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 [1]. Commercial loans have always been a big part of the banking industry, and lenders are always aiming to reduce their credit risk [5]. 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 [1]. 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 [5].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 [1] 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 [2]. 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.

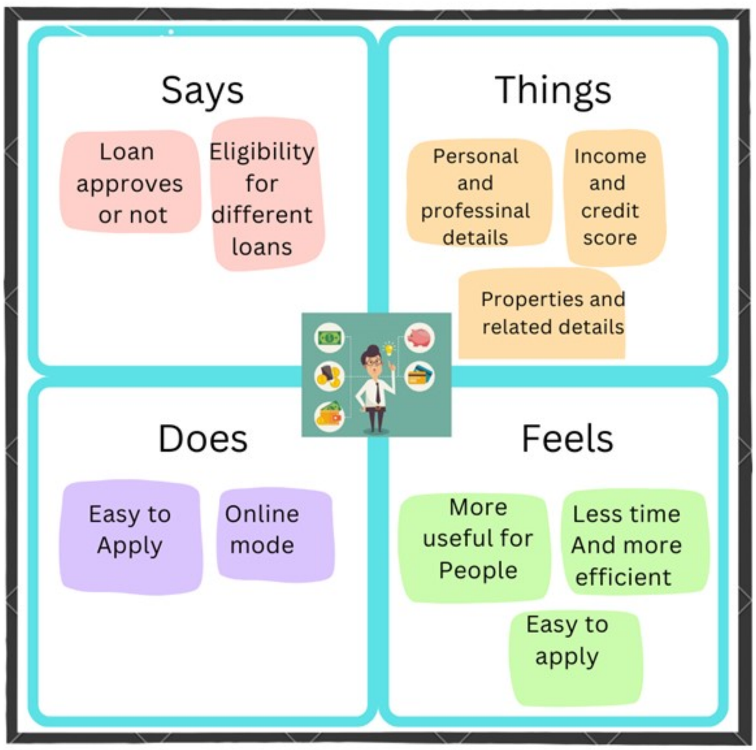
2.3 Problem Statement Defination

- 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.

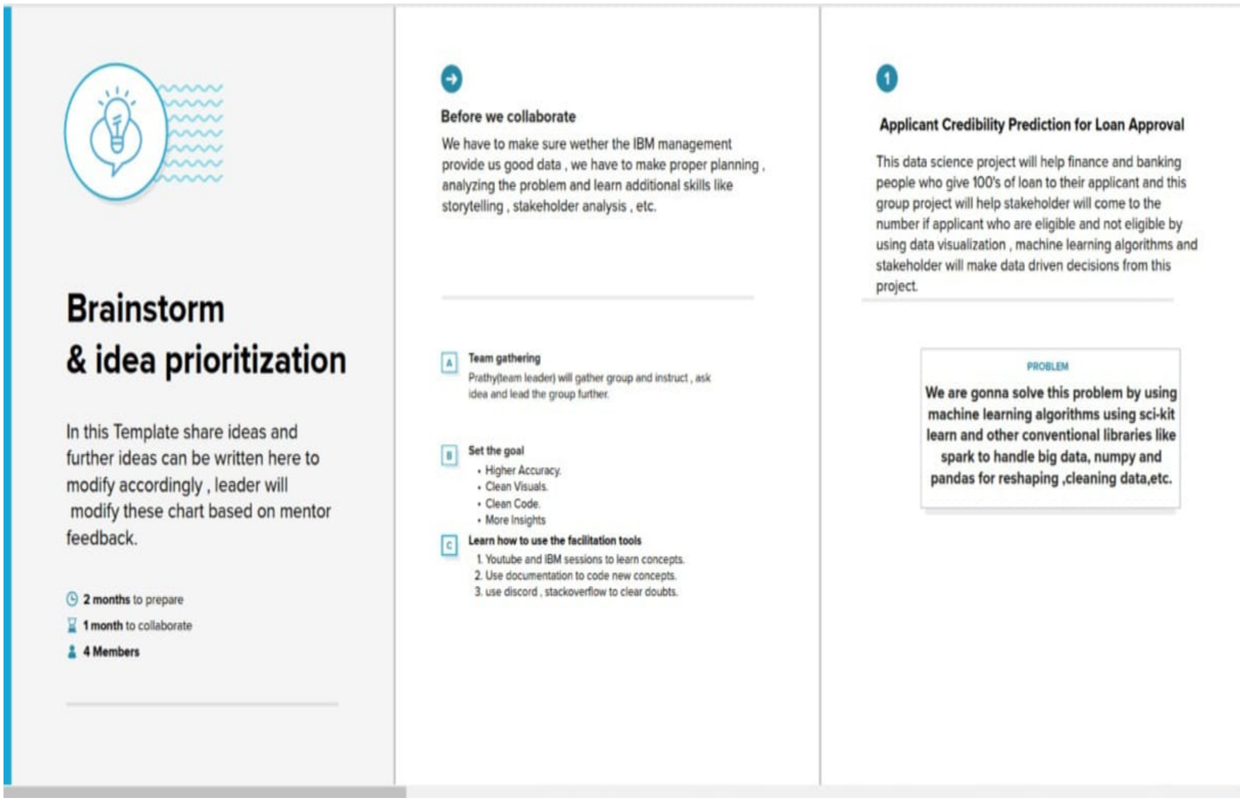
CHAPTER-3

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 be solved)	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	6. CUSTOMER CONSTRAINTS CC	5. AVAILABLE SOLUTIONS AS	Explore AS, differentiate
	I. Bank higher authority. II. Bank decision makers. III. Stakeholders and customers. IV. Persons who are giving and applying for loans.	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.	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.	

Focus on J&P, tap into BE, understand RC	2. JOBS-TO-BE-DONE / PROBLEMS J&P	9. PROBLEM ROOT CAUSE RC	7. BEHAVIOUR BE	Focus on J&P, tap into BE, understand RC
	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.	I. Faster loan approval . II. Profit for stakeholders. III. Maintain standards in company. IV. Scalability.	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.	

3. TRIGGERS TR	10. YOUR SOLUTION SL	1. CHANNELS of BEHAVIOUR CH
A. Scope of ML and data science increasesd ay byday. B. Financial and Banks arein need of fasterloan approval model.	1. Providing cleaner visuals to stakeholders. 2. Helping higherlevel and employees to takedatadriven decision. 3. More accuracy ML model forpredicting customerdat a.	a. ONLI NE Online loan approval system - By online services of company customers can know their loan eligibility. b. OFFLI NE

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 Confirmationvia 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

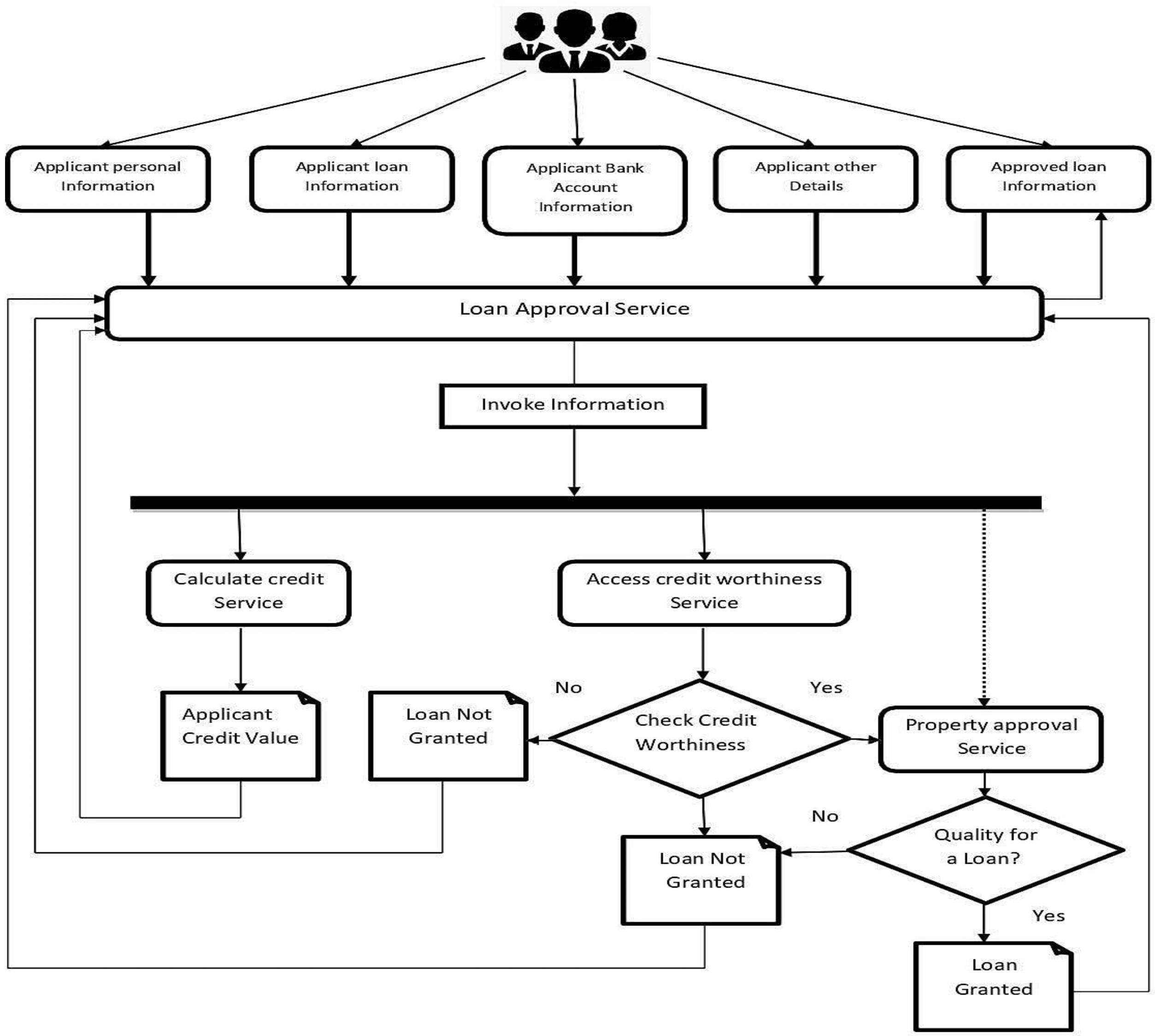
5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

Solution Architecture

1. 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.
2. 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.
3. The userscan get the results in the comfort of their home.
4. 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	User Story Numb	User Story / Task	Acceptance criteria	Priority	Release

	(Epic)	**er**				
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 entering my email, password, and confirming my password.	3	High	Akhil Anvesh Praveen Thangathamil
Sprint-1		USN-2	As a user, I will receive confirmation email once I haveregistered for the application	3	High	Akhil Anvesh Praveen Thangathamil

Sprint-1		USN-3	As a user, I can register for the application through Facebook	1	Low	Akhil Anvesh Praveen Thangathamil
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Akhil Anvesh Praveen Thangathamil
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	Akhil Anvesh Praveen Thangathamil

Sprint-1	Dashboard	USN-6	As a user, I should be able to access the dashboard with everything I am allowed touse.	2	Medium	Akhil Anvesh Praveen Thangathamil

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (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

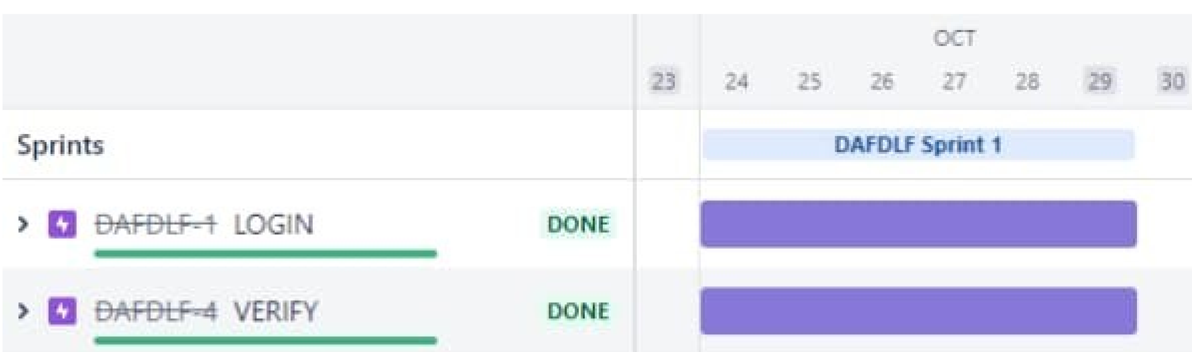
Velocity

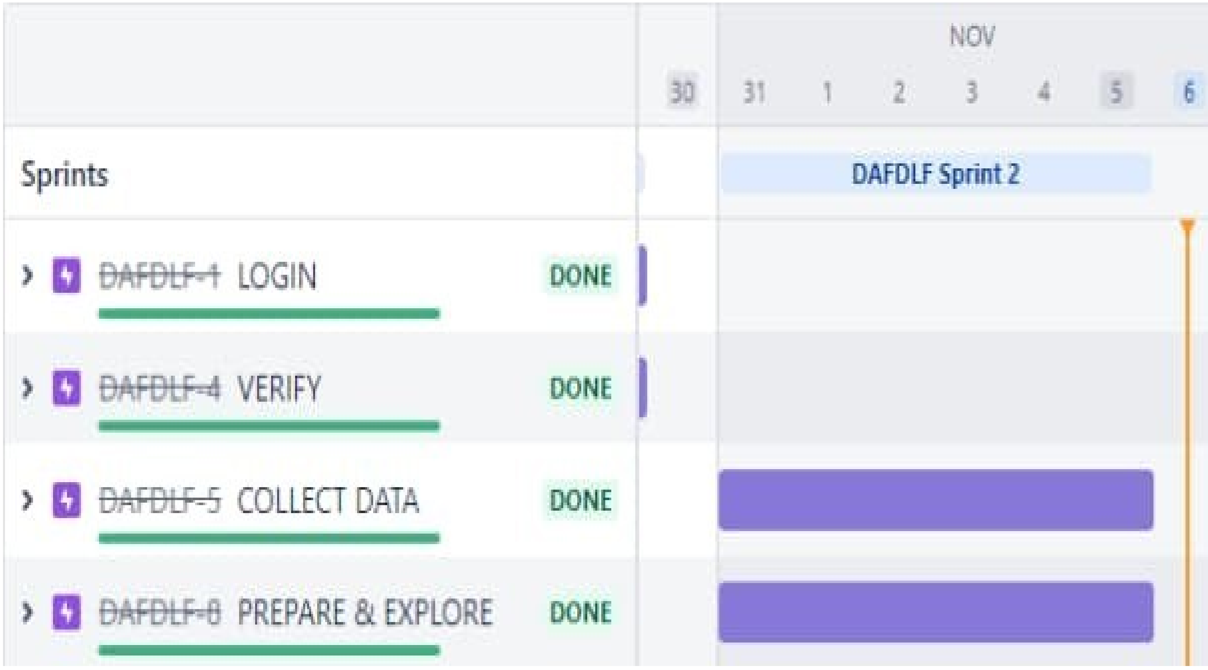
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) periteration unit (storypoints per day)

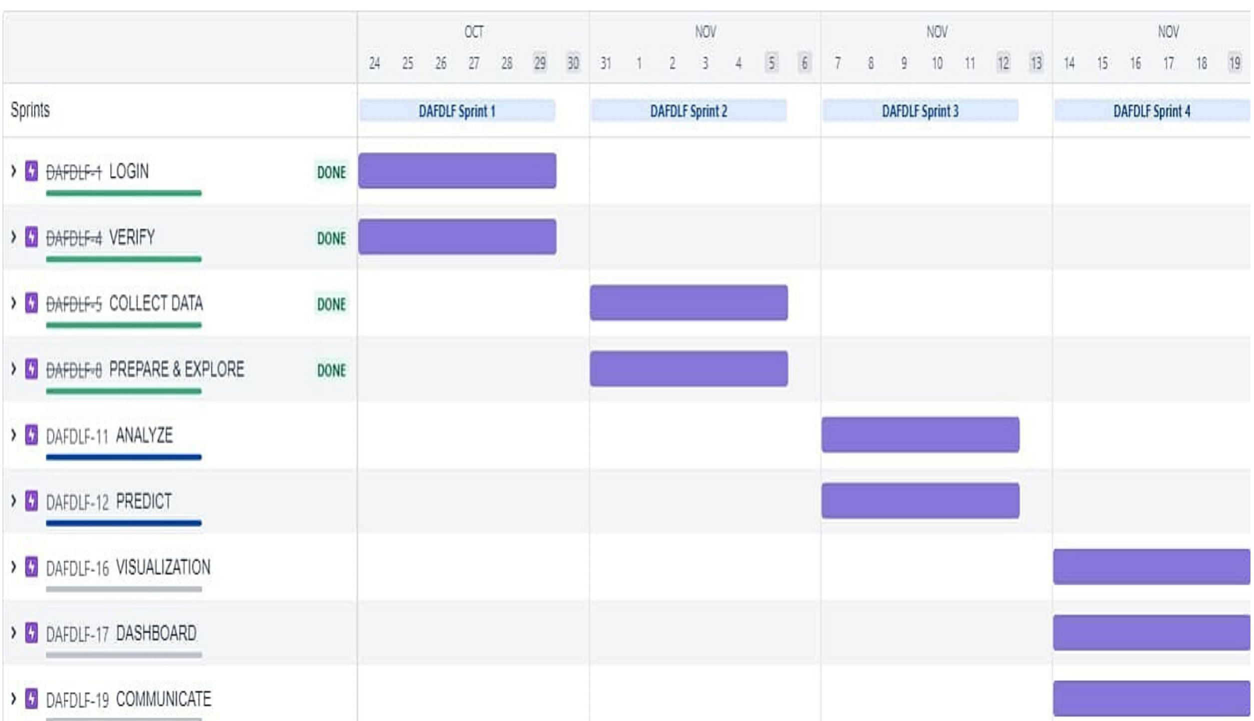
Burndown Chart

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

6.3 Reports From JIRA

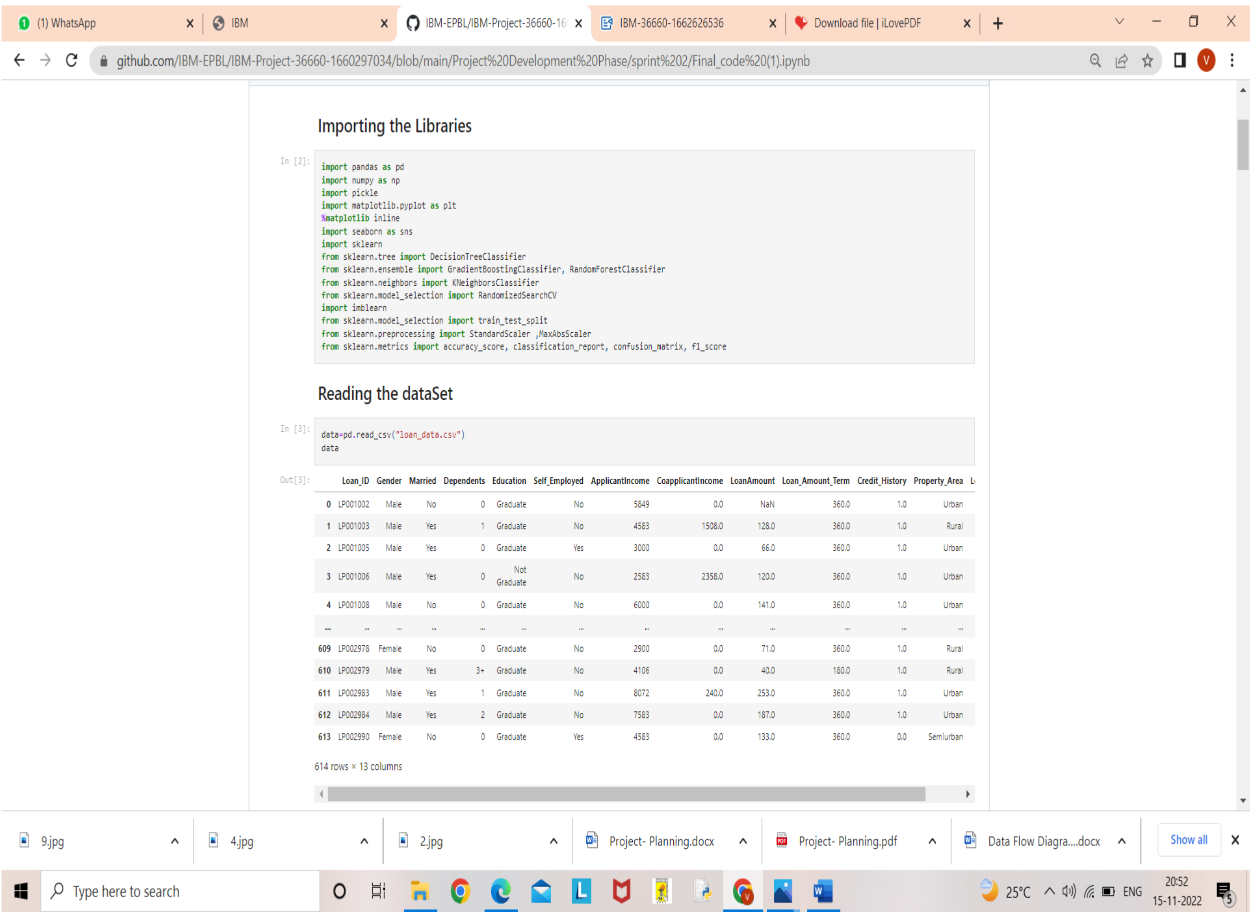



NOV
30 31 1 2 3 4 5 6
Sprints
DAFDLF Sprint 2
~~DAFDLF-1~~ LOGIN DONE
~~DAFDLF-4~~ VERIFY DONE
~~DAFDLF-5~~ COLLECT DATA DONE
~~DAFDLF-8~~ PREPARE & EXPLORE DONE


OCT NOV NOV NOV
24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Sprints
DAFDLF Sprint 1
DAFDLF Sprint 2
DAFDLF Sprint 3
DAFDLF Sprint 4
~~DAFDLF-1~~ LOGIN DONE
~~DAFDLF-4~~ VERIFY DONE
~~DAFDLF-5~~ COLLECT DATA DONE
~~DAFDLF-8~~ PREPARE & EXPLORE DONE
DAFDLF-11 ANALYZE
DAFDLF-12 PREDICT
DAFDLF-16 VISUALIZATION
DAFDLF-17 DASHBOARD
DAFDLF-19 COMMUNICATE

CHAPTER-7
CODING AND SOLUTIONING

7.1 Feature 1



Importing the Libraries

In [2]:
```
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 [3]:
```
data=pd.read_csv("loan_data.csv")
data
```

Out[3]:

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

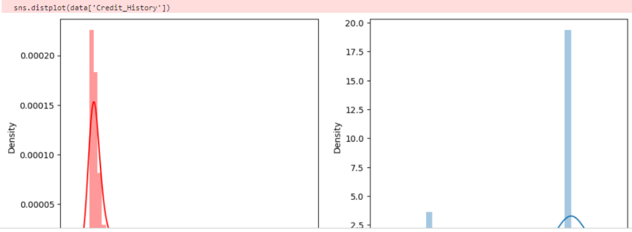
614 rows × 13 columns

Visualizations

In [7]:

```
#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()
```

```
C:\Users\72081\AppData\Local\Temp\ipykernel_14816\3941809966.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['ApplicantIncome'], color='r')
C:\Users\72081\AppData\Local\Temp\ipykernel_14816\3941809966.py:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['Credit_History'])
```



github.com/IBM-EPBL/IBM-Project-36660-1660297034/blob/main/Project%20Development%20Phase/sprint%202/Final_code%20(1).ipynb

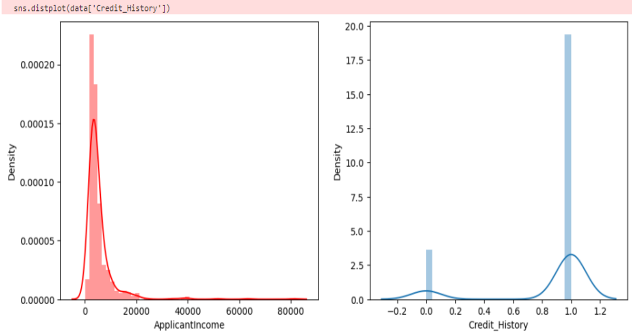
C:\Users\72081\AppData\Local\Temp\ipykernel_14816\3941809966.py:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

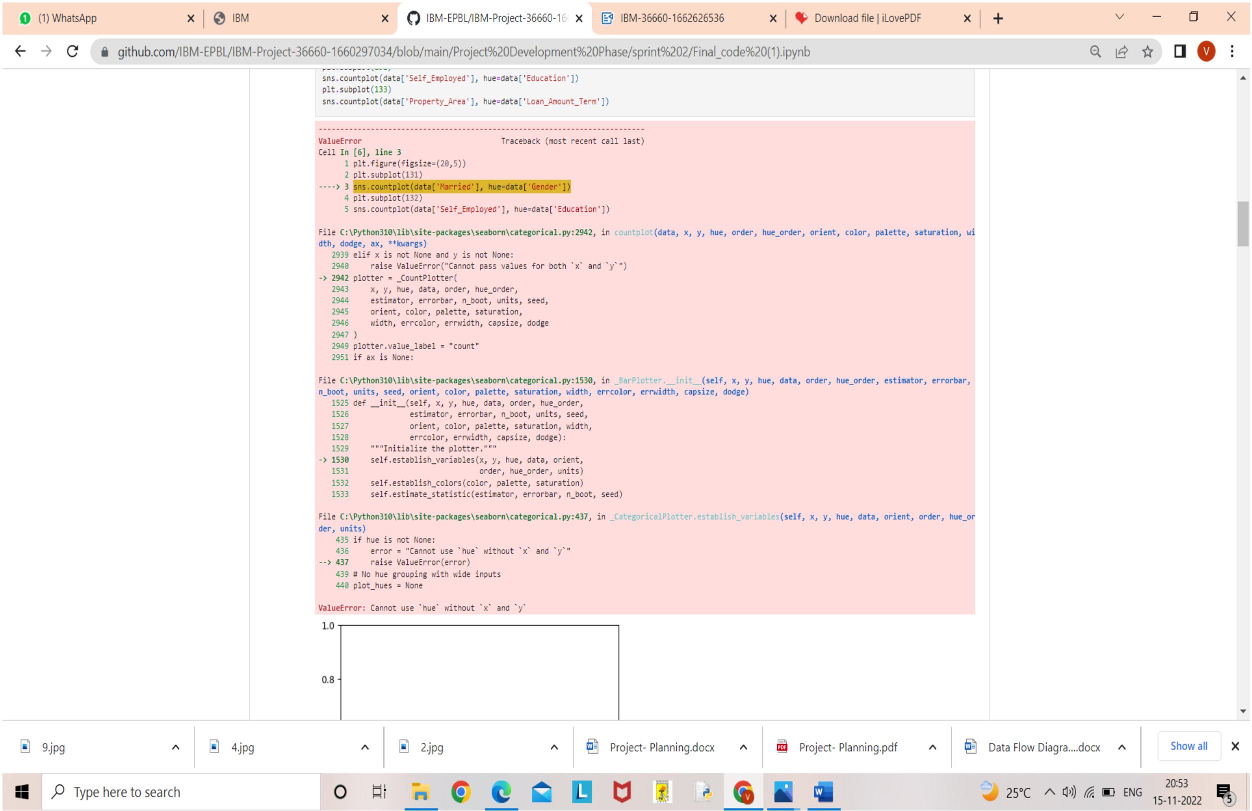
  sns.distplot(data['Credit_History'])

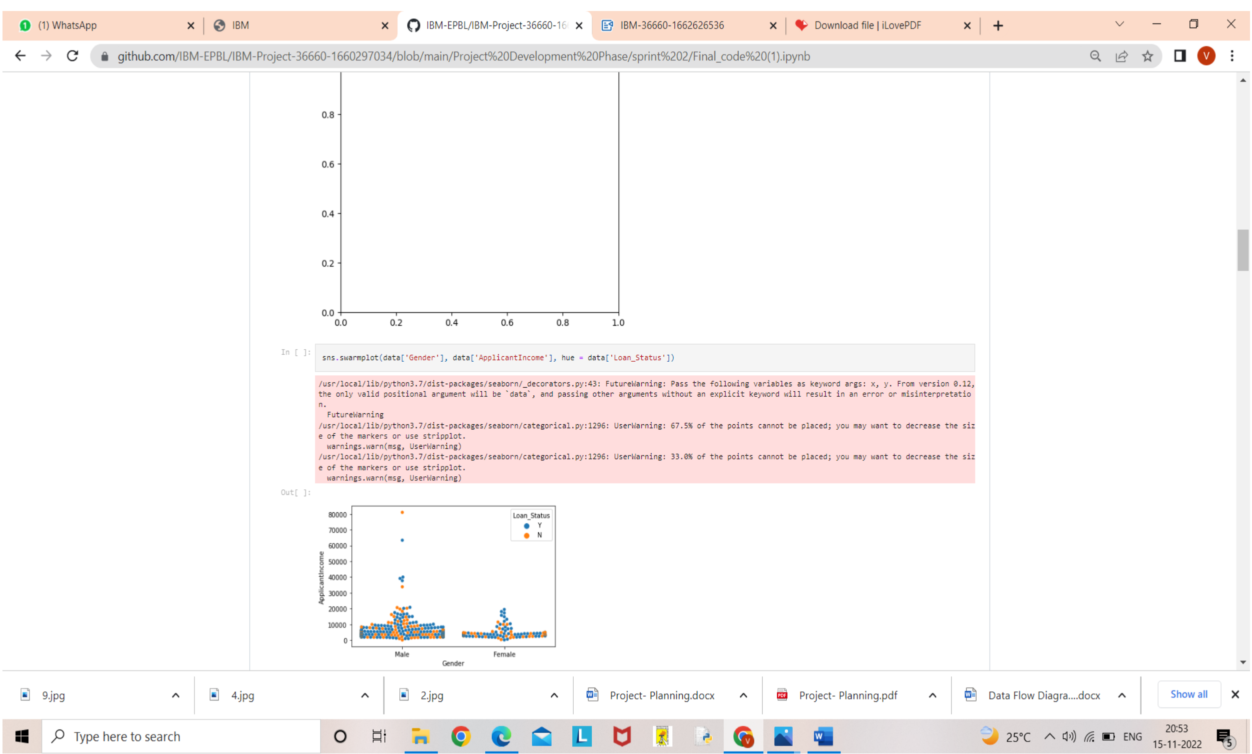


In [6]:

```
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'])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [6], line 3
      1 plt.figure(figsize=(20,5))
      2 plt.subplot(131)
----> 3 sns.countplot(data['Married'], hue=data['Gender'])
      4 plt.subplot(132)
      5 sns.countplot(data['Self_Employed'], hue=data['Education'])
```


(1) WhatsApp
IBM
IBM-EPBL/IBM-Project-36660-16
IBM-36660-1662626536
Download file | iLovePDF
github.com/IBM-EPBL/IBM-Project-36660-1660297034/blob/main/Project%20Development%20Phase/sprint%202/Final_code%20(1).ipynb
sns.countplot(data['Self_Employed'], hue=data['Education'])
plt.subplot(133)
sns.countplot(data['Property_Area'], hue=data['Loan_Amount_Term'])
ValueError Traceback (most recent call last)
Cell In [6], line 3
1 plt.figure(figsize=(20,5))
2 plt.subplot(131)
----> 3 sns.countplot(data['Married'], hue=data['Gender'])
4 plt.subplot(132)
5 sns.countplot(data['Self_Employed'], hue=data['Education'])
File C:\Python310\lib\site-packages\seaborn\categorical.py:2942, in countplot(data, x, y, hue, order, hue_order, orient, color, palette, saturation, width, dodge, ax, **kwargs)
2939 elif x is not None and y is not None:
2940 raise ValueError("Cannot pass values for both `x` and `y`")
-> 2942 plotter = _CountPlotter(
2943 x, y, hue, data, order, hue_order,
2944 estimator, errorbar, n_boot, units, seed,
2945 orient, color, palette, saturation,
2946 width, errcolor, errwidth, capsize, dodge
2947)
2949 plotter.value_label = "count"
2951 if ax is None:
File C:\Python310\lib\site-packages\seaborn\categorical.py:1530, in _BarPlotter.__init__(self, x, y, hue, data, order, hue_order, estimator, errorbar, n_boot, units, seed, orient, color, palette, saturation, width, errcolor, errwidth, capsize, dodge)
1525 def __init__(self, x, y, hue, data, order, hue_order,
1526 estimator, errorbar, n_boot, units, seed,
1527 orient, color, palette, saturation, width,
1528 errcolor, errwidth, capsize, dodge):
1529 """Initialize the plotter."""
-> 1530 self.establish_variables(x, y, hue, data, orient,
1531 order, hue_order, units)
1532 self.establish_colors(color, palette, saturation)
1533 self.estimate_statistic(estimator, errorbar, n_boot, seed)
File C:\Python310\lib\site-packages\seaborn\categorical.py:437, in _CategoricalPlotter.establish_variables(self, x, y, hue, data, orient, order, hue_order, units)
435 if hue is not None:
436 error = "Cannot use `hue` without `x` and `y`"
--> 437 raise ValueError(error)
439 # No hue grouping with wide inputs
440 plot_hues = None
ValueError: Cannot use `hue` without `x` and `y`
1.0
0.8
9.jpg
4.jpg
2.jpg
Project- Planning.docx
Project- Planning.pdf
Data Flow Diagra....docx
Show all
Type here to search
25°C
ENG
20:53
15-11-2022


(1) WhatsApp
IBM
IBM-EPBL/IBM-Project-36660-16
IBM-36660-1662626536
Download file | iLovePDF
github.com/IBM-EPBL/IBM-Project-36660-1660297034/blob/main/Project%20Development%20Phase/sprint%202/Final_code%20(1).ipynb
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
In [ ]:
sns.swarmplot(data['Gender'], data['ApplicantIncome'], hue = data['Loan_Status'])
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 67.5% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 33.0% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
warnings.warn(msg, UserWarning)
Out[ ]:
Loan_Status
Y
N
80000
70000
60000
50000
40000
30000
20000
10000
0
ApplicantIncome
Male
Female
Gender
9.jpg
4.jpg
2.jpg
Project- Planning.docx
Project- Planning.pdf
Data Flow Diagra....docx
Show all
Type here to search
25°C
ENG
20:53
15-11-2022

Data Pre-processing

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

Out[ ]:

	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 [ ]:

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

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


(1) WhatsApp | IBM | IBM-EPBL/IBM-Project-36660-16 | IBM-36660-1662626536 | Download file | iLovePDF
github.com/IBM-EPBL/IBM-Project-36660-1660297034/blob/main/Project%20Development%20Phase/sprint%202/Final_code%20(1).ipynb
memory usage: 62.5+ KB
Handling the Null Values
In [ ]: data.isnull().sum()

Out[ ]:
```
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 [ ]:
```
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])
```

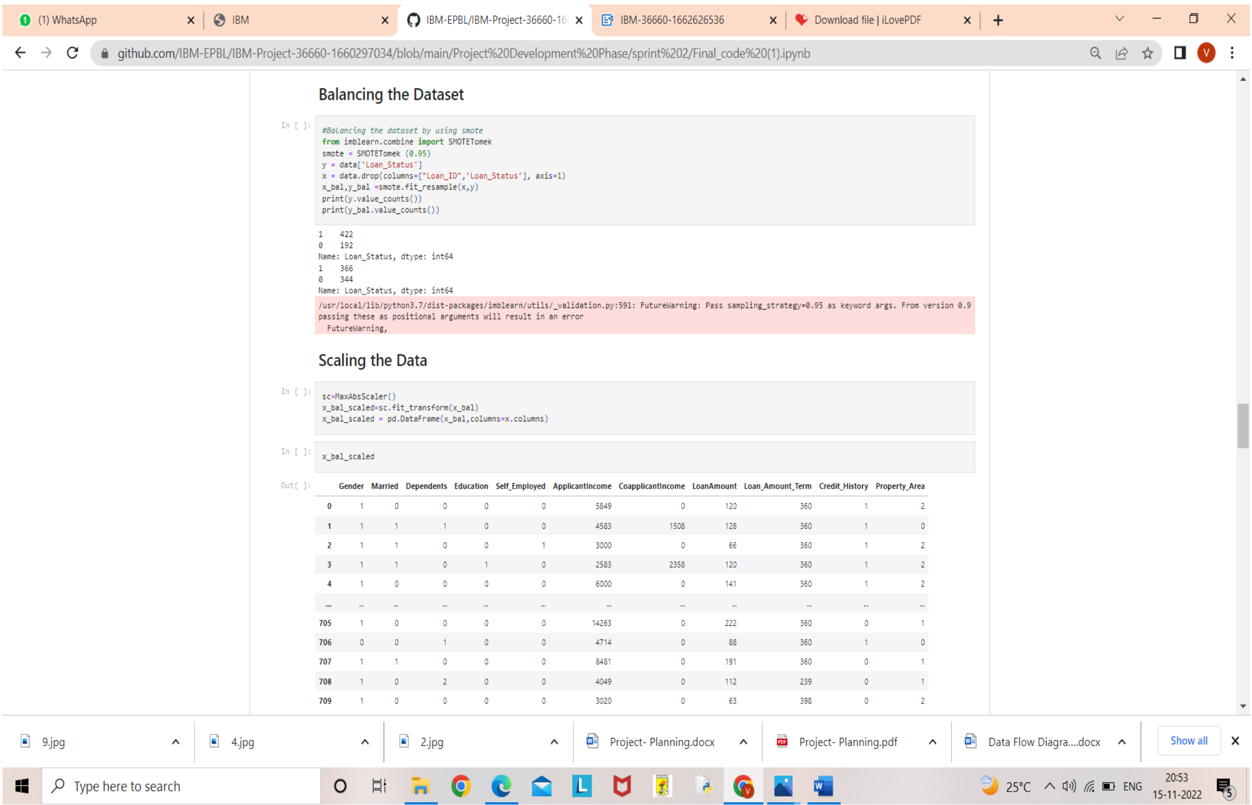
In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
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

9.jpg | 4.jpg | 2.jpg | Project- Planning.docx | Project- Planning.pdf | Data Flow Diagra....docx | Show all
Type here to search 25°C ENG 20:53 15-11-2022

7.2 Feature 2


Balancing the Dataset

```
#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    366
0    344
Name: Loan_Status, dtype: int64
/usr/local/lib/python3.7/dist-packages/imblearn/utils/_validation.py:591: FutureWarning: Pass sampling_strategy=0.95 as keyword args. From version 0.9 passing these as positional arguments will result in an error
  FutureWarning,
```

Scaling the Data

```
sc=MaxAbsScaler()
x_bal_scaled=sc.fit_transform(x_bal)
x_bal_scaled = pd.DataFrame(x_bal,columns=x.columns)
```

```
x_bal_scaled
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	1	0	0	0	0	5849	0	120	360	1	2
1	1	1	1	0	0	4583	1508	128	360	1	0
2	1	1	0	0	1	3000	0	66	360	1	2
3	1	1	0	1	0	2583	2358	120	360	1	2
4	1	0	0	0	0	6000	0	141	360	1	2
...	...	...	...	...	...	...	...	...	...	...	...
705	1	0	0	0	0	14263	0	222	360	0	1
706	0	0	1	0	0	4714	0	88	360	1	0
707	1	1	0	0	0	8481	0	191	360	0	1
708	1	0	2	0	0	4049	0	112	239	0	1
709	1	0	0	0	0	3020	0	63	398	0	2

Processed Data

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

In [ ]:
```
final_df
```

Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	1	0	0	0	0	5849	0	120	360	1	2	1
1	1	1	1	0	0	4583	1508	128	360	1	0	0
2	1	1	0	0	1	3000	0	66	360	1	2	1
3	1	1	0	1	0	2583	2358	120	360	1	2	1
4	1	0	0	0	0	6000	0	141	360	1	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...
705	1	0	0	0	0	14263	0	222	360	0	1	0
706	0	0	1	0	0	4714	0	88	360	1	0	0
707	1	1	0	0	0	8481	0	191	360	0	1	0
708	1	0	2	0	0	4049	0	112	239	0	1	0
709	1	0	0	0	0	3020	0	63	398	0	2	0

710 rows × 12 columns

Saving into train test datasets

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

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

Splitting the data

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

In [ ]:
```
x
```


(1) WhatsApp | IBM | IBM-EPBL/IBM-Project-36660-16 | IBM-36660-1662626536 | Download file | iLovePDF
github.com/IBM-EPBL/IBM-Project-36660-1660297034/blob/main/Project%20Development%20Phase/sprint%202/Final_code%20(1).ipynb

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

Splitting the data

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

In [ ]:
```
x
```


Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	1	0	0	0	0	5849	0	120	360	1	2
1	1	1	1	0	0	4583	1508	128	360	1	0
2	1	1	0	0	1	3000	0	66	360	1	2
3	1	1	0	1	0	2583	2358	120	360	1	2
4	1	0	0	0	0	6000	0	141	360	1	2
...	...	...	...	...	...	...	...	...	...	...	...
705	1	0	0	0	0	14263	0	222	360	0	1
706	0	0	1	0	0	4714	0	88	360	1	0
707	1	1	0	0	0	8481	0	191	360	0	1
708	1	0	2	0	0	4049	0	112	239	0	1
709	1	0	0	0	0	3020	0	63	398	0	2

710 rows × 11 columns

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

Out[ ]:
```
0      1
1      0
2      1
3      1
4      1
      ..
705    0
706    0
707    0
708    0
709    0
Name: Loan_Status, Length: 710, dtype: int64
```

9.jpg | 4.jpg | 2.jpg | Project- Planning.docx | Project- Planning.pdf | Data Flow Diagra....docx | Show all
Type here to search 25°C ENG 20:53 15-11-2022

Building the Models

Descision tree

```
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

```
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

```
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

```
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))
```

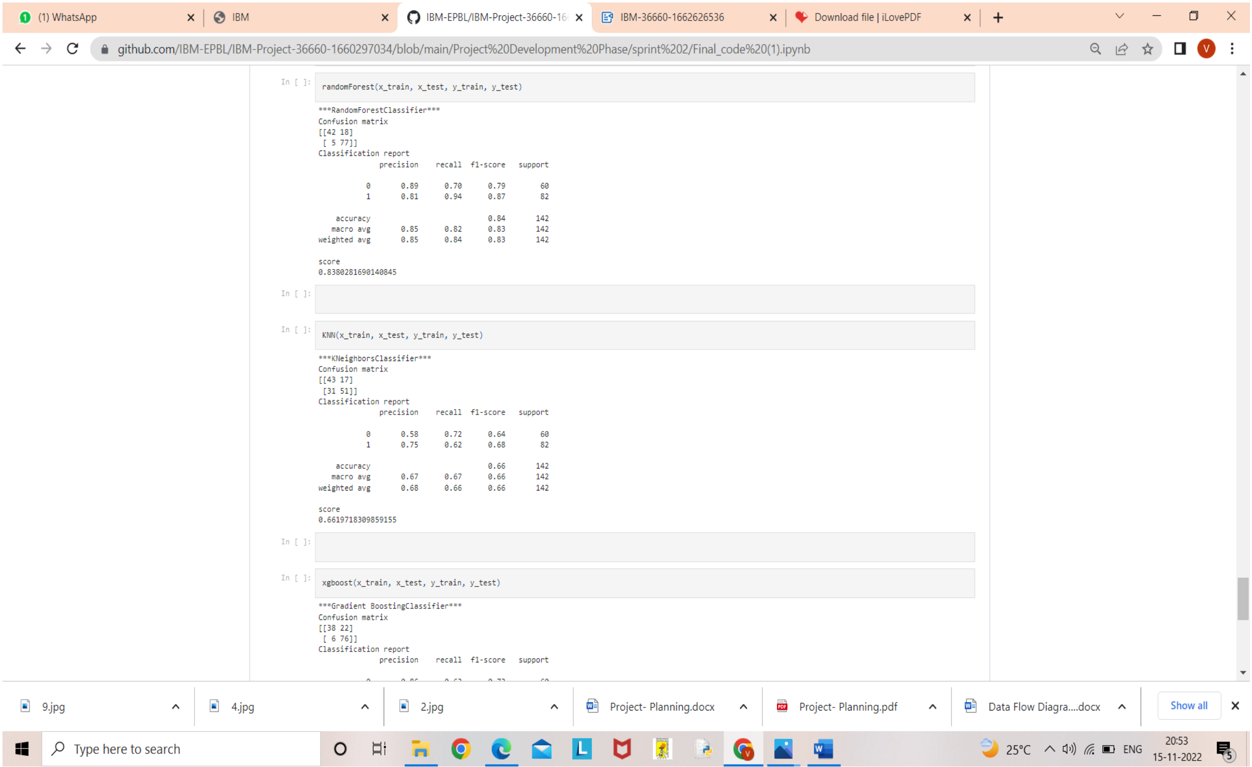
Comapring Models

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

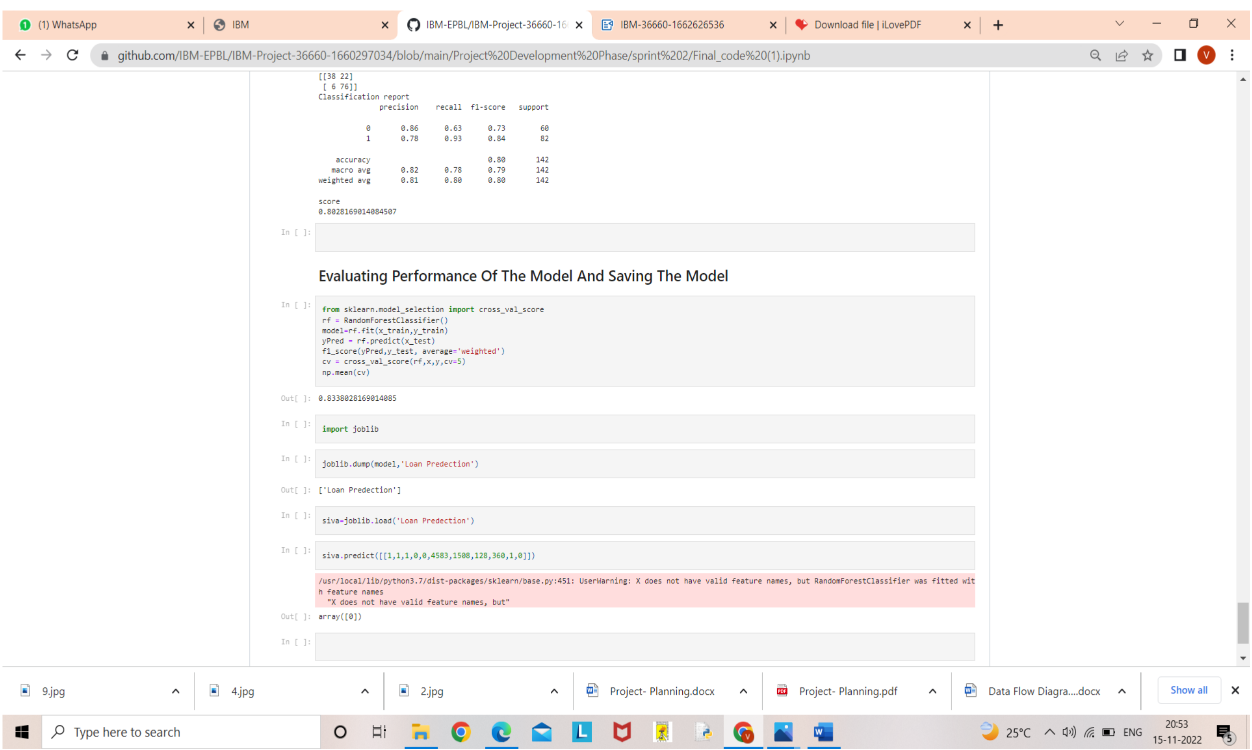
```
***DecisionTreeClassifier***
Confusion matrix
[[47 13]
 [19 63]]
Classification report
              precision    recall  f1-score   support

           0       0.71      0.78      0.75        60
           1       0.83      0.77      0.80        82

    accuracy                           0.77       142
   macro avg       0.77      0.78      0.77       142
weighted avg       0.78      0.77      0.78       142

score
0.7746478873239436
```

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

```
***RandomForestClassifier***
Confusion matrix
[[42 18]
 [ 5 77]]
Classification report
              precision    recall  f1-score   support
```



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

***RandomForestClassifier***
Confusion matrix
[[42 18]
 [ 5 77]]
Classification report
              precision    recall  f1-score   support

           0       0.89      0.70      0.79        60
           1       0.81      0.94      0.87        82

    accuracy                           0.84       142
   macro avg       0.85      0.82      0.83       142
weighted avg       0.85      0.84      0.83       142

score
0.8380281690140845

In [ ]:

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

***KNeighborsClassifier***
Confusion matrix
[[43 17]
 [31 51]]
Classification report
              precision    recall  f1-score   support

           0       0.58      0.72      0.64        60
           1       0.75      0.62      0.68        82

    accuracy                           0.66       142
   macro avg       0.67      0.67      0.66       142
weighted avg       0.68      0.66      0.66       142

score
0.6619718309859155

In [ ]:

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

***Gradient BoostingClassifier***
Confusion matrix
[[38 22]
 [ 6 76]]
Classification report
              precision    recall  f1-score   support
```

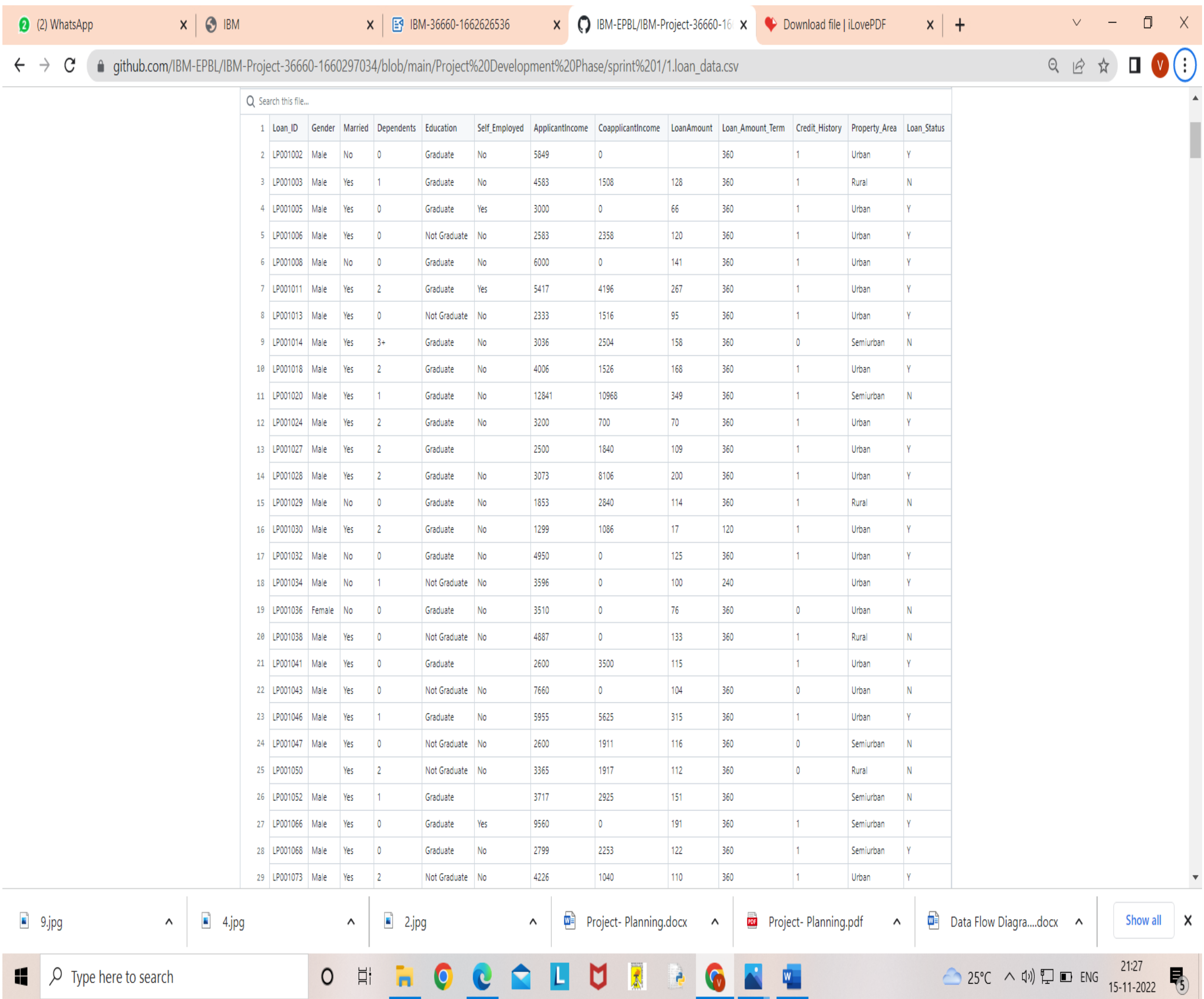


```
[[38 22]
 [ 6 76]]
Classification report
              precision    recall  f1-score   support

           0       0.86      0.63      0.73        60
           1       0.78      0.93      0.84        82

    accuracy                           0.80       142
   macro avg       0.82      0.78      0.79       142
weighted avg       0.81      0.80      0.80       142

score
0.8028169014084507

In [ ]:
```

Evaluating Performance Of The Model And Saving The Model

```
In [ ]: from sklearn.model_selection import cross_val_score
        rf = RandomForestClassifier()
        model=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[ ]: 0.8338028169014085

In [ ]: import joblib

In [ ]: joblib.dump(model,'Loan Predection')

Out[ ]: ['Loan Predection']

In [ ]: siva=joblib.load('Loan Predection')

In [ ]: siva.predict([[1,1,1,0,0,4583,1508,128,360,1,0]])

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted wit
h feature names
  "X does not have valid feature names, but"

Out[ ]: array([0])

In [ ]:
```

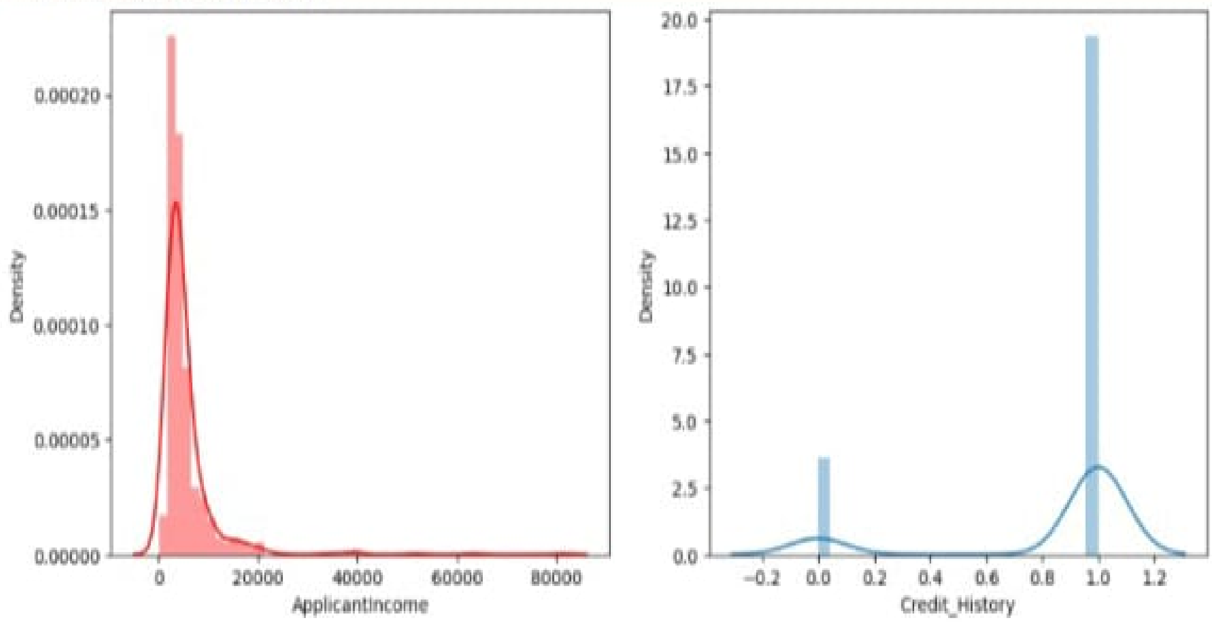
7.3 Database Schema

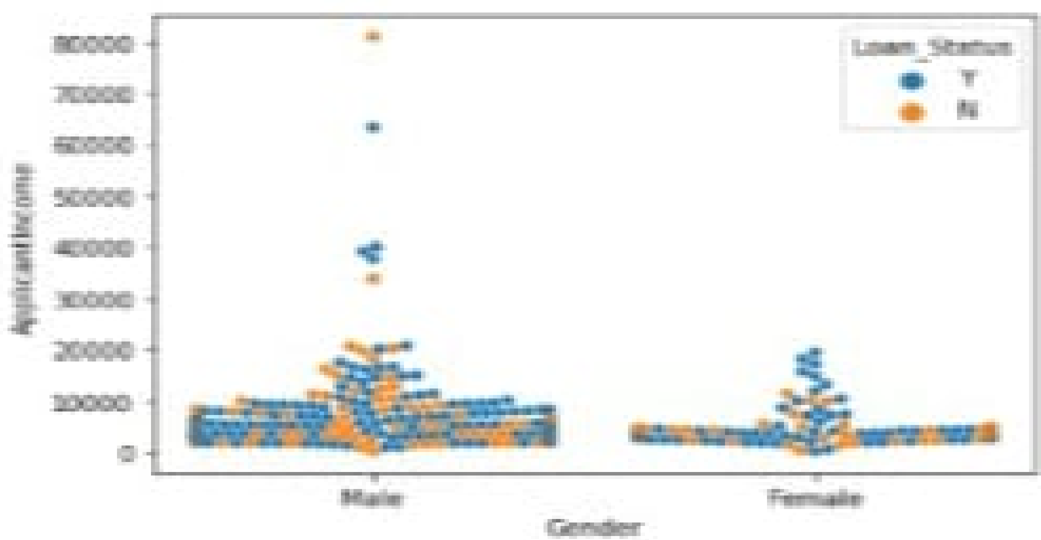


1	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
5	LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1	Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
8	LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360	1	Urban	Y
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
13	LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
18	LP001034	Male	No	1	Not Graduate	No	3596	0	100	240		Urban	Y
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
20	LP001038	Male	Yes	0	Not Graduate	No	4887	0	133	360	1	Rural	N
21	LP001041	Male	Yes	0	Graduate		2600	3500	115		1	Urban	Y
22	LP001043	Male	Yes	0	Not Graduate	No	7660	0	104	360	0	Urban	N
23	LP001046	Male	Yes	1	Graduate	No	5955	5625	315	360	1	Urban	Y
24	LP001047	Male	Yes	0	Not Graduate	No	2600	1911	116	360	0	Semiurban	N
25	LP001050		Yes	2	Not Graduate	No	3365	1917	112	360	0	Rural	N
26	LP001052	Male	Yes	1	Graduate		3717	2925	151	360		Semiurban	N
27	LP001066	Male	Yes	0	Graduate	Yes	9560	0	191	360	1	Semiurban	Y
28	LP001068	Male	Yes	0	Graduate	No	2799	2253	122	360	1	Semiurban	Y
29	LP001073	Male	Yes	2	Not Graduate	No	4226	1040	110	360	1	Urban	Y

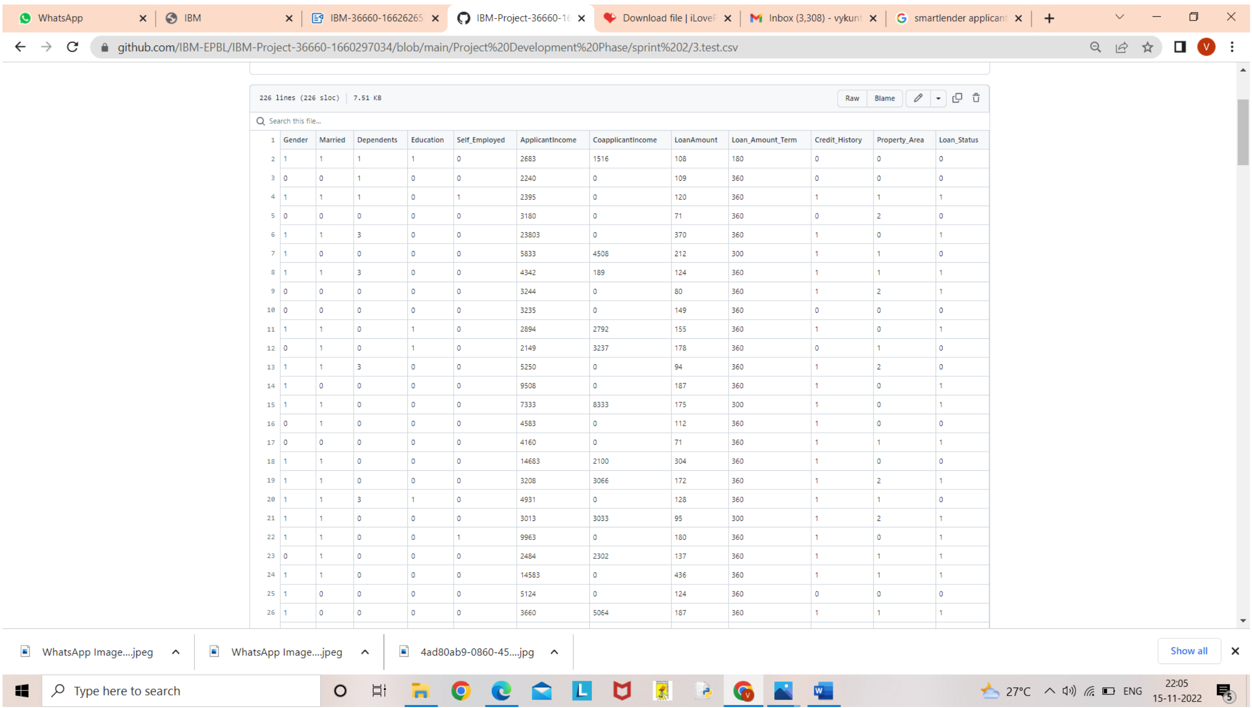
CHAPTER-8
TESTING

8.1 Test Cases

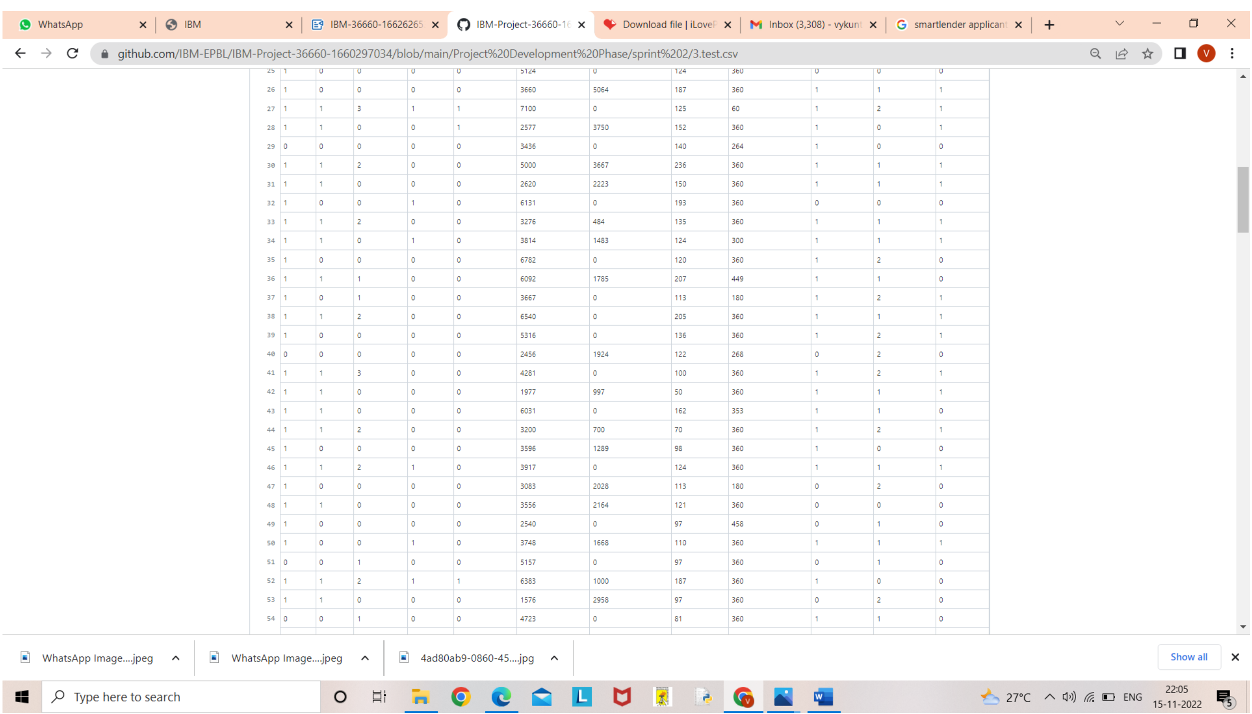




8.2 User Acceptance Testing



1	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	1	1	1	1	0	2683	1516	108	180	0	0	0
3	0	0	1	0	0	2240	0	109	360	0	0	0
4	1	1	1	0	1	2395	0	120	360	1	1	1
5	0	0	0	0	0	3180	0	71	360	0	2	0
6	1	1	3	0	0	23803	0	370	360	1	0	1
7	1	0	0	0	0	5833	4508	212	300	1	1	0
8	1	1	3	0	0	4342	189	124	360	1	1	1
9	0	0	0	0	0	3244	0	80	360	1	2	1
10	0	0	0	0	0	3235	0	149	360	0	0	0
11	1	1	0	1	0	2894	2792	155	360	1	0	1
12	0	1	0	1	0	2149	3237	178	360	0	1	0
13	1	1	3	0	0	5250	0	94	360	1	2	0
14	1	0	0	0	0	9508	0	187	360	1	0	1
15	1	1	0	0	0	7333	8333	175	300	1	0	1
16	0	1	0	0	0	4583	0	112	360	1	0	0
17	0	0	0	0	0	4160	0	71	360	1	1	1
18	1	1	0	0	0	14683	2100	304	360	1	0	0
19	1	1	0	0	0	3208	3066	172	360	1	2	1
20	1	1	3	1	0	4931	0	128	360	1	1	0
21	1	1	0	0	0	3013	3033	95	300	1	2	1
22	1	1	0	0	1	9963	0	180	360	1	0	1
23	0	1	0	0	0	2484	2302	137	360	1	1	1
24	1	1	0	0	0	14583	0	436	360	1	1	1
25	1	0	0	0	0	5124	0	124	360	0	0	0
26	1	0	0	0	0	3660	5064	187	360	1	1	1



26	1	0	0	0	0	3660	5064	187	360	1	1	1
27	1	1	3	1	1	7100	0	125	60	1	2	1
28	1	1	0	0	1	2577	3750	152	360	1	0	1
29	0	0	0	0	0	3436	0	140	264	1	0	0
30	1	1	2	0	0	5000	3667	236	360	1	1	1
31	1	1	0	0	0	2620	2223	150	360	1	1	1
32	1	0	0	1	0	6131	0	193	360	0	0	0
33	1	1	2	0	0	3276	484	135	360	1	1	1
34	1	1	0	1	0	3814	1483	124	300	1	1	1
35	1	0	0	0	0	6782	0	120	360	1	2	0
36	1	1	1	0	0	6092	1785	207	449	1	1	0
37	1	0	1	0	0	3667	0	113	180	1	2	1
38	1	1	2	0	0	6540	0	205	360	1	1	1
39	1	0	0	0	0	5316	0	136	360	1	2	1
40	0	0	0	0	0	2456	1924	122	268	0	2	0
41	1	1	3	0	0	4281	0	100	360	1	2	1
42	1	1	0	0	0	1977	997	50	360	1	1	1
43	1	1	0	0	0	6031	0	162	353	1	1	0
44	1	1	2	0	0	3200	700	70	360	1	2	1
45	1	0	0	0	0	3596	1289	98	360	1	0	0
46	1	1	2	1	0	3917	0	124	360	1	1	1
47	1	0	0	0	0	3083	2028	113	180	0	2	0
48	1	1	0	0	0	3556	2164	121	360	0	0	0
49	1	0	0	0	0	2540	0	97	458	0	1	0
50	1	0	0	1	0	3748	1668	110	360	1	1	1
51	0	0	1	0	0	5157	0	97	360	0	1	0
52	1	1	2	1	1	6383	1000	187	360	1	0	0
53	1	1	0	0	0	1576	2958	97	360	0	2	0
54	0	0	1	0	0	4723	0	81	360	1	1	0

CHAPTER-9
RESULTS

We have successfully compared different machine learning algorithms for the Property Loan dataset; theyare Random Forest,Naive Bayes, LogisticRegression and K Nearest Neighbors. The Logistic Regression algorithm gave the bestaccuracy (88.70%).

Table -1: Comparison of Algorithms

Sr.No.	Algorithm	Accuracy
1.	**Random Forest**	**79.03%**
2.	**Naive Bayes**	**85.48%**
3.	**Decision Tree**	**79.03%**
4.	**Logistic Regression**	**88.70%**
5.	**K NearestNeighbor**	**80.64%**

Implementation Output

First, we have our home page where we get information about our system, details of the developers of the system and also a button to go to the prediction page.

The next is the prediction page where the user can fill the form to check whether he/she is eligible for loan approval or not. It also includes comparison of different algorithms in terms of accuracyin graphical representation.

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

The system is trained on old training dataset in future software can be made such that new testing data should also take part in training data after some fix time.

CHAPTER-13
APPENDIX

Source Code

Importing the Libraries
In [2]:
```
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 [3]:
```
data=pd.read_csv("loan_data.csv")
data
```
Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanA nt
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0

3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0
...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	

614 rows × 13 columns

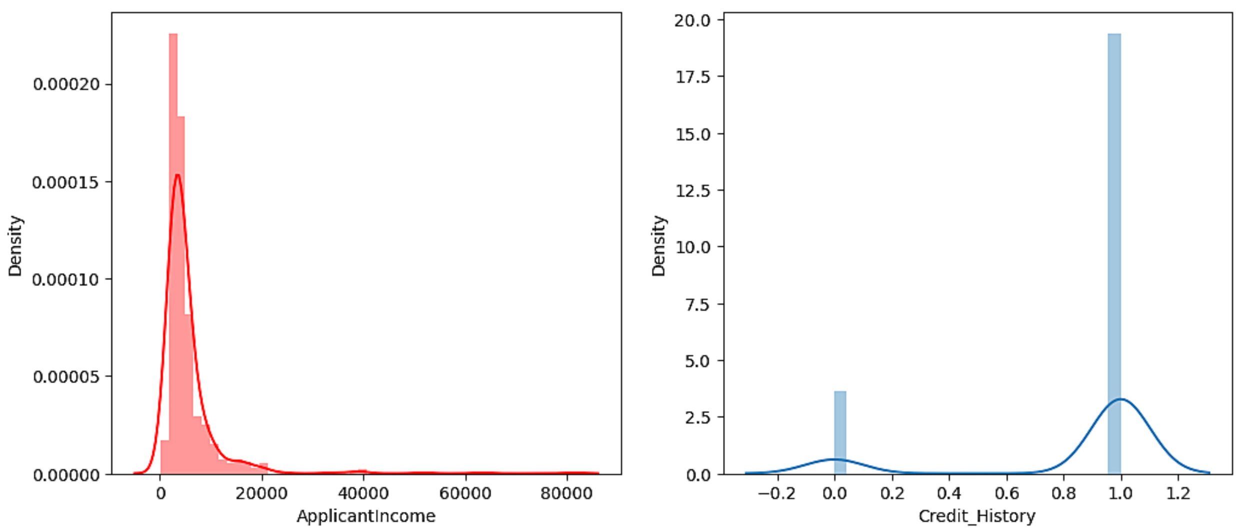
Visualizations

In [7]:

```
#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()
```

```
C:\Users\72081\AppData\Local\Temp\ipykernel_14816\3941809966.py:4:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level function
with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
  sns.distplot(data['ApplicantIncome'], color='r')
C:\Users\72081\AppData\Local\Temp\ipykernel_14816\3941809966.py:6:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level function
with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['Credit_History'])
```

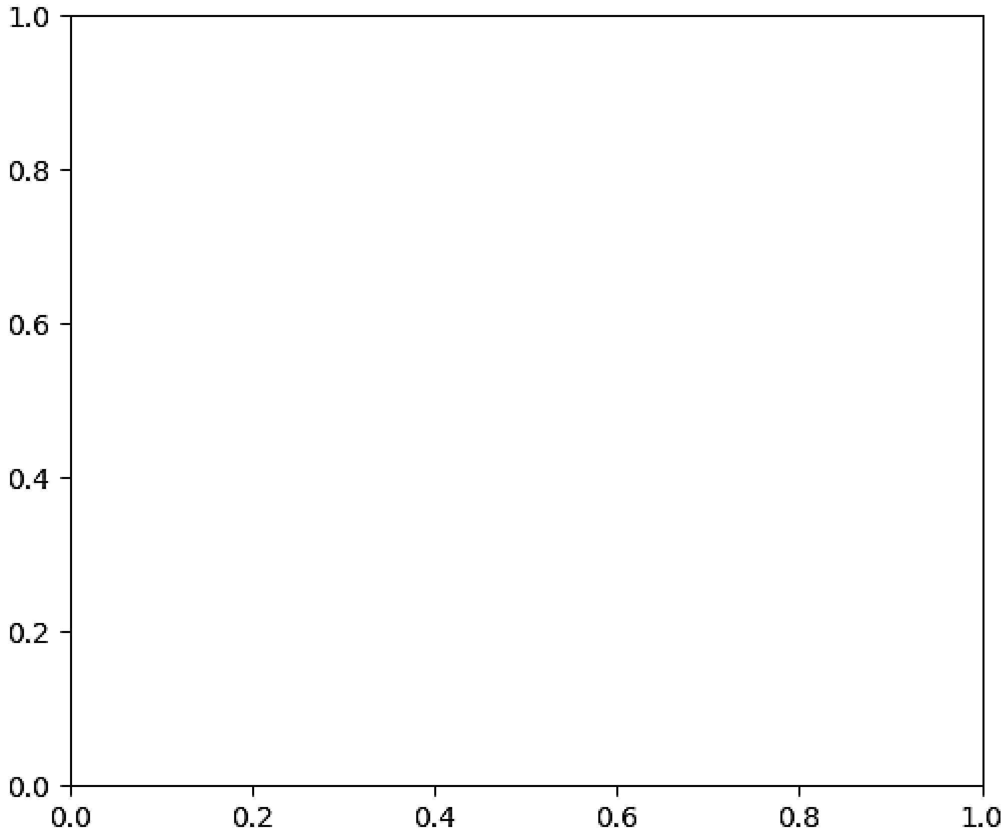


In [6]:

```
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'])
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call
last)
Cell In [6], line 3
```

```
      1 plt.figure(figsize=(20,5))
      2 plt.subplot(131)
----> 3 sns.countplot(data['Married'], hue=data['Gender'])
      4 plt.subplot(132)
      5 sns.countplot(data['Self_Employed'], hue=data['Education'])

File C:\Python310\lib\site-packages\seaborn\categorical.py:2942, in
countplot(data, x, y, hue, order, hue_order, orient, color, palette,
saturation, width, dodge, ax, **kwargs)
   2939 elif x is not None and y is not None:
   2940     raise ValueError("Cannot pass values for both `x` and `y`")
-> 2942 plotter = _CountPlotter(
   2943     x, y, hue, data, order, hue_order,
   2944     estimator, errorbar, n_boot, units, seed,
   2945     orient, color, palette, saturation,
   2946     width, errcolor, errwidth, capsize, dodge
   2947 )
   2949 plotter.value_label = "count"
   2951 if ax is None:

File C:\Python310\lib\site-packages\seaborn\categorical.py:1530, in
_BarPlotter.__init__(self, x, y, hue, data, order, hue_order, estimator,
errorbar, n_boot, units, seed, orient, color, palette, saturation, width,
errcolor, errwidth, capsize, dodge)
   1525 def __init__(self, x, y, hue, data, order, hue_order,
   1526              estimator, errorbar, n_boot, units, seed,
   1527              orient, color, palette, saturation, width,
   1528              errcolor, errwidth, capsize, dodge):
   1529     """Initialize the plotter."""
-> 1530     self.establish_variables(x, y, hue, data, orient,
   1531                              order, hue_order, units)
   1532     self.establish_colors(color, palette, saturation)
   1533     self.estimate_statistic(estimator, errorbar, n_boot, seed)

File C:\Python310\lib\site-packages\seaborn\categorical.py:437, in
_CategoricalPlotter.establish_variables(self, x, y, hue, data, orient,
order, hue_order, units)
    435 if hue is not None:
    436     error = "Cannot use `hue` without `x` and `y`"
--> 437     raise ValueError(error)
    439 # No hue grouping with wide inputs
    440 plot_hues = None
```

```
ValueError: Cannot use `hue` without `x` and `y`
```

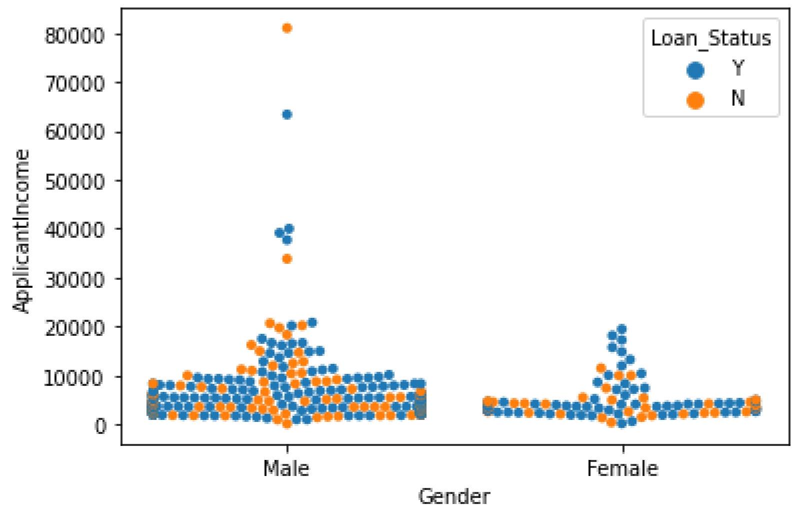


In [ ]:

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296:
UserWarning: 67.5% of the points cannot be placed; you may want to
decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296:
UserWarning: 33.0% of the points cannot be placed; you may want to
decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[ ]:



Data Pre-processing

In [ ]:

```
data.describe()
```

Out[ ]:

	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 [ ]:

In [ ]:

```
data.info()
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 [ ]:

```
data.isnull().sum()
```

Out[ ]:

```
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 [ ]:

```
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 [ ]:

```
data.isnull().sum()
```

Out[ ]:

```
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 [ ]:

```
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 [ ]:

```
data
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loar
0	LP001002	1	0	0	0	0	5849	0.0	
1	LP001003	1	1	1	0	0	4583	1508.0	
2	LP0010	1	1	0	0	1	3000	0.0	

	05							
3	LP001006	1	1	0	1	0	2583	2358.0
4	LP001008	1	0	0	0	0	6000	0.0
...	...	...	...	...	...	...	...	...
609	LP002978	0	0	0	0	0	2900	0.0
610	LP002979	1	1	3	0	0	4106	0.0
611	LP002983	1	1	1	0	0	8072	240.0
612	LP002984	1	1	2	0	0	7583	0.0
613	LP002990	0	0	0	0	1	4583	0.0

614 rows × 13 columns

In [ ]:

```
#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 [ ]:

```
#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    366
0    344
Name: Loan_Status, dtype: int64
/usr/local/lib/python3.7/dist-packages/imblearn/utils/_validation.py:591: FutureWarning: Pass sampling_strategy=0.95 as keyword args. From version 0.9 passing these as positional arguments will result in an error
  FutureWarning,
```

Scaling the Data

In [ ]:

```
sc=MaxAbsScaler()
x_bal_scaled=sc.fit_transform(x_bal)
x_bal_scaled = pd.DataFrame(x_bal,columns=x.columns)
```

In [ ]:

```
x_bal_scaled
```

Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	L
0	1	0	0	0	0	5849	0	120	
1	1	1	1	0	0	4583	1508	128	
2	1	1	0	0	1	3000	0	66	
3	1	1	0	1	0	2583	2358	120	
4	1	0	0	0	0	6000	0	141	
...	...	...	...	...	...	...	...	...	
705	1	0	0	0	0	14263	0	222	
706	0	0	1	0	0	4714	0	88	
707	1	1	0	0	0	8481	0	191	
708	1	0	2	0	0	4049	0	112	
709	1	0	0	0	0	3020	0	63	

710 rows × 11 columns

Processed Data

In [ ]:

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

In [ ]:

```
final_df
```

Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	L
0	1	0	0	0	0	5849	0	120	
1	1	1	1	0	0	4583	1508	128	
2	1	1	0	0	1	3000	0	66	
3	1	1	0	1	0	2583	2358	120	
4	1	0	0	0	0	6000	0	141	
...	...	...	...	...	...	...	...	...	
705	1	0	0	0	0	14263	0	222	
706	0	0	1	0	0	4714	0	88	
707	1	1	0	0	0	8481	0	191	
708	1	0	2	0	0	4049	0	112	
709	1	0	0	0	0	3020	0	63	

710 rows × 12 columns

Saving into train test datasets

In [ ]:

```
train,test = train_test_split(final_df, test_size=0.33, random_state=42)
```

In [ ]:

```
train.to_csv('train.csv',encoding='utf-8',index=False)
test.to_csv('test.csv',encoding='utf-8',index=False)
```

Splitting the data

In [ ]:

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

In [ ]:

```
x
```

Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	L
0	1	0	0	0	0	5849	0	120	
1	1	1	1	0	0	4583	1508	128	
2	1	1	0	0	1	3000	0	66	

3	1	1	0	1	0	2583	2358	120
4	1	0	0	0	0	6000	0	141
...	...	...	...	...	...	...	...	...
705	1	0	0	0	0	14263	0	222
706	0	0	1	0	0	4714	0	88
707	1	1	0	0	0	8481	0	191
708	1	0	2	0	0	4049	0	112
709	1	0	0	0	0	3020	0	63

710 rows × 11 columns

In [ ]:

```
y=final_df.Loan_Status
y
```

Out[ ]:

```
0      1
1      0
2      1
3      1
4      1
      ..
705    0
706    0
707    0
708    0
709    0
Name: Loan_Status, Length: 710, dtype: int64
```

In [ ]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=42)
```

Building the Models

Descision tree

In [ ]:

```
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 [ ]:

```
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 [ ]:

```
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 [ ]:

```
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))
```

In [ ]:

Comapring Models

In [ ]:

```
decisionTree(x_train, x_test, y_train, y_test)
***DecisionTreeClassifier***
Confusion matrix
[[47 13]
 [19 63]]
Classification report
              precision    recall  f1-score   support

           0       0.71      0.78      0.75        60
           1       0.83      0.77      0.80        82

    accuracy                           0.77       142
   macro avg       0.77      0.78      0.77       142
weighted avg       0.78      0.77      0.78       142

score
0.7746478873239436
```

In [ ]:

In [ ]:

```
randomForest(x_train, x_test, y_train, y_test)
***RandomForestClassifier***
Confusion matrix
[[42 18]
 [ 5 77]]
Classification report
              precision    recall  f1-score   support

           0       0.89      0.70      0.79        60
           1       0.81      0.94      0.87        82

    accuracy                           0.84       142
   macro avg       0.85      0.82      0.83       142
weighted avg       0.85      0.84      0.83       142
```

```
score
0.8380281690140845
```

In [ ]:

In [ ]:

```
KNN(x_train, x_test, y_train, y_test)
***KNeighborsClassifier***
Confusion matrix
[[43 17]
 [31 51]]
Classification report
              precision    recall  f1-score   support

           0       0.58      0.72      0.64        60
           1       0.75      0.62      0.68        82

    accuracy                           0.66       142
   macro avg       0.67      0.67      0.66       142
weighted avg       0.68      0.66      0.66       142

score
0.6619718309859155
```

In [ ]:

In [ ]:

```
xgboost(x_train, x_test, y_train, y_test)
***Gradient BoostingClassifier***
Confusion matrix
[[38 22]
 [ 6 76]]
Classification report
              precision    recall  f1-score   support

           0       0.86      0.63      0.73        60
           1       0.78      0.93      0.84        82

    accuracy                           0.80       142
   macro avg       0.82      0.78      0.79       142
weighted avg       0.81      0.80      0.80       142

score
0.8028169014084507
```

In [ ]:

Evaluating Performance Of The Model And Saving The Model

In [ ]:

```
from sklearn.model_selection import cross_val_score
rf = RandomForestClassifier()
model=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[ ]:

```
0.8338028169014085
```

In [ ]:

```
import joblib
```

In [ ]:

```
joblib.dump(model,'Loan Predection')
```

Out[ ]:

```
['Loan Predection']
```

In [ ]:

```
siva=joblib.load('Loan Predection')
```

In [ ]:

```
siva.predict([[1,1,1,0,0,4583,1508,128,360,1,0]])
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted
with feature names
  "X does not have valid feature names, but"
```

Out[ ]:

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

In [ ]:

GITHUB LINK :

https://github.com/IBM-EPBL/IBM-Project-36660-1660297034

PROJECT DEMO LINK :

https://www.youtube.com/embed/5JRloNdTRr0