERM=request.form['Loan_Amount_Term']
        CREDIT_HISTORY=request.form['Credit_History']
        PROPERTY_AREA=request.form['Property_Area']
        if GENDER == 'Male':
            GENDER = 1
        else:
            GENDER = 0
        if MARRIED == 'yes':
            MARRIED = 1
        else:
            MARRIED = 0
        if DEPENDENTS == '3+':
            DEPENDENTS = 3
        elif DEPENDENTS==1:
            DEPENDENTS=1
        elif DEPENDENTS==2:
            DEPENDENTS=2
        else:
            DEPENDENTS=0
        if EDUCATION == 'Graduate':
            EDUCATION = 0
```

ibm_app.py × app.py ×

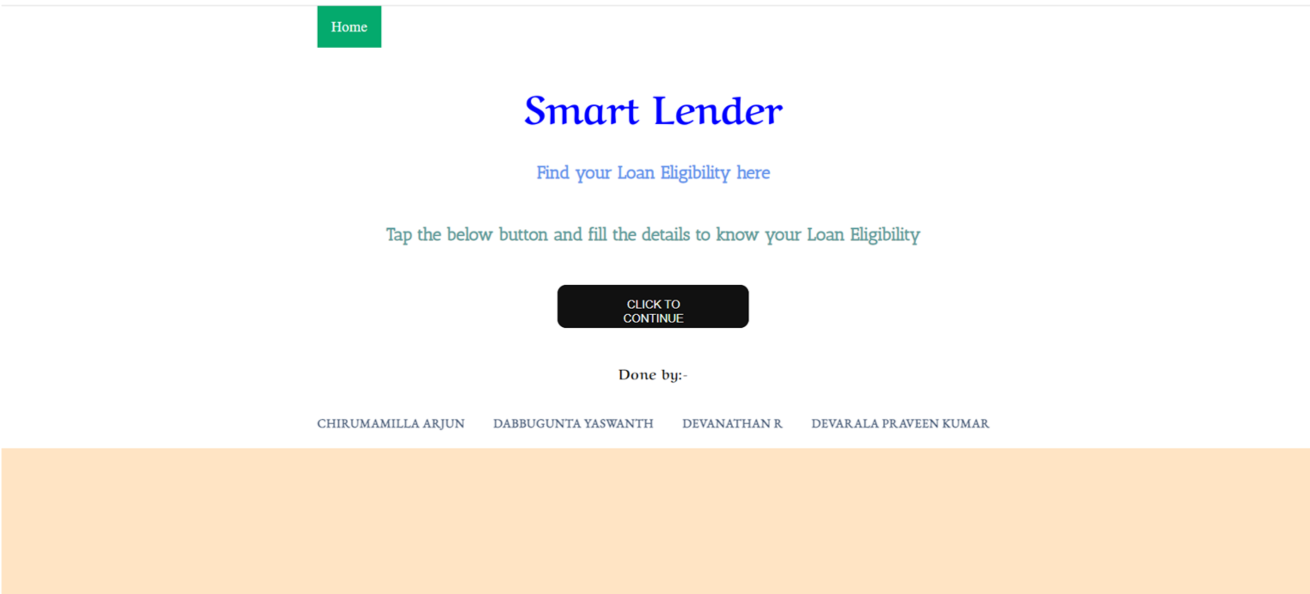
```python
        else:
            EDUCATION = 1
        if SELF_EMPLOYES == 'yes':
            SELF_EMPLOYES = 1
        else:
            SELF_EMPLOYES = 0
        if CREDIT_HISTORY == 'yes':
            CREDIT_HISTORY = 1
        else:
            CREDIT_HISTORY = 0
        if  PROPERTY_AREA == 'Urban':
            PROPERTY_AREA = 2
        elif PROPERTY_AREA == 'Semiurban':
            PROPERTY_AREA = 1
        else:
            PROPERTY_AREA = 0
        prediction =[[GENDER, MARRIED, int(DEPENDENTS), EDUCATION, SELF_EMPLOYES, int(APPLICANTINCOME), int(COAAPLICANTINCOME), int(LO
        payload_scoring = {
        "input_data": [
                {
                        "field": [
                                "Gender",
                                "Married",
                                "Dependents",
                                "Education",
                                "Self_Employed",
                                "ApplicantIncome",
                                "CoapplicantIncome",
                                "LoanAmount",
                                "Loan_Amount_Term",
```

ibm_app.py × app.py ×

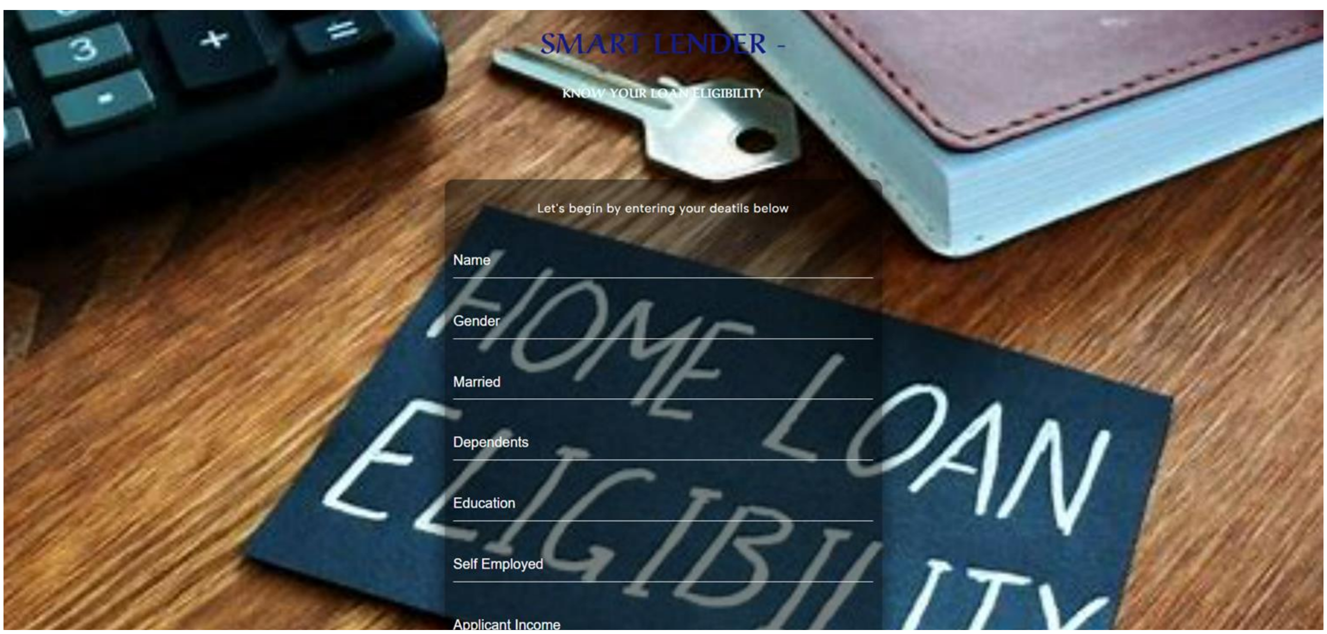
```python
                                "ApplicantIncome",
                                "CoapplicantIncome",
                                "LoanAmount",
                                "Loan_Amount_Term",
                                "Credit_History",
                                "Property_Area"
                        ],
                        "values": prediction
                }
        ]
}

        response_scoring = requests.post(
            'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/e3b81466-5919-4ba7-a4d9-13f2096d3f13/predictions?version=2022-11-13',
            json=payload_scoring,
            headers={'Authorization': 'Bearer ' + mltoken})
        print("Scoring response")
        print(response_scoring.json())
        output=prediction[0]
        if(output==1):
            return render_template('submit.html', prediction_text="Congratulations Your are Eligible for LOAN")
        else:
            return render_template('submit.html', prediction_text="Sorry, Your are Not Eligible for LOAN")
    else:
        return render_template('predict1.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Front End Code HTML Files
1.Index.html

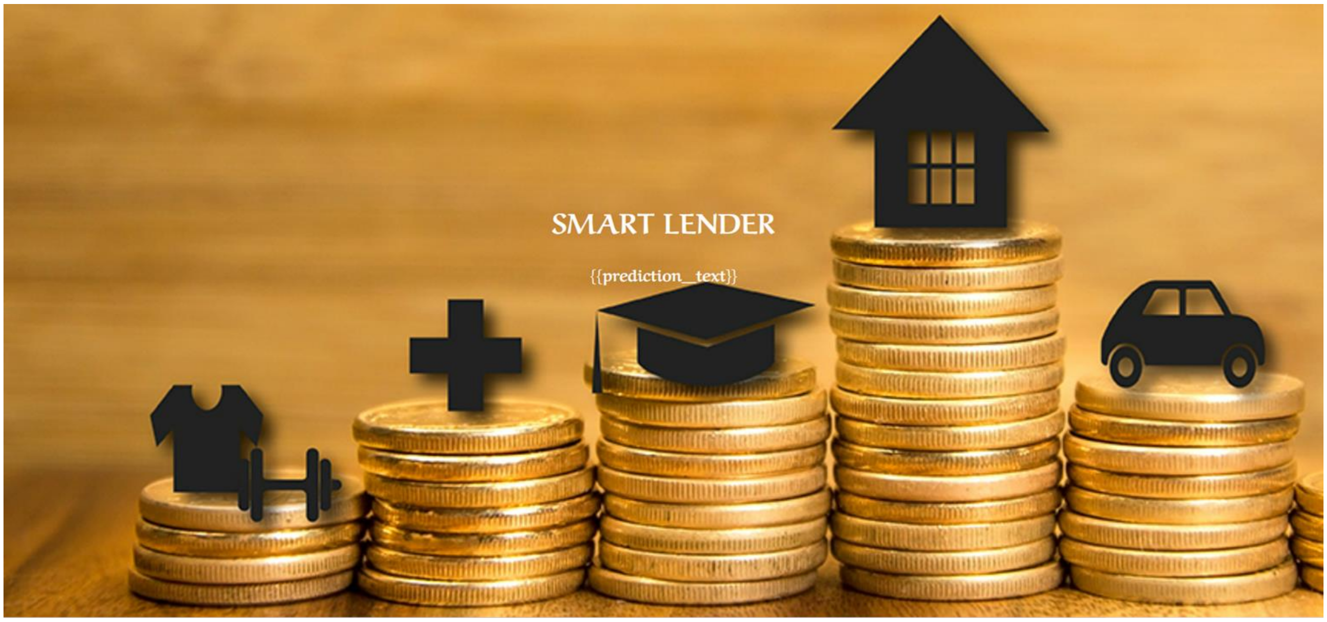
#result of index page



2.Predict.hmtl



3.Submit.html



GITHUB : https://github.com/IBM-EPBL/IBM-Project-27353-1660054443

DEMO_LINK:

https://www.youtube.com/watch?v=LAI5SzyeJOk