PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING

DONE BY

RAJALAKSHMI G

SANGAVI S

SANJAI

ROOBINI

Overview :

INTRODUCTION:

One of the most important factors which affect our country's economy and financial condition is the credit system governed by the banks. The process of bank credit risk evaluation is recognized at banks across the globe. "As we know credit risk evaluation isvery crucial, there is a variety of techniques are used for risk level calculation. In addition, credit risk is one of the main functions of the banking community.

The prediction of credit defaulters is one of the difficult tasks for any bank. But by forecasting the loan defaulters, the banks definitely may reduce their loss by reducing their non-profit assets, so that recovery of approved loans can take place without any loss and it can play as the contributing parameter of the bank statement. This makes the study of this loan approval prediction important. Machine Learning techniques are very crucial and useful in the prediction of these types of data.

We will be using classification algorithms such as Decision tree, Random forest, KNN, ANNand xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing flask integration and IBMdeployment.

PURPOSE:

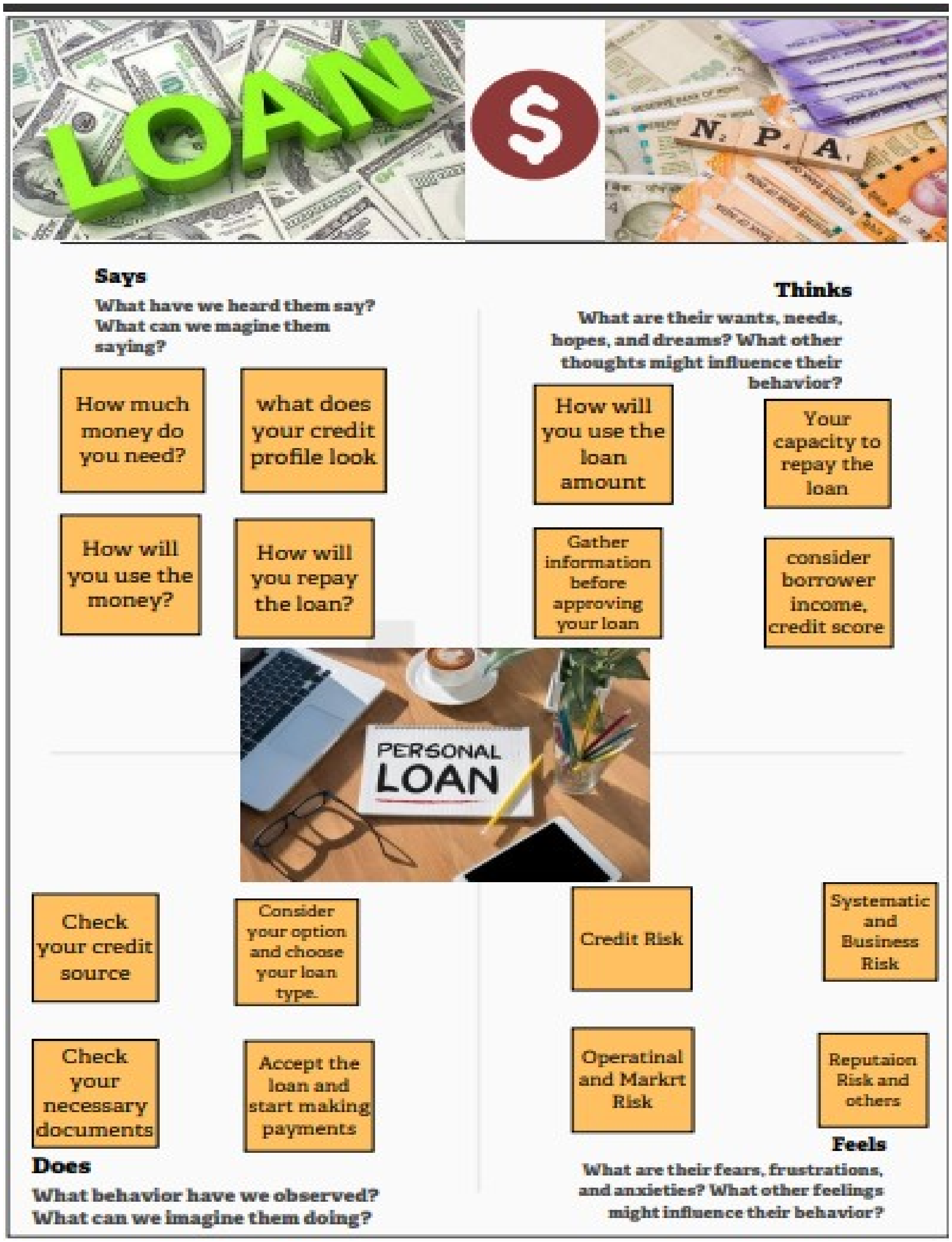
A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest. Thereare various types of loans available to individuals and businesses, such as personal loans, mortgages, auto loans, student loans, business loans and many more. They are offered by banks, credit unions,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

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

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

PROBLEM DEFINITION & DESING THINKING :

1) Empathy Map:



Says

What have we heard them say? What can we magine them saying?

How much money do you need?

what does your credit profile look

How will you use the money?

How will you repay the loan?

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

How will you use the loan amount

Your capacity to repay the loan

Gather information before approving your loan

consider borrower income, credit score

Check your credit source

Consider your option and choose your loan type.

Check your necessary documents

Accept the loan and start making payments

Does

What behavior have we observed? What can we imagine them doing?

Credit Risk

Systematic and Business Risk

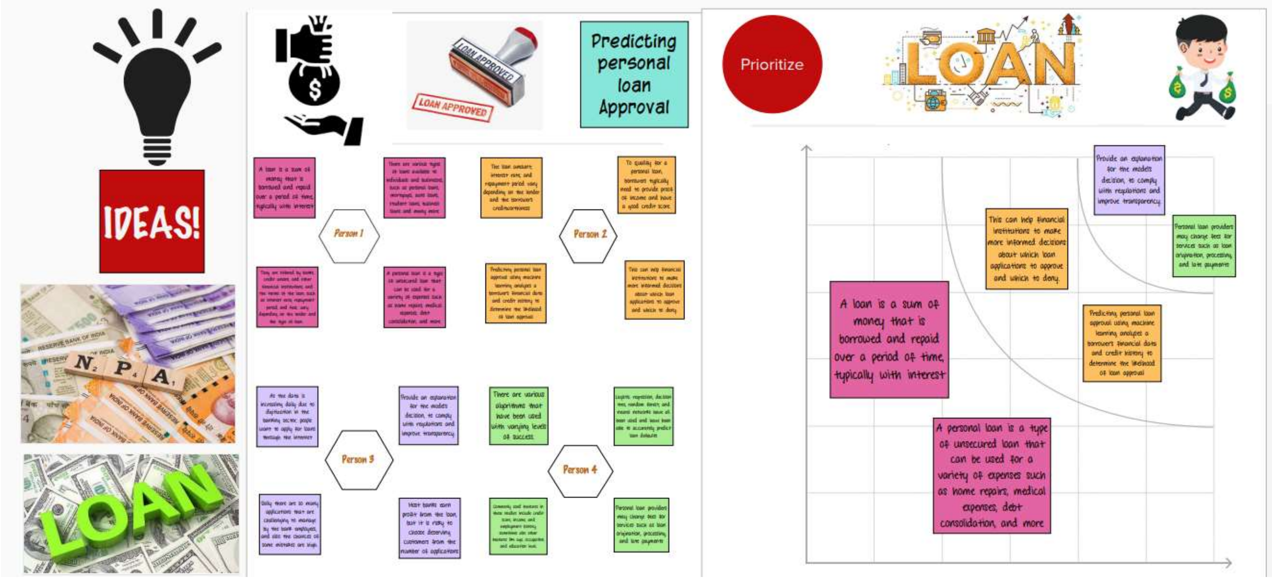
Operatinal and Markrt Risk

Reputaion Risk and others

Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

2)Ideation & Brainstorming Map



A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest

There are various types of loans available to individuals and businesses, such as personal loans, mortgages, auto loans, student loans, business loans and many more

They are offered by banks, credit unions and other financial institutions and the terms of the loan such as interest rate, repayment period and fees vary depending on the lender and the type of loan

A personal loan is a type of unsecured loan that can be used for a variety of expenses such as home repairs, medical expenses, debt consolidation, and more

The loan amount, interest rate, and repayment period vary depending on the lender and the borrowers creditworthiness

To qualify for a personal loan, borrowers typically need to provide proof of income and have a good credit score

Predicting personal loan approval using machine learning analyzes a borrowers financial data and credit history to determine the likelihood of loan approval

This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny

As the data is increasing daily due to digitization in the banking sector, people want to apply for loans through the internet

Provide an explanation for the models decision, to comply with regulations and improve transparency

Daily there are so many applications that are challenging to manage by the bank employees, and also the chances of some mistakes are high

Most banks earn profit from the loan, but it is risky to choose deserving customers from the number of applications

There are various algorithms that have been used with varying levels of success

Logistic regression, decision trees, random forests, and neural networks have all been used and have been able to accurately predict loan defaults

Commonly used features in these models include credit score, income and employment history, sometimes also other features like age, occupation and education level.

Personal loan providers may charge fees for services such as loan origination, processing, and late payments

Provide an explanation for the models decision, to comply with regulations and improve transparency

This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

Personal loan providers may charge fees for services such as loan origination, processing, and late payments

A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest

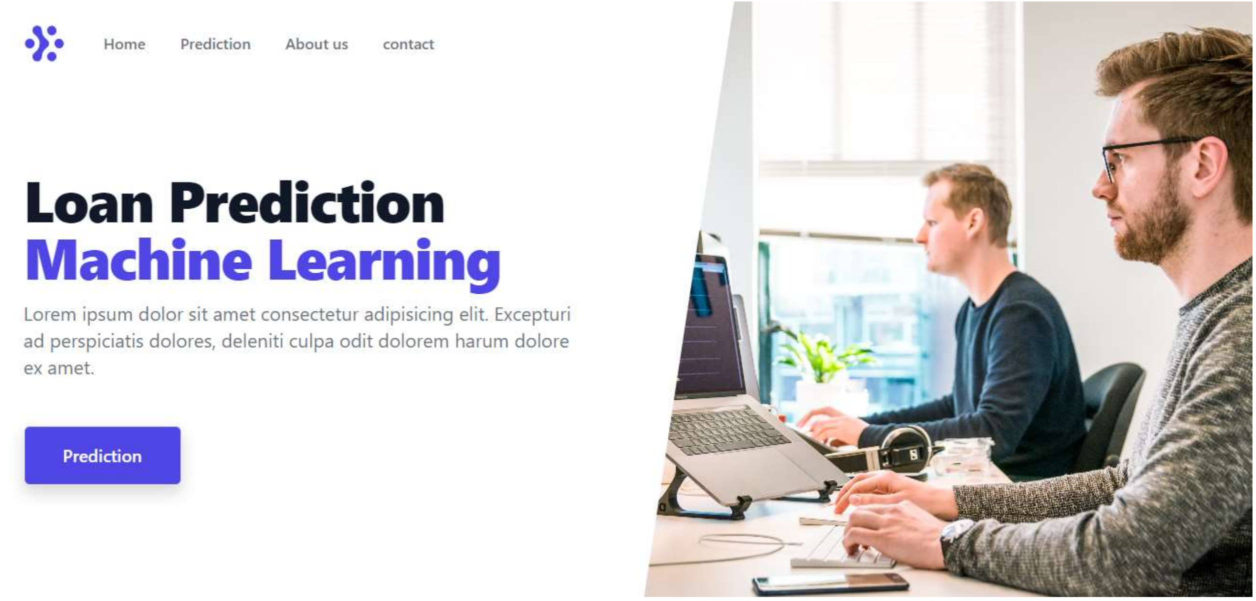
Predicting personal loan approval using machine learning analyzes a borrowers financial data and credit history to determine the likelihood of loan approval

A personal loan is a type of unsecured loan that can be used for a variety of expenses such as home repairs, medical expenses, debt consolidation, and more

RESULT:

Final Findings of the project:

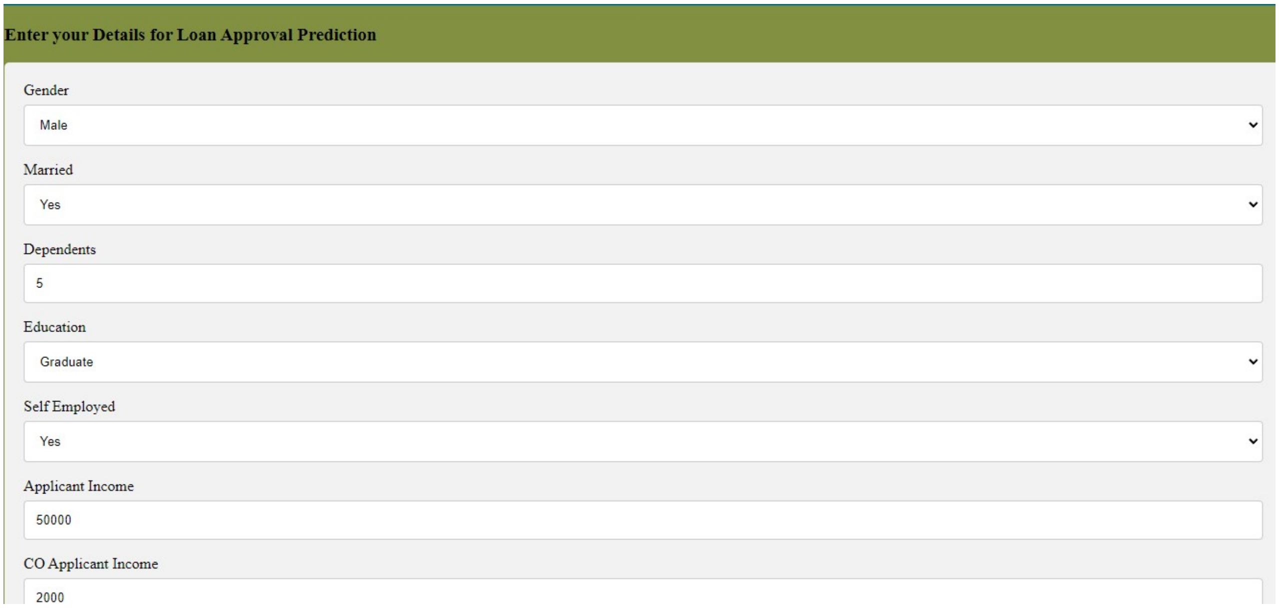
Home page of the project:



Loan Prediction
Machine Learning

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

Filling the Form for Loan Prediction :



Enter your Details for Loan Approval Prediction

Gender

Male

Married

Yes

Dependents

5

Education

Graduate

Self Employed

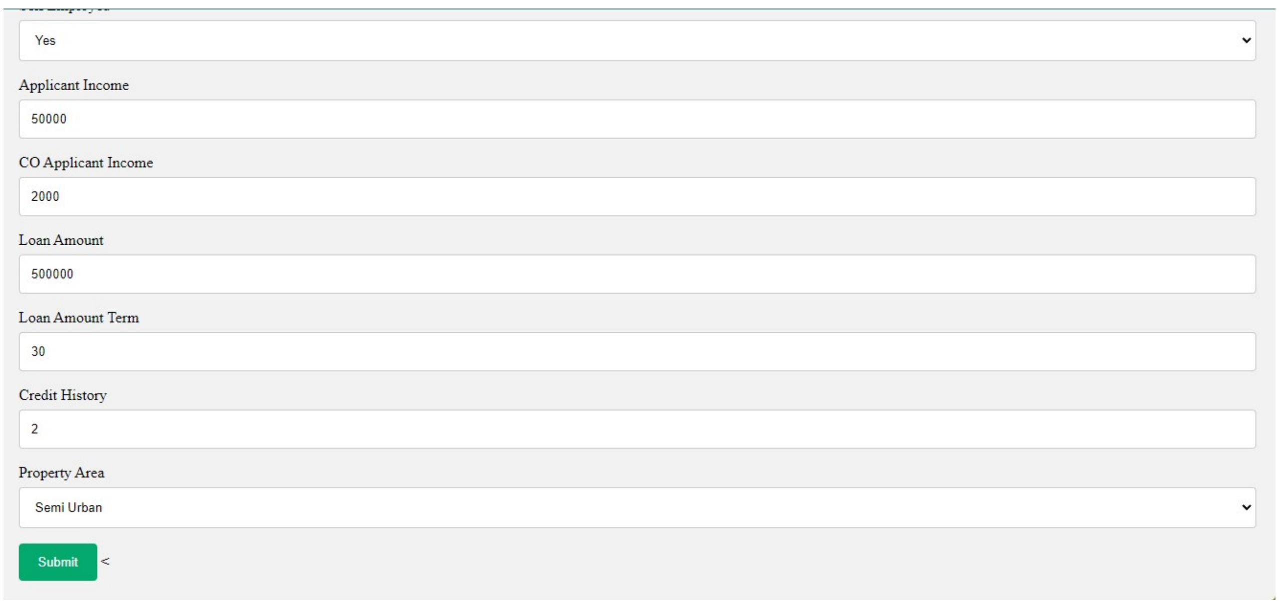
Yes

Applicant Income

50000

CO Applicant Income

2000



Yes

Applicant Income

50000

CO Applicant Income

2000

Loan Amount

500000

Loan Amount Term

30

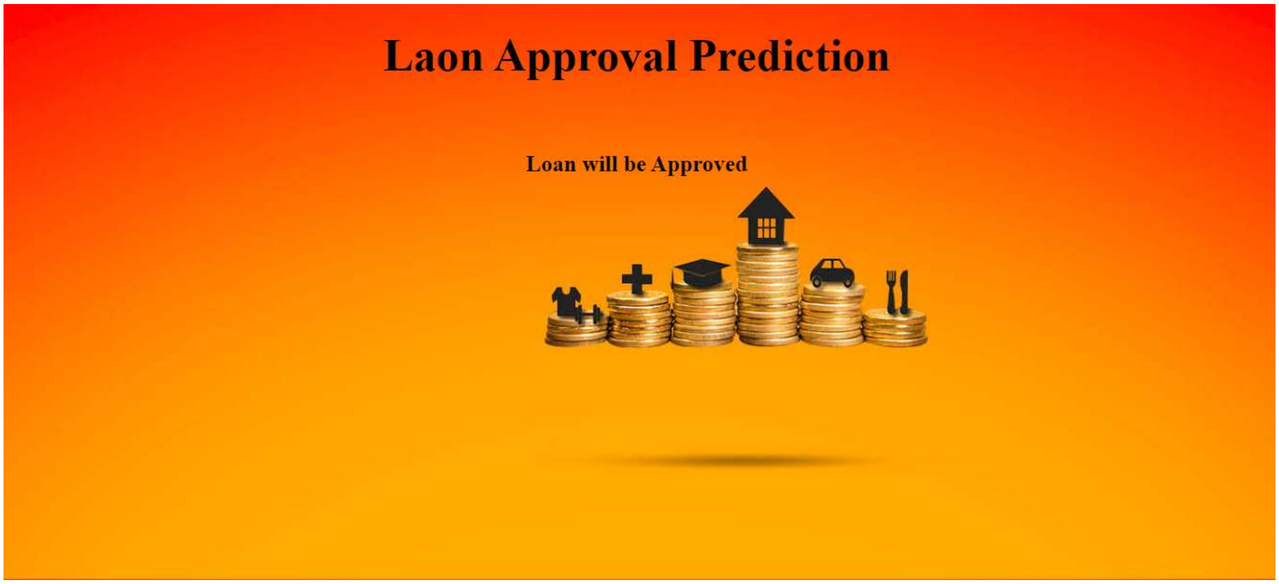
Credit History

2

Property Area

Semi Urban

Output page:



Laon Approval Prediction

Loan will be Approved

ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

Flexibility: A bank loan allows one to repay as per convenience as long as the instalments are regular and timely. Unlike an overdraft where all the credit is deducted in go, or a consumer credit card where the maximum limit cannot be utilised in one go.

Cost Effectiveness: When it comes to interest rates, bank loans are usually the cheapest option compared to overdraft and credit card.

- Profit Retention: When you raise funds through equity you have to share profits with shareholders. However, in a bank loan raised finance you do not have to share profits with the bank.

- Benefit of Tax: Government makes the interest payable on the loan a tax-deductible item when the loan has been taken for business purpose.

DISADVANTAGES:

Hard Prerequisite: Since big finance from a bank is based on collateral, most young businesses will find it hard to finance the operations based on bank loan.

- Irregular Payment Amounts: Over a long duration payback via monthly instalment might witness variation in the rate of interest. This means that the EMI will not be constant, rather it will change as per the influence of the market on the interest applicable.

APPLICATIONS:

A Prediction Model uses data mining, statistics and probability to forecast an outcome. Every model has some variables known as predictors that are likely to influence future results. The data that was collected from various resources then a statistical model is made. It can use a simple linear equation or a sophisticated neural network mapped using a complex software. As more data becomes available the model becomes more refined and the error decreases meaning then it'll be able to predict with the least risk and consuming as less time as it can. The Prediction Model helps the banks by minimizing the risk associated with the loan approval system and helps the applicant by decreasing the time taken in the process. The main objective of the Project is to compare the Loan Prediction Models made implemented using various algorithms and choose the best one out of them that can shorten the loan approval time and decrease the risk associated with it. It is done by predicting if the loan can be given to that person on the basis of various parameters like credit score, income, age, marital status, gender, etc. The prediction model not only helps the applicant but also helps the bank by minimizing the risk and reducing the number of defaulters.

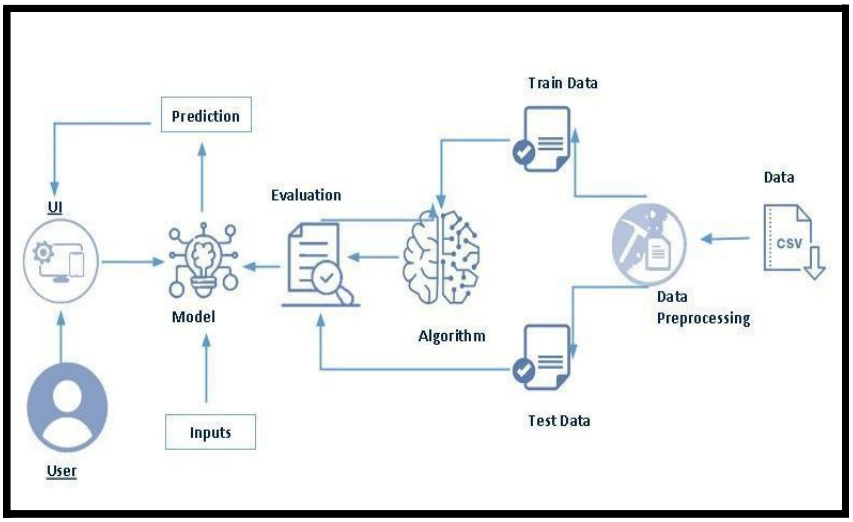
In the present scenario, a loan needs to be approved manually by a representative of the bank which means that person will be responsible for whether the person is eligible for the loan or not and also calculating the risk associated with it. As it is done by a human it is a time consuming process and is susceptible to errors. If the loan is not repaid, then it accounts as a loss to the bank and banks earn most of their profits by the interest paid to them. If the banks lose too much money, then it will result in a banking crisis. These banking crisis affects the economy of the country. So it is very important that the loan should be approved with the least amount of error in risk calculation while taking up as the least time possible. So a loan prediction model is required that can predict quickly whether the loan can be passed or not with the least amount of risk possible.

FUTURE SCOPE:

The two most pressing issues in the banking sector are: How risky is the borrower. Should we lend to the borrower given the risk? The response to the first question dictates the borrower's interest rate. Interest rate, among other things (such as time value of money), tests the riskiness of the borrower, i.e. the higher the interest rate, the riskier the borrower. We will then decide whether the applicant is suitable for the loan based on the interest rate. Lenders (investors) make loans to creditors in return for the guarantee of interest-bearing repayment. That is, the lender only makes a return (interest) if the borrower repays the loan. However, whether he or she does not repay the loan, the lender loses money. Banks make loans to customers in exchange for the guarantee of repayment. Some would default on their debts, unable to repay them for a number of reasons. The bank retains insurance to minimize the possibility of failure in the case of a default. The insured sum can cover the whole loan amount or just a portion of it. Banking processes use manual procedures to determine whether or not a borrower is suitable for a loan based on results. Manual procedures were mostly effective, but they were insufficient when there were a large number of loan applications. At that time, making a decision would take a long time. As a result, the loan prediction machine learning model can be used to assess a customer's loan status and build strategies. This model extracts and introduces the essential features of a borrower that influence the customer's loan status. Finally, it produces the planned performance (loan status). These reports make a bank manager's job simpler and quicker.

APPENDIX:

Technical Architecture:



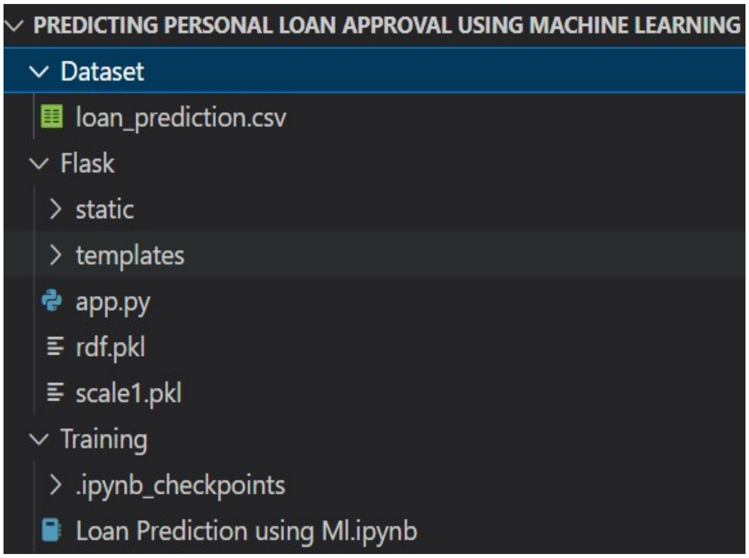
Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
  - Once model analyses the input the prediction is showcased on the UI .
  - To accomplish this, we have to complete all the activities listed below,
- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation

- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

Project Structure:

Create the Project folder which contains files as shown below



```
PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING
  Dataset
    loan_prediction.csv
  Flask
    static
    templates
    app.py
    rdf.pkl
    scale1.pkl
  Training
    .ipynb_checkpoints
    Loan Prediction using Ml.ipynb
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.

- rdf.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

Define Problem / Problem Understanding

Specify the business problem

Business requirements

The business requirements for a machine learning model to predict personal loan approval include the ability to accurately predict loan approval based on applicant information, Minimise the number of false positives (approved loans that default) and false negatives (rejected loans that would have been successful).Provide an explanation for the model's decision, to comply with regulations and improve transparency.

Social or Business Impact.

Social Impact:- Personal loans can stimulate economic growth by providing individuals with the funds they need to make major purchases, start businesses, or invest in their education.

Business Model/Impact:- Personal loan providers may charge fees for services such as loan origination, processing, and late payments.Advertising the brand awareness and marketing to reach out to potential borrowers to generate revenue.

Data Collection & Preparation:

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Collect the dataset:

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

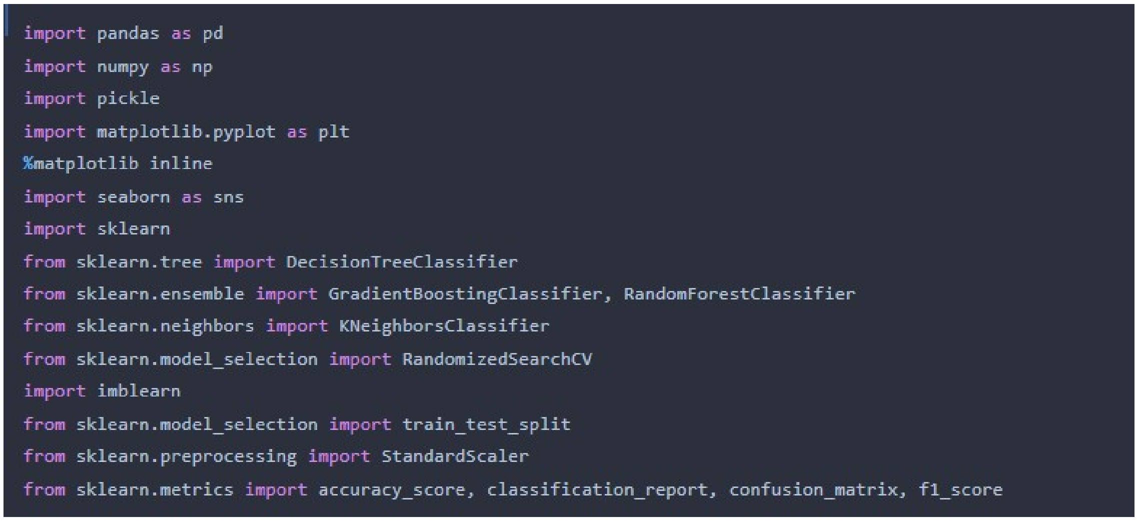
Link: https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Importing the libraries:

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

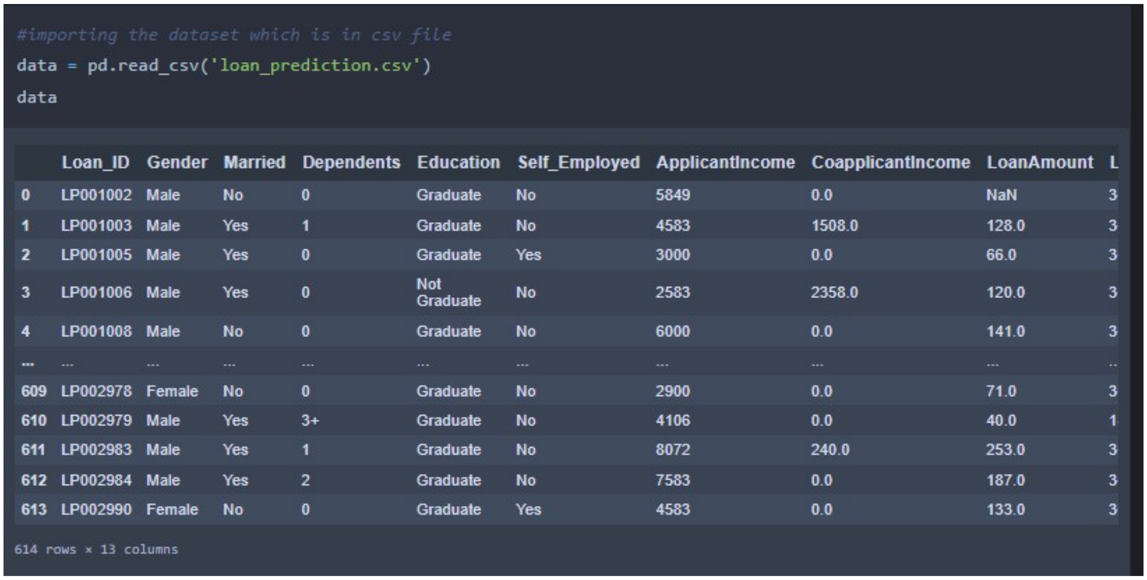


```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

Read the Dataset:

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have togive the directory of the csv file.



```python
#importing the dataset which is in csv file
data = pd.read_csv('loan_prediction.csv')
data
```

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

```
614 rows × 13 columns
```

Data Preparation:

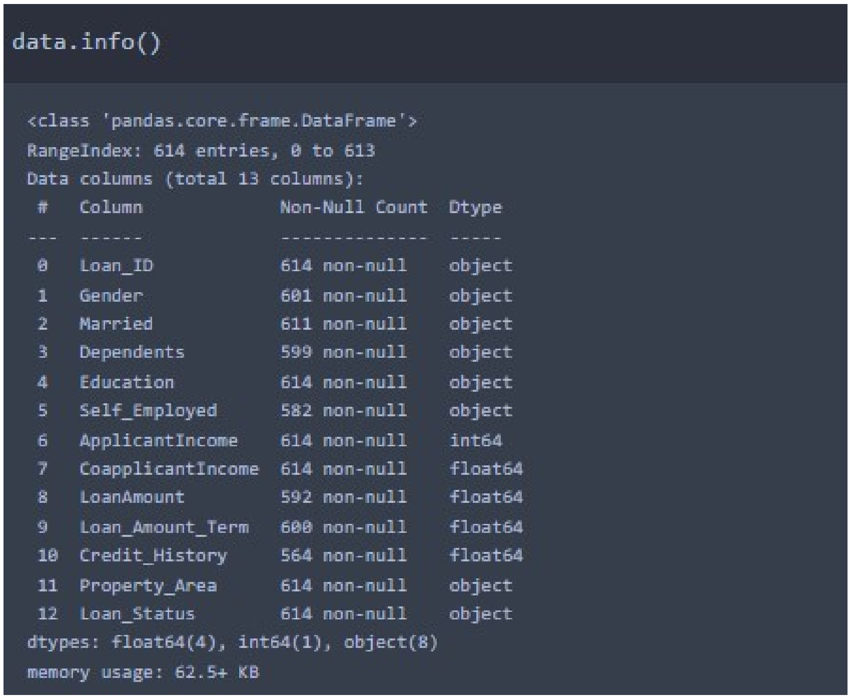
As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Imbalance Data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps. **Handling missing values:**

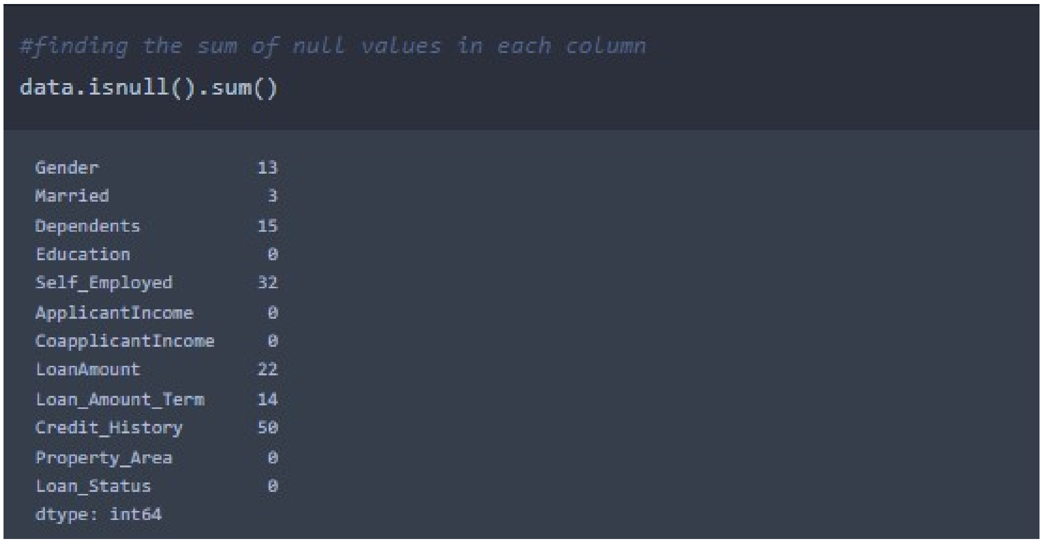
- Let’s find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.



```
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
```

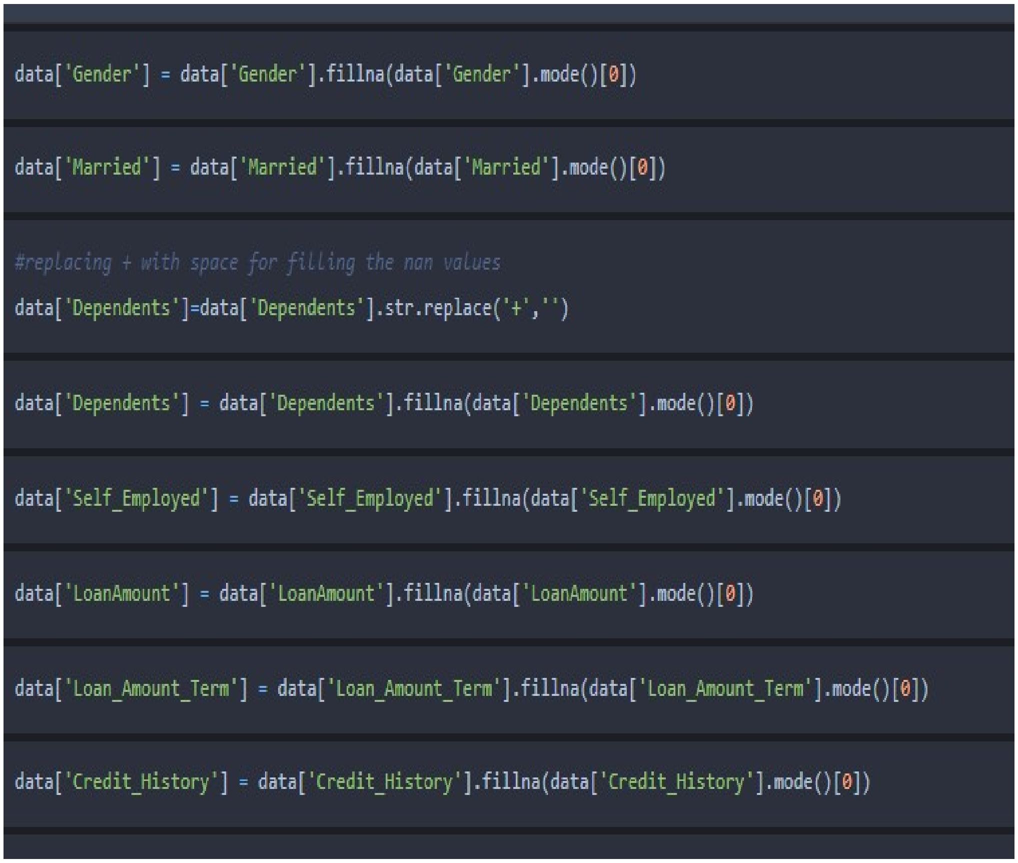
- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.



```
#finding the sum of null values in each column
data.isnull().sum()
```

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

- From the above code of analysis, we can infer that columns such as gender, married,dependents,self employed ,loan amount, loan amount term and credit history are having the missing values, we need to treat them in a required way.



```
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'].str.replace('+','')
```

```
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])
```

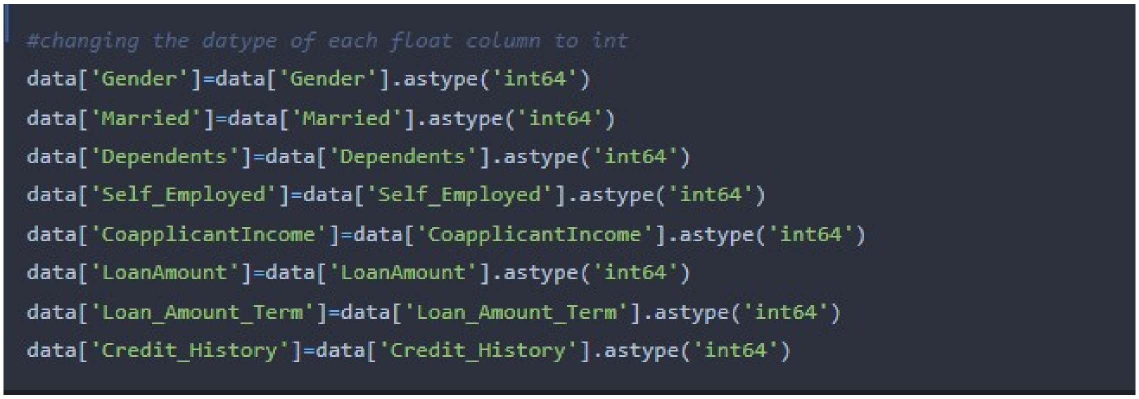
- We will fill in the missing values in the numeric data type using the mean value of that particular column and categorical data type using the most repeated value.

Handling Categorical Values:

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

- In our project, Gender ,married,dependents,self-employed,co-applicants income,loan amount ,loan amount term, credit history With list comprehension encoding is done.



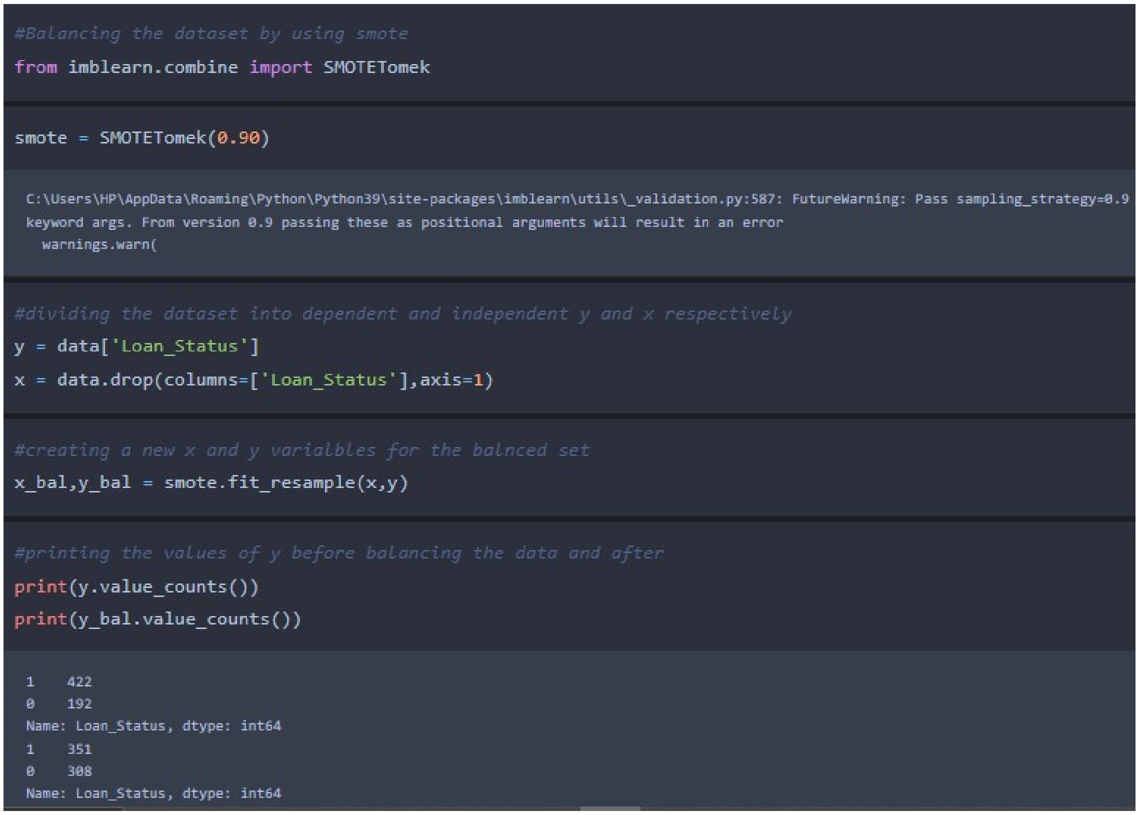
```
#changing the datype of each float column to int
data['Gender']=data['Gender'].astype('int64')
data['Married']=data['Married'].astype('int64')
data['Dependents']=data['Dependents'].astype('int64')
data['Self_Employed']=data['Self_Employed'].astype('int64')
data['CoapplicantIncome']=data['CoapplicantIncome'].astype('int64')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

Handling Imbalance Data:

Data Balancing is one of the most important step, which need to be performed for classification models, because when we train our model on imbalanced dataset ,we will get biassed results, which means our model is able to predict only one class element

For Balancing the data we are using the SMOTE Method.

SMOTE: Synthetic minority over sampling technique, which will create new synthetic data points for under class as per the requirements given by us using KNN method.



```
#Balancing the dataset by using smote
from imblearn.combine import SMOTETomek
```

```
smote = SMOTETomek(0.90)
```

```
 C:\Users\HP\AppData\Roaming\Python\Python39\site-packages\imblearn\utils\_validation.py:587: FutureWarning: Pass sampling_strategy=0.9
 keyword args. From version 0.9 passing these as positional arguments will result in an error
   warnings.warn(
```

```
#dividing the dataset into dependent and independent y and x respectively
y = data['Loan_Status']
x = data.drop(columns=['Loan_Status'],axis=1)
```

```
#creating a new x and y varialbles for the balnced set
x_bal,y_bal = smote.fit_resample(x,y)
```

```
#printing the values of y before balancing the data and after
print(y.value_counts())
print(y_bal.value_counts())
```

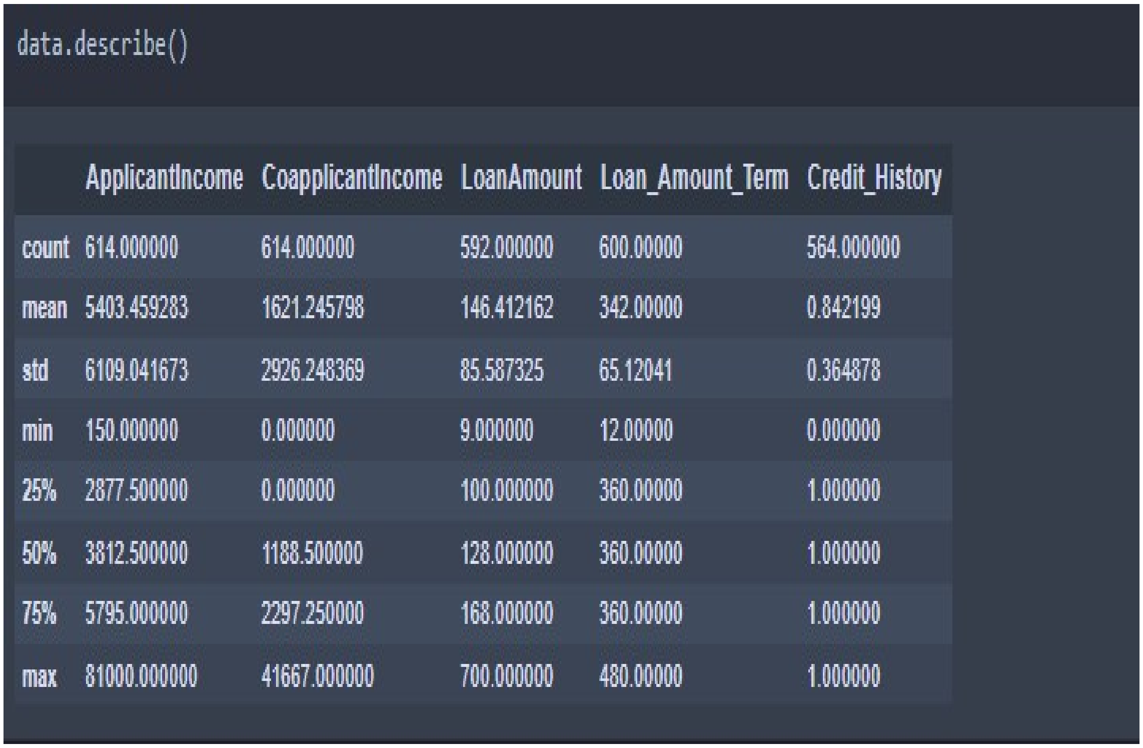
```
 1    422
 0    192
 Name: Loan_Status, dtype: int64
 1    351
 0    308
 Name: Loan_Status, dtype: int64
```

From the above picture, we can infer that ,previously our dataset had 492 class 1, and 192 class items, after applying smote technique on the dataset the size has been changed for minority class.

Exploratory Data Analysis:

Descriptive statistical:

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.



```
data.describe()
```

	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

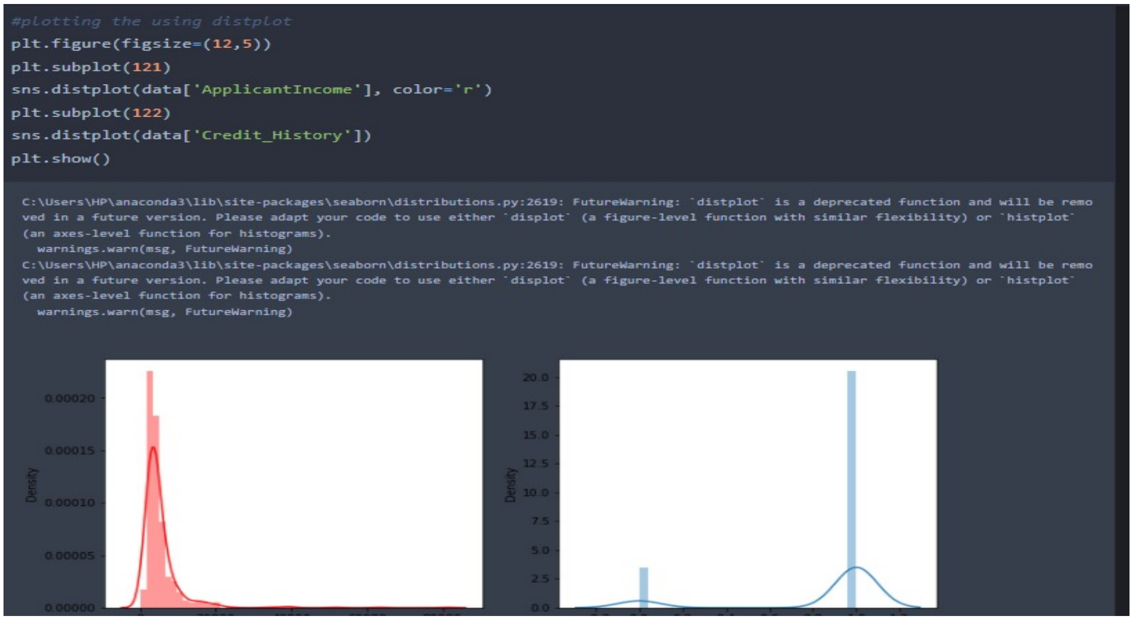
Visual analysis:

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Univariate analysis:

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

- The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.



```
#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\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be remo
ved in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot`
(an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be remo
ved in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot`
(an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy dataframe with categorical features. With for loop and subplot we have plotted this below graph.

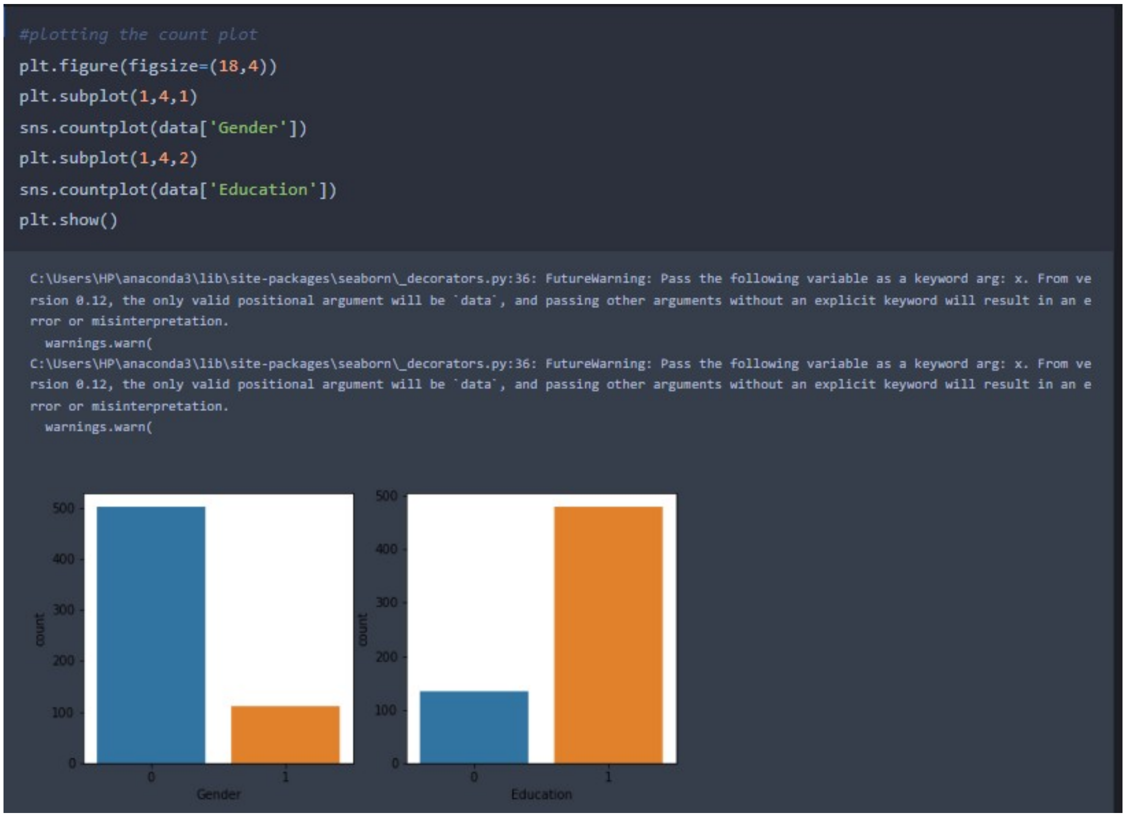
- From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.0

Countplot:-

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

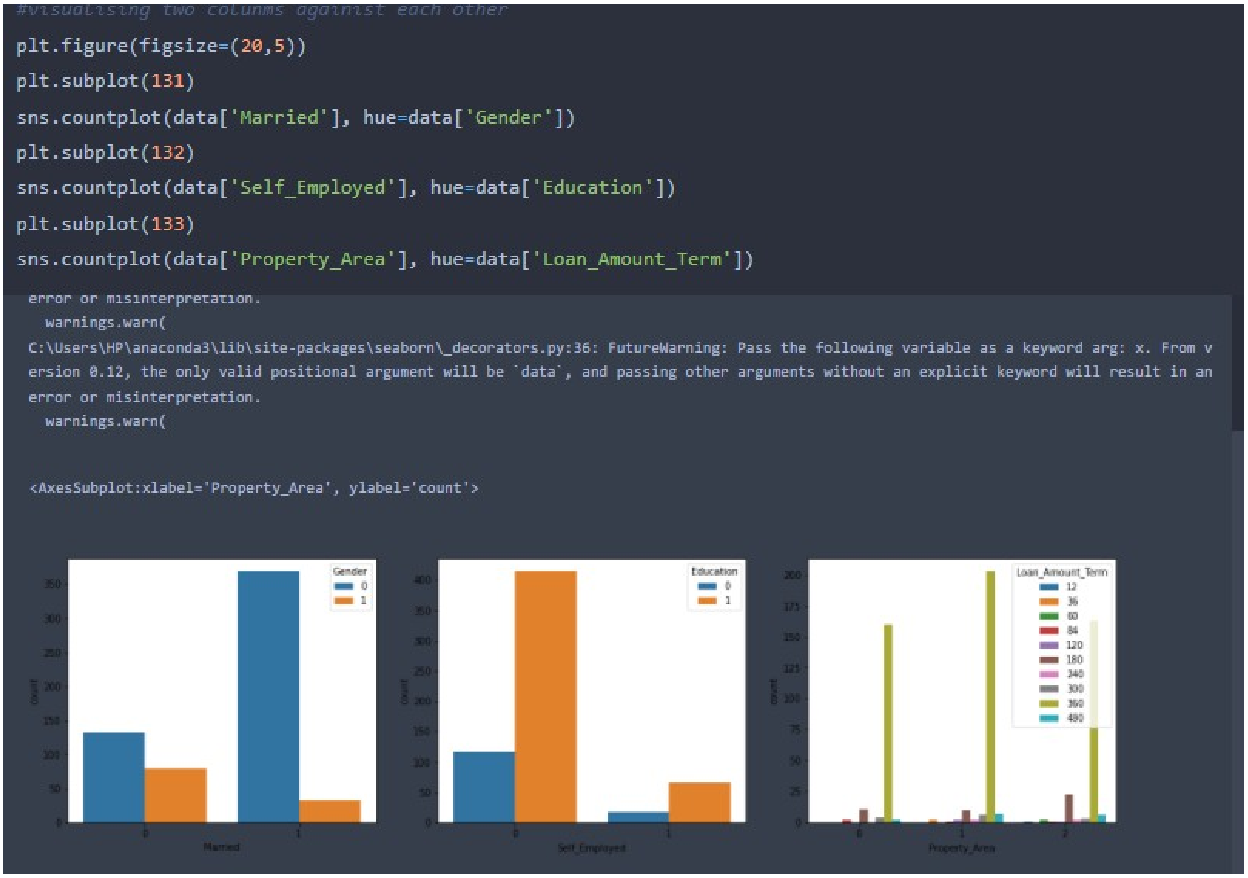
From the graph we can infer that , gender and education is a categorical variables with 2 categories , from gender column we can infer that 0-category is having more weightage than category-1,while education with 0,it means no education is a underclass when compared with category -1, which means educated .

Bivariate analysis:



```
#plotting the count plot
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(data['Gender'])
plt.subplot(1,4,2)
sns.countplot(data['Education'])
plt.show()
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From ve
rsion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an e
rror or misinterpretation.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From ve
rsion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an e
rror or misinterpretation.
  warnings.warn(
```



```
#visualising two columns against each other
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'])
```

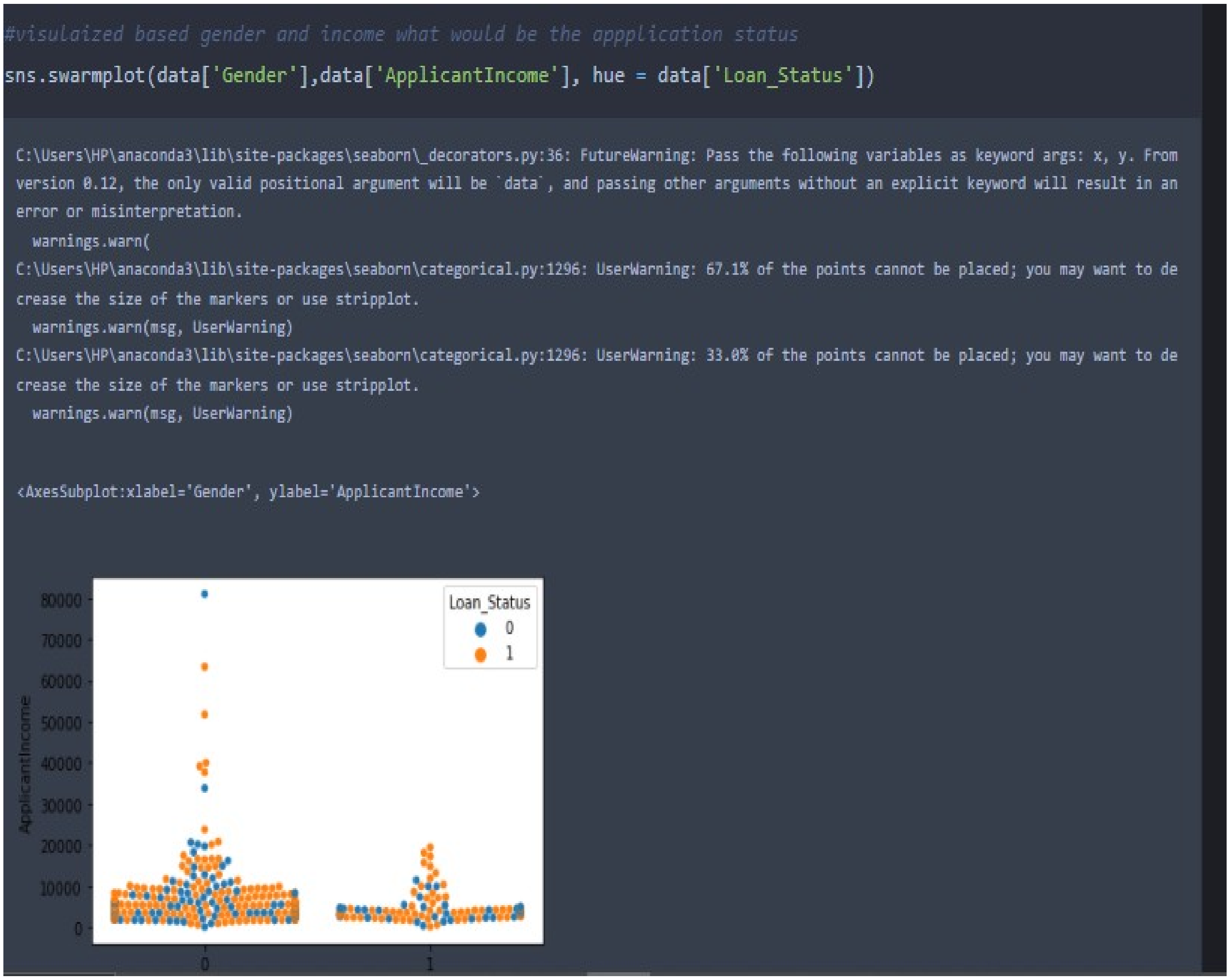
```
error or misinterpretation.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(


<AxesSubplot:xlabel='Property_Area', ylabel='count'>
```

From the above graph we can infer the analysis such as

- Segmenting the gender column and married column based on bar graphs
- Segmenting the Education and Self-employed based on bar graphs ,for drawing insights such as educated people are employed.
- Loan amount term based on the property area of a person holding

Multivariate analysis:

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used a swarm plot from the seaborn package.



```
#visulaized based gender and income what would be the appplication status
sns.swarmplot(data['Gender'],data['ApplicantIncome'], hue = data['Loan_Status'])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: 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.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 67.1% of the points cannot be placed; you may want to de
crease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\HP\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 33.0% of the points cannot be placed; you may want to de
crease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)


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

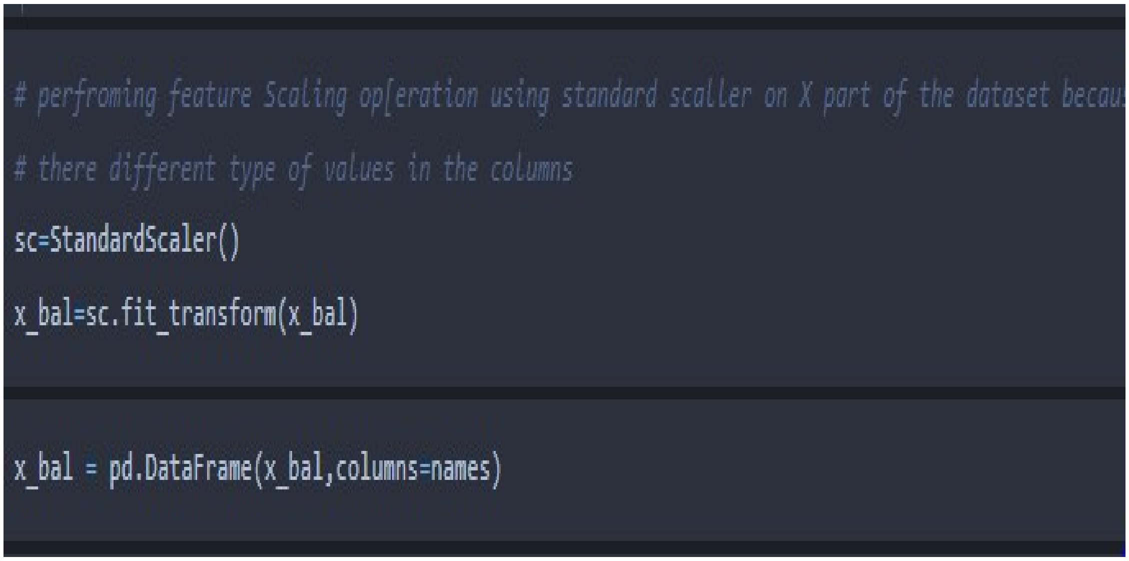
From the above graph we are plotting the relationship between the Gender, applicants income and loan status of the person.

Now, the code would be normalising the data by scaling it to have a similar range of values, and then splitting that data into a training set and a test set for training the model and testing its performance, respectively.

Scaling the Data:

Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction

Models such as KNN, Logistic regression need scaled data, as they follow distance based method and Gradient Descent concept.



```
# perfroming feature Scaling op[eration using standard scaller on X part of the dataset becau
# there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
```

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

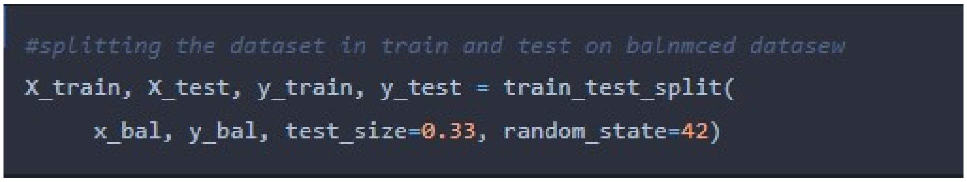
We will perform scaling only on the input values.Once the dataset is scaled, it will be converted into an array and we need to convert it back to a dataframe.

Splitting data into train and test:

Now let’s split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.



```
#splitting the dataset in train and test on balnmced datasew
X_train, X_test, y_train, y_test = train_test_split(
     x_bal, y_bal, test_size=0.33, random_state=42)
```

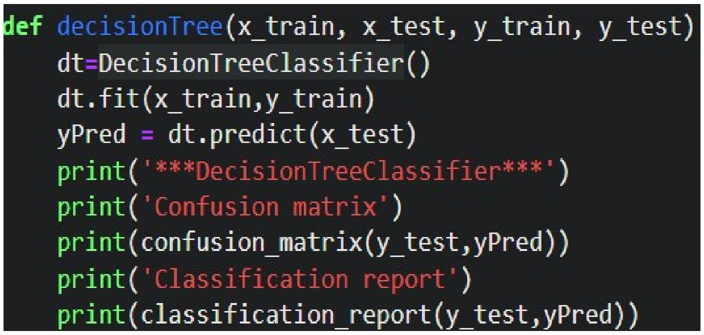
Model Building:

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is savedbased on its performance.

Decision tree model:

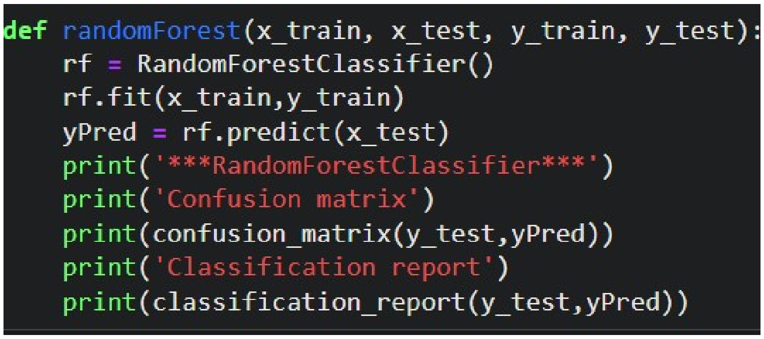
A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



```
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))
```

Random forest model:

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

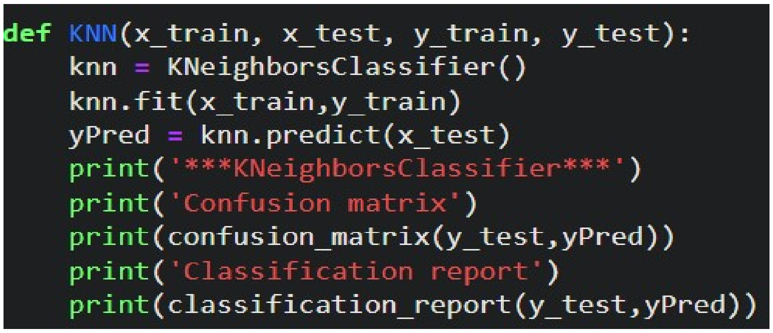


```
def randomForest(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
  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

KNN model:

A function named KNN is created and train and test data are passed as the parameters. Inside thefunction, KNeighborsClassifier algorithm is initialised and training data is passed to the model with

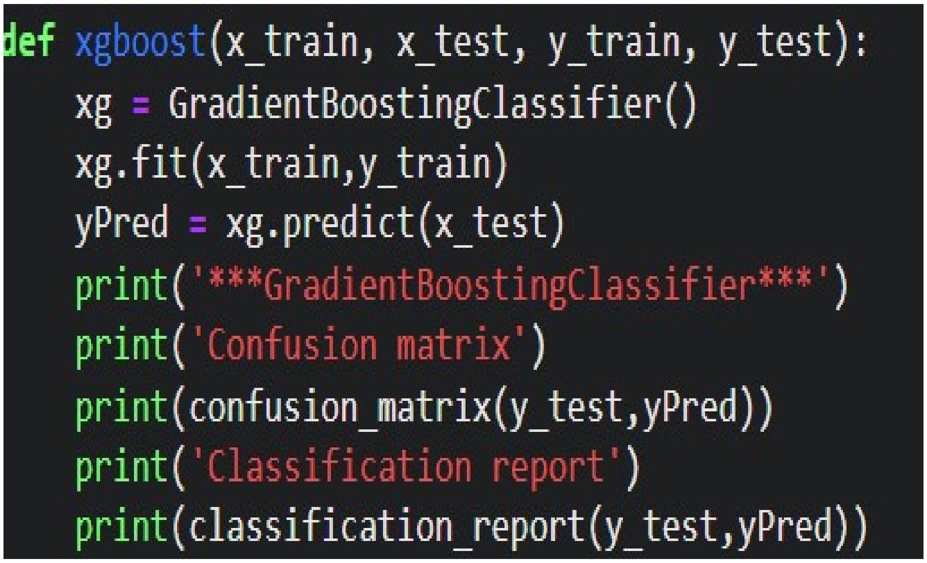
.fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.



```
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))
```

Xgboost model:

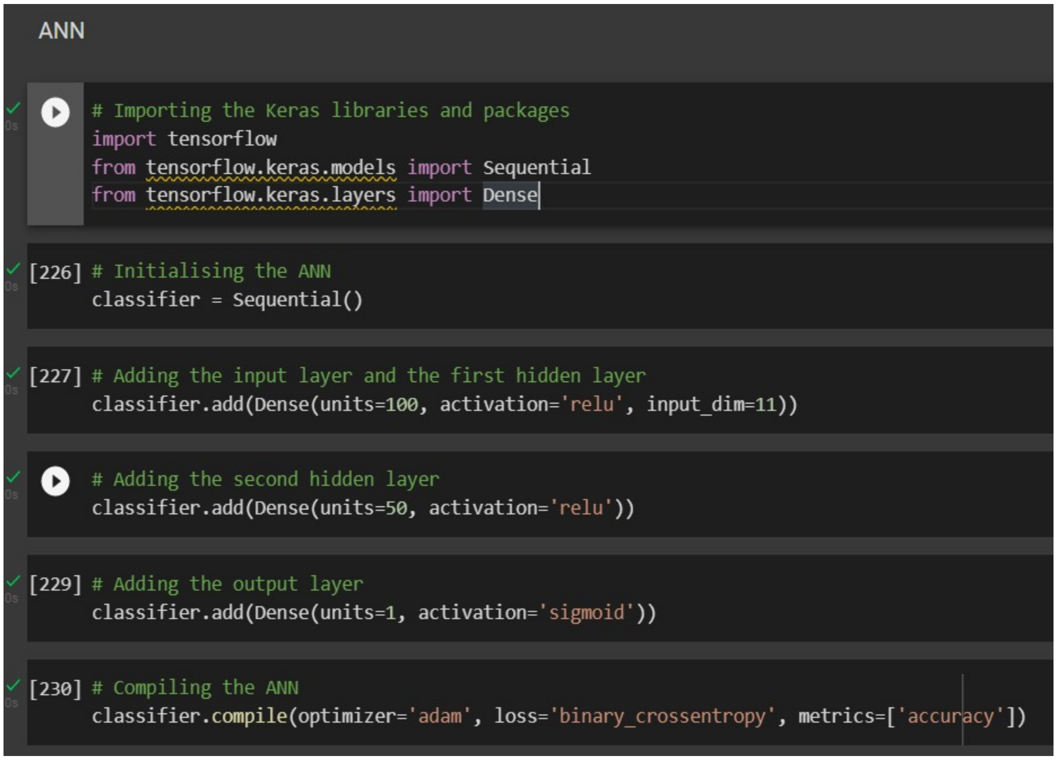
A function named xgboost is created and train and test data are passed as the parameters. Insidethe function, GradientBoostingClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.



```
def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print('***GradientBoostingClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

ANN model:

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.



```
# Importing the Keras libraries and packages
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

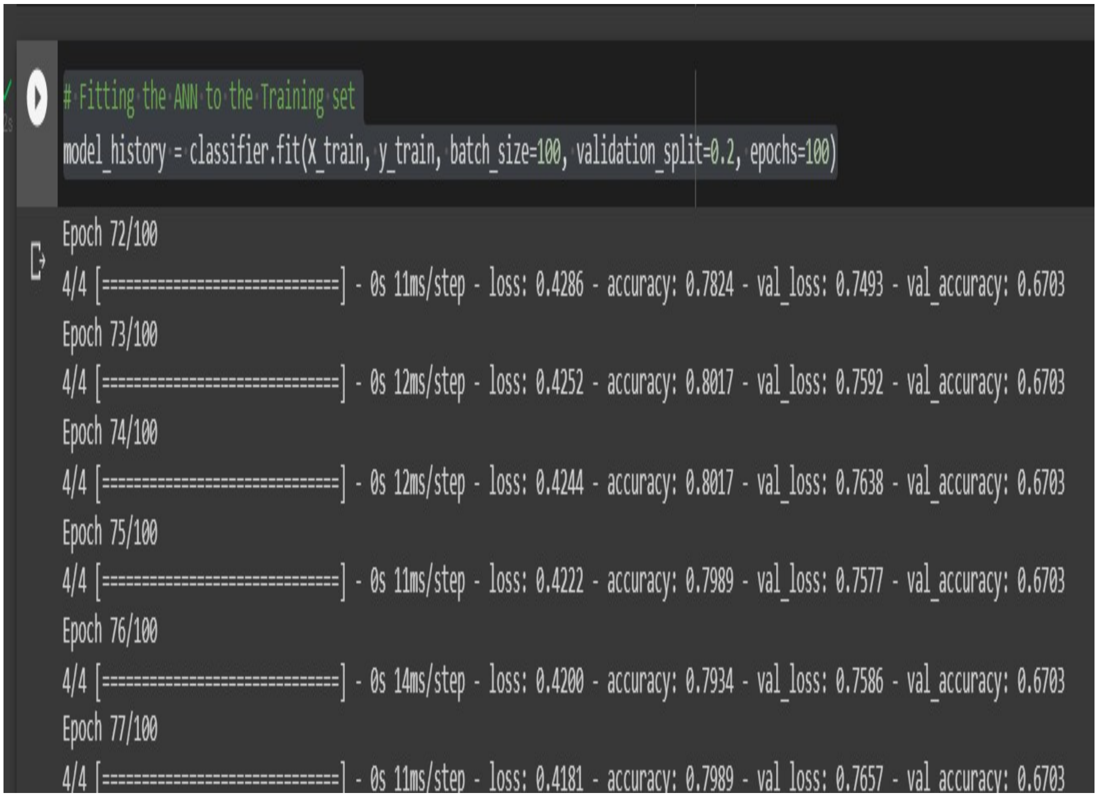
```
[226] # Initialising the ANN
      classifier = Sequential()
```

```
[227] # Adding the input layer and the first hidden layer
      classifier.add(Dense(units=100, activation='relu', input_dim=11))
```

```
# Adding the second hidden layer
classifier.add(Dense(units=50, activation='relu'))
```

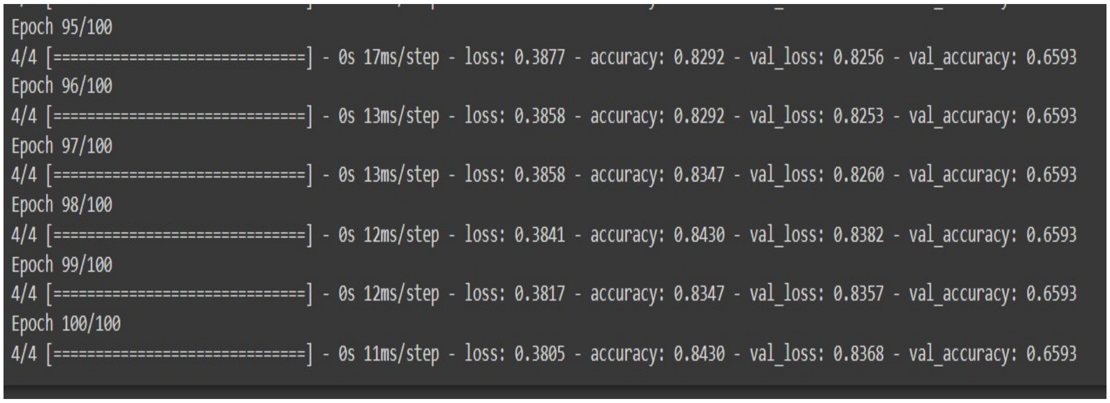
```
[229] # Adding the output layer
      classifier.add(Dense(units=1, activation='sigmoid'))
```

```
[230] # Compiling the ANN
      classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```



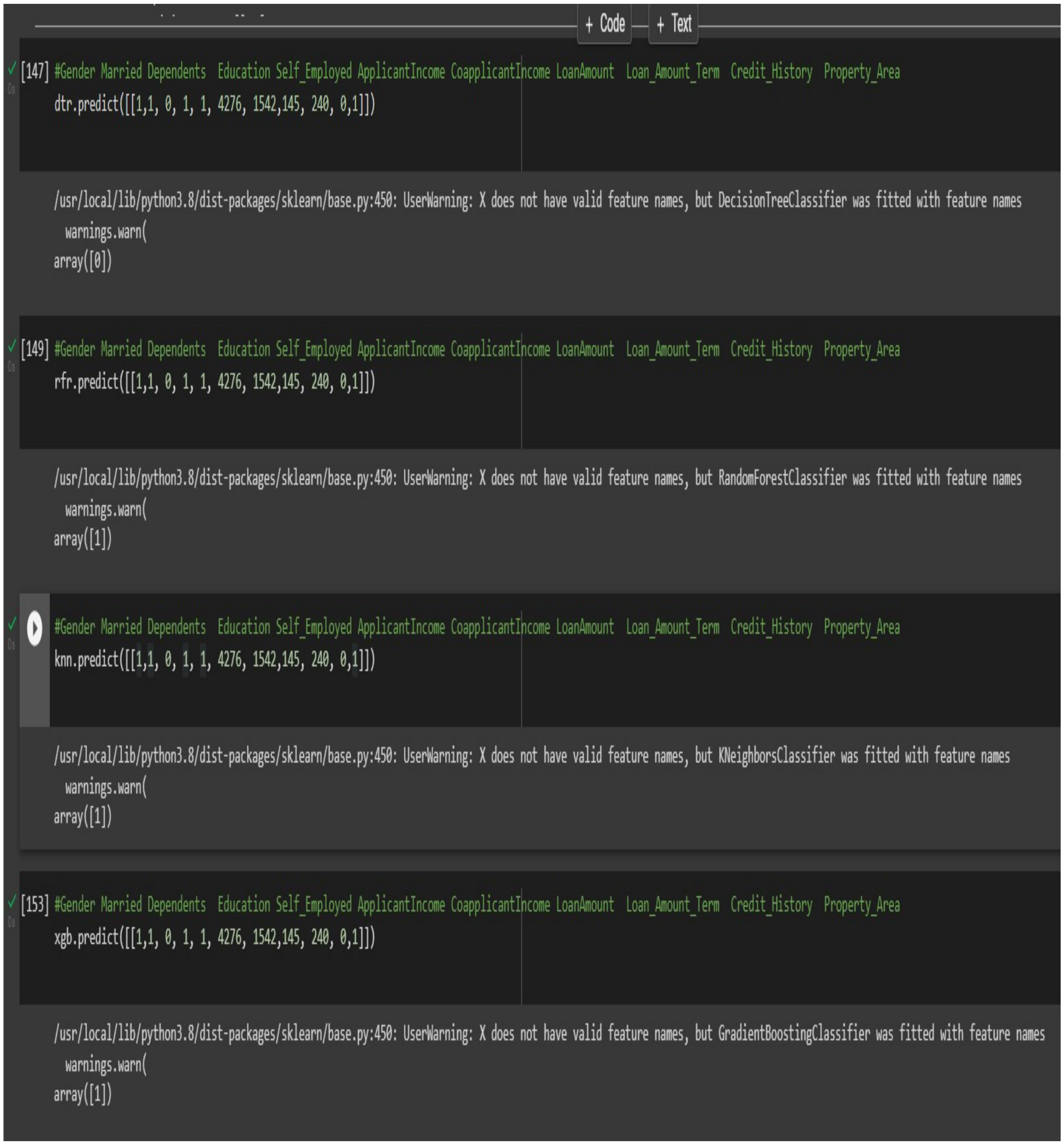
```
# Fitting the ANN to the Training set
model_history = classifier.fit(X_train, y_train, batch_size=100, validation_split=0.2, epochs=100)
```

```
Epoch 72/100
4/4 [==============================] - 0s 11ms/step - loss: 0.4286 - accuracy: 0.7824 - val_loss: 0.7493 - val_accuracy: 0.6703
Epoch 73/100
4/4 [==============================] - 0s 12ms/step - loss: 0.4252 - accuracy: 0.8017 - val_loss: 0.7592 - val_accuracy: 0.6703
Epoch 74/100
4/4 [==============================] - 0s 12ms/step - loss: 0.4244 - accuracy: 0.8017 - val_loss: 0.7638 - val_accuracy: 0.6703
Epoch 75/100
4/4 [==============================] - 0s 11ms/step - loss: 0.4222 - accuracy: 0.7989 - val_loss: 0.7577 - val_accuracy: 0.6703
Epoch 76/100
4/4 [==============================] - 0s 14ms/step - loss: 0.4200 - accuracy: 0.7934 - val_loss: 0.7586 - val_accuracy: 0.6703
Epoch 77/100
4/4 [==============================] - 0s 11ms/step - loss: 0.4181 - accuracy: 0.7989 - val_loss: 0.7657 - val_accuracy: 0.6703
```



```
Epoch 95/100
4/4 [==============================] - 0s 17ms/step - loss: 0.3877 - accuracy: 0.8292 - val_loss: 0.8256 - val_accuracy: 0.6593
Epoch 96/100
4/4 [==============================] - 0s 13ms/step - loss: 0.3858 - accuracy: 0.8292 - val_loss: 0.8253 - val_accuracy: 0.6593
Epoch 97/100
4/4 [==============================] - 0s 13ms/step - loss: 0.3858 - accuracy: 0.8347 - val_loss: 0.8260 - val_accuracy: 0.6593
Epoch 98/100
4/4 [==============================] - 0s 12ms/step - loss: 0.3841 - accuracy: 0.8430 - val_loss: 0.8382 - val_accuracy: 0.6593
Epoch 99/100
4/4 [==============================] - 0s 12ms/step - loss: 0.3817 - accuracy: 0.8347 - val_loss: 0.8357 - val_accuracy: 0.6593
Epoch 100/100
4/4 [==============================] - 0s 11ms/step - loss: 0.3805 - accuracy: 0.8430 - val_loss: 0.8368 - val_accuracy: 0.6593
```

Testing the model:



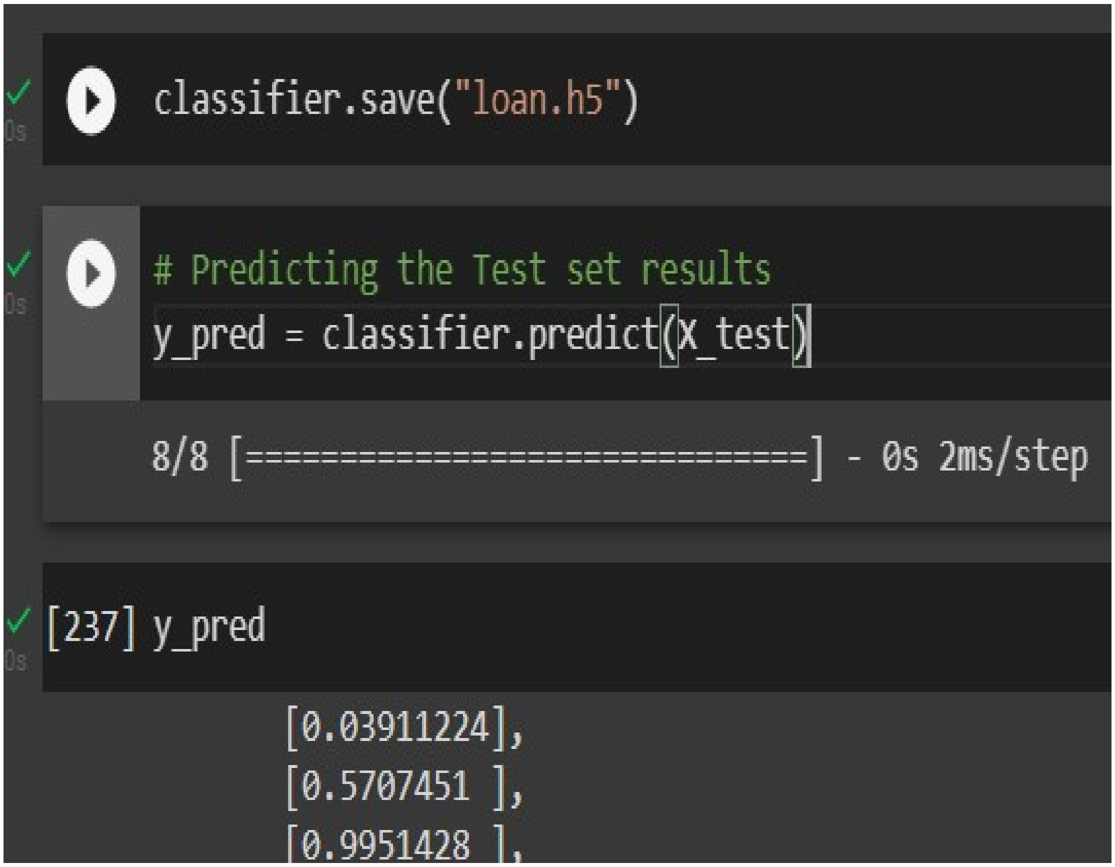
```
[147] #Gender Married Dependents  Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount  Loan_Amount_Term  Credit_History  Property_Area
      dtr.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])

      /usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
        warnings.warn(
      array([0])
```

```
[149] #Gender Married Dependents  Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount  Loan_Amount_Term  Credit_History  Property_Area
      rfr.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])

      /usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
        warnings.warn(
      array([1])
```

```
      #Gender Married Dependents  Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount  Loan_Amount_Term  Credit_History  Property_Area
      knn.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])

      /usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
        warnings.warn(
      array([1])
```

```
[153] #Gender Married Dependents  Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount  Loan_Amount_Term  Credit_History  Property_Area
      xgb.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])

      /usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but GradientBoostingClassifier was fitted with feature names
        warnings.warn(
      array([1])
```

In ANN we first have to save the model to the test the inputs



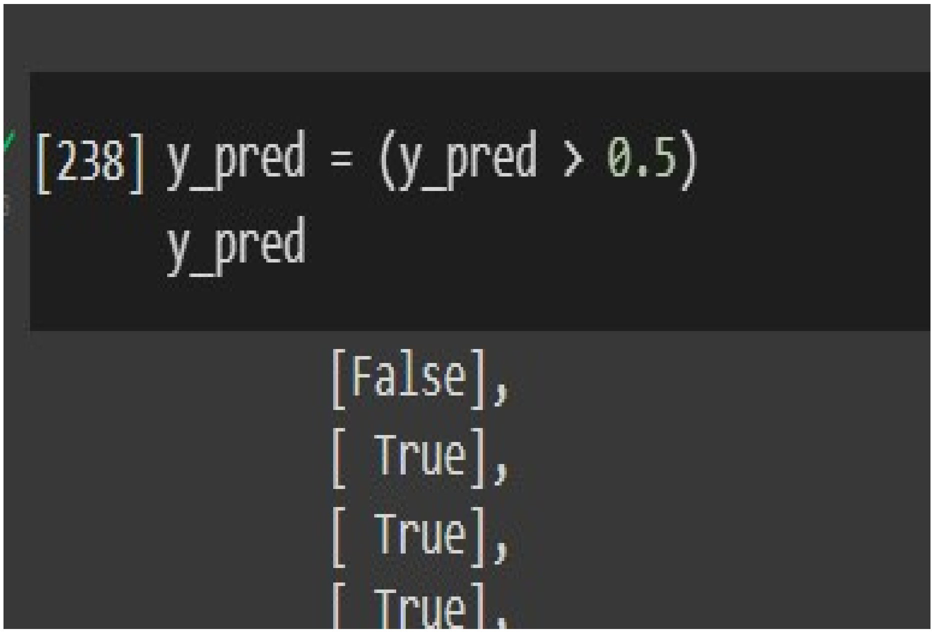
```
classifier.save("loan.h5")
```

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

```
8/8 [==============================] - 0s 2ms/step
```

```
[237] y_pred
```

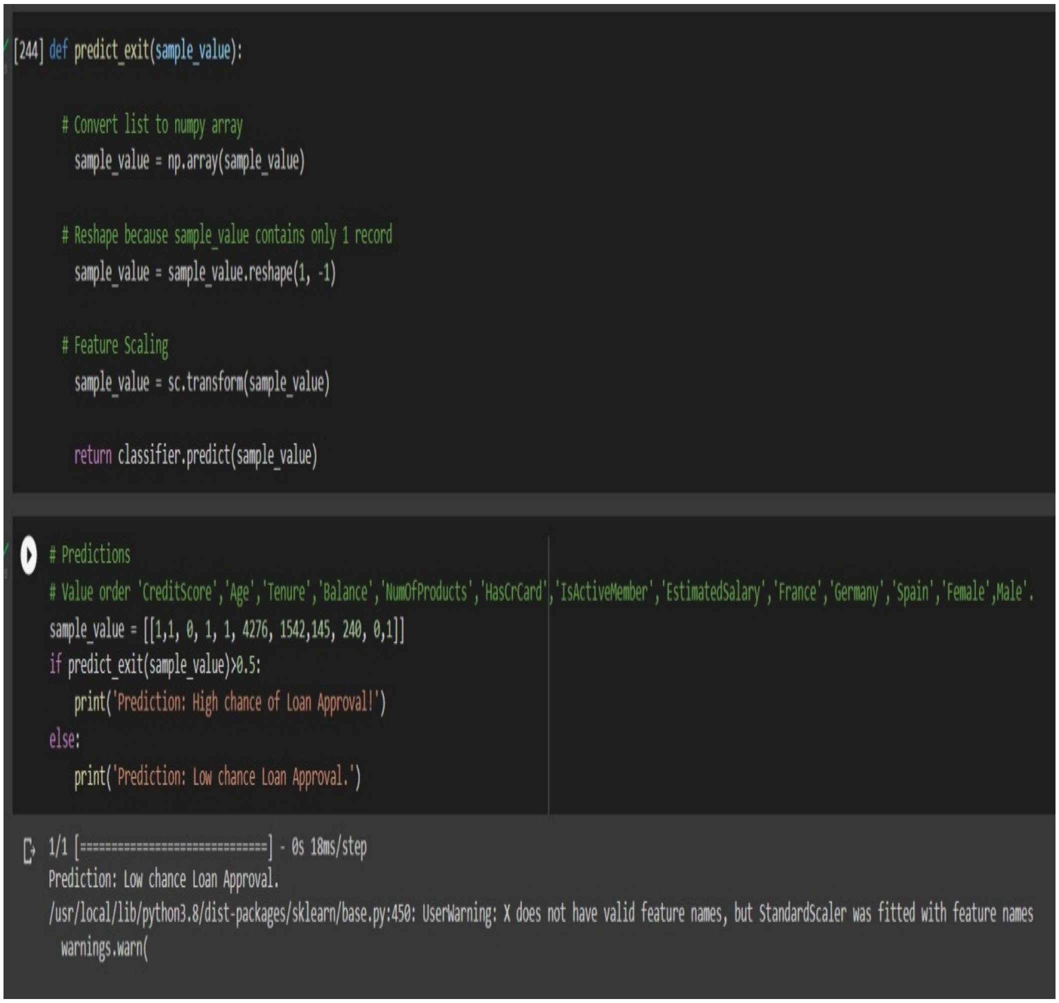
```
        [0.03911224],
        [0.5707451 ],
        [0.9951428 ],
```



```
[238] y_pred = (y_pred > 0.5)
      y_pred
```

```
        [False],
        [ True],
        [ True],
        [ True],
```

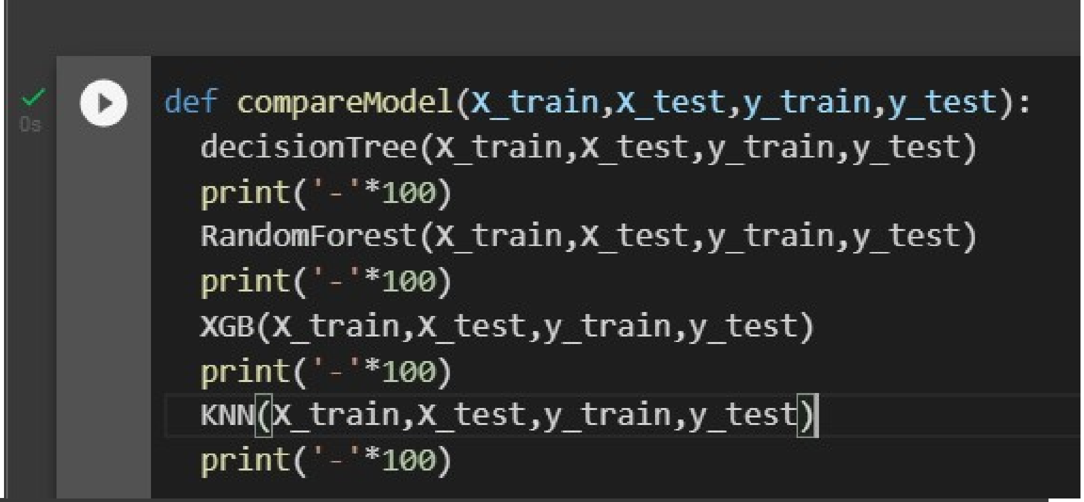
This code defines a function named "predict_exit" which takes in a sample_value as an input. The function then converts the input sample_value from a list to a numpy array. It reshapes the sample_value array as it contains only one record. Then, it applies feature scaling to the reshaped sample_value array using a scaler object 'sc' that should have been previously defined and fitted. Finally, the function returns the prediction of the classifier on the scaled sample_value.



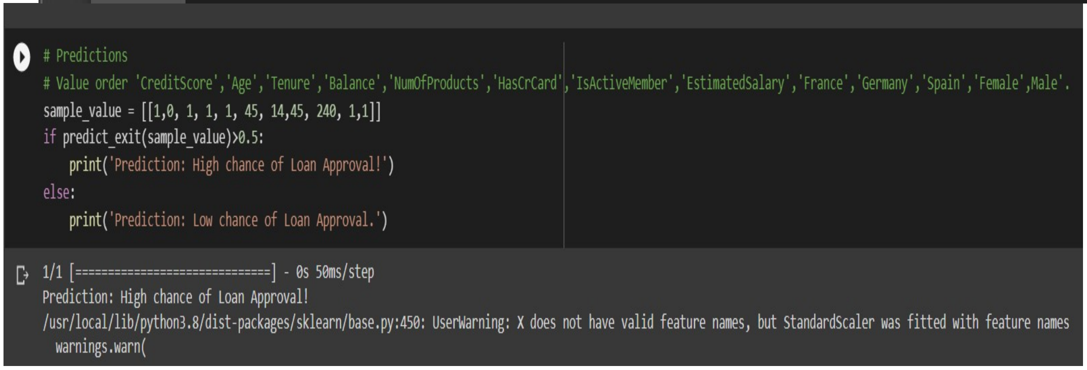
```
[244] def predict_exit(sample_value):

    # Convert list to numpy array
     sample_value = np.array(sample_value)

    # Reshape because sample_value contains only 1 record
     sample_value = sample_value.reshape(1, -1)

    # Feature Scaling
     sample_value = sc.transform(sample_value)

     return classifier.predict(sample_value)
```

```
# Predictions
# Value order 'CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary','France','Germany','Spain','Female',Male'.
sample_value = [[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]]
if predict_exit(sample_value)>0.5:
    print('Prediction: High chance of Loan Approval!')
else:
    print('Prediction: Low chance Loan Approval.')
```

```
1/1 [==============================] - 0s 18ms/step
Prediction: Low chance Loan Approval.
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```



```
def compareModel(X_train,X_test,y_train,y_test):
  decisionTree(X_train,X_test,y_train,y_test)
  print('-'*100)
  RandomForest(X_train,X_test,y_train,y_test)
  print('-'*100)
  XGB(X_train,X_test,y_train,y_test)
  print('-'*100)
  KNN(X_train,X_test,y_train,y_test)
  print('-'*100)
```



```
# Predictions
# Value order 'CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary','France','Germany','Spain','Female',Male'.
sample_value = [[1,0, 1, 1, 1, 45, 14,45, 240, 1,1]]
if predict_exit(sample_value)>0.5:
    print('Prediction: High chance of Loan Approval!')
else:
    print('Prediction: Low chance of Loan Approval.')
```

```
1/1 [==============================] - 0s 50ms/step
Prediction: High chance of Loan Approval!
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

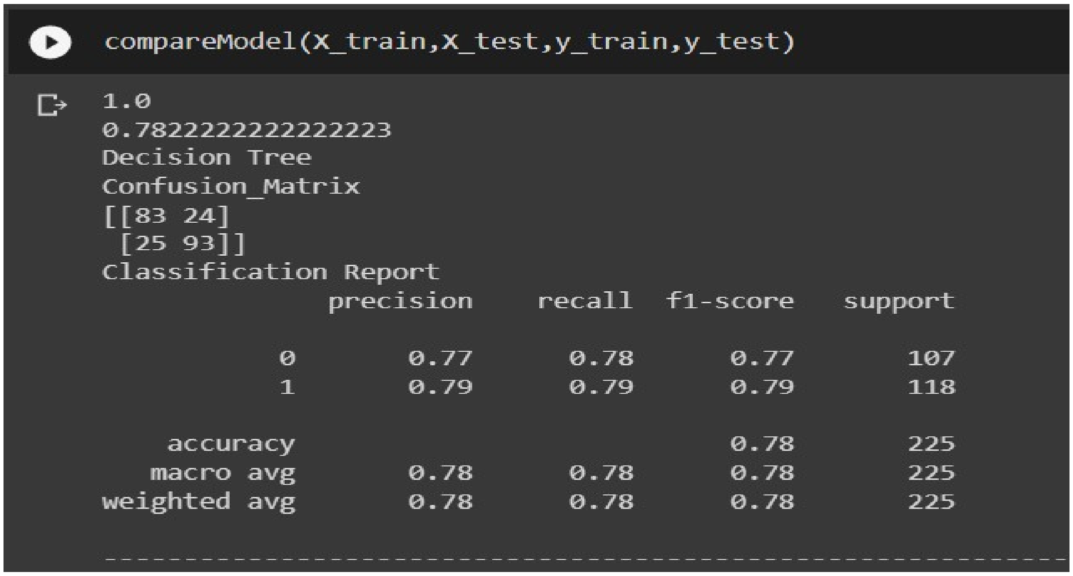
Performance Testing & Hyperparameter Tuning:

Testing model with multiple evaluation metrics:

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

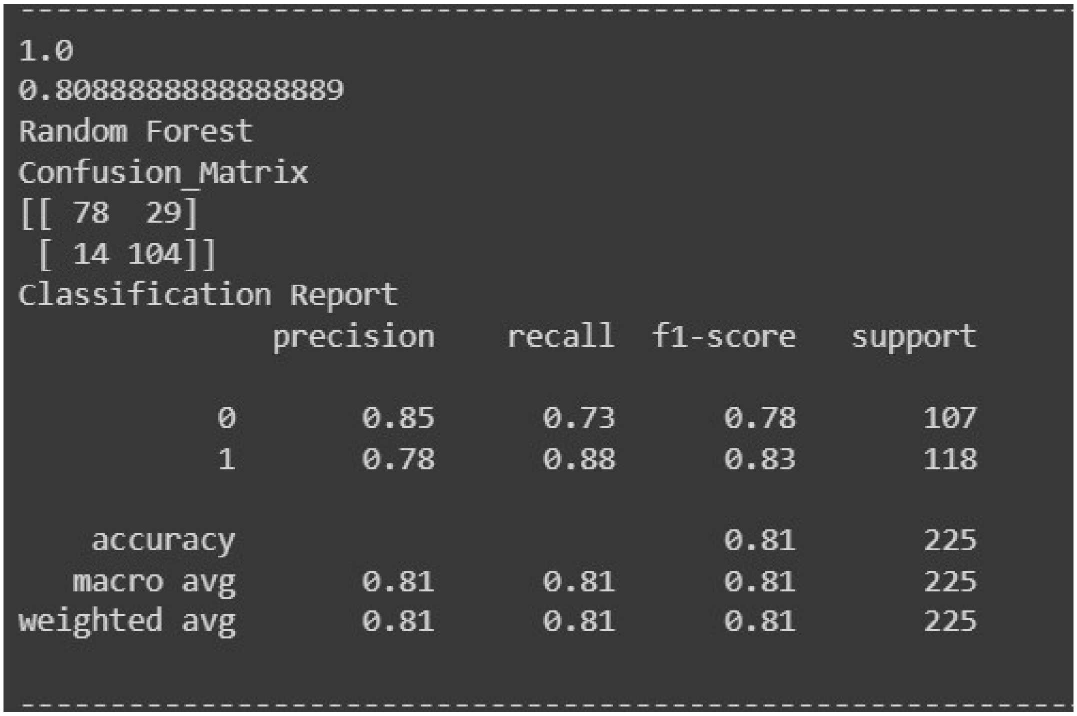
Compare the model:

For comparing the above four models, the compareModel function is defined.

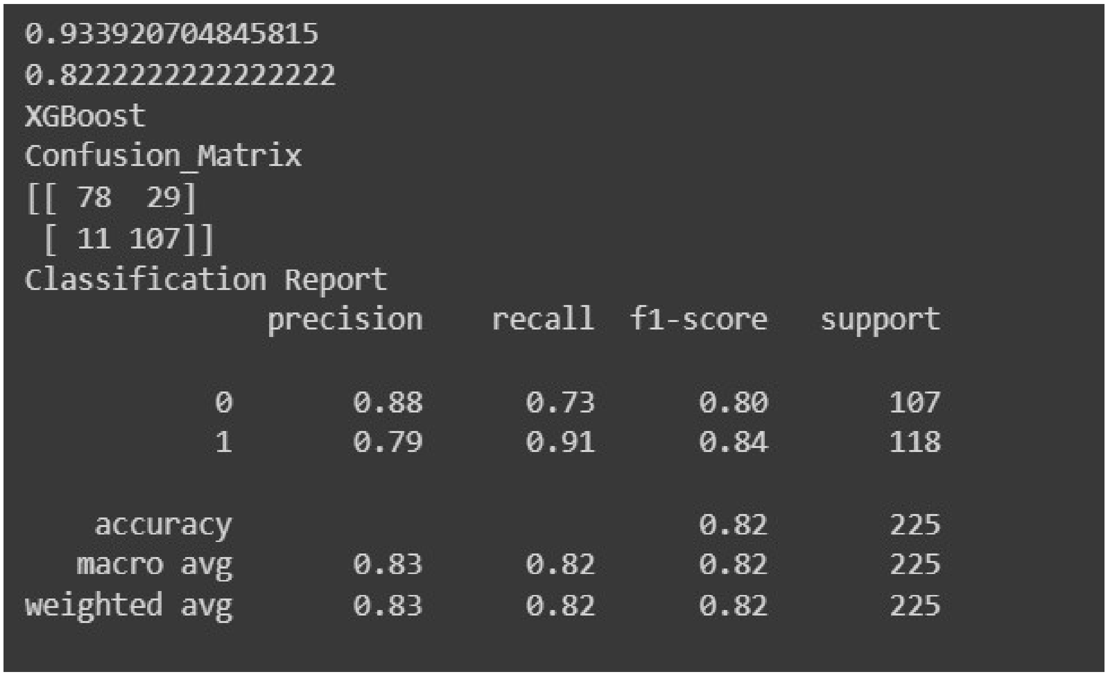


```
compareModel(X_train,X_test,y_train,y_test)
```

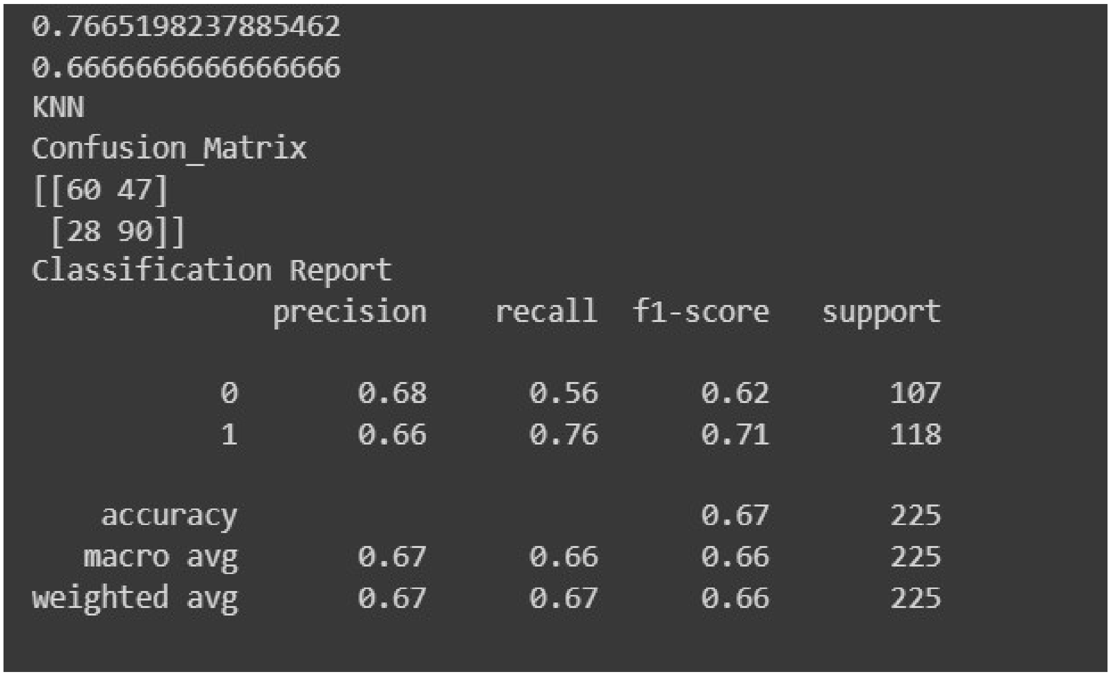
```
1.0
0.7822222222222223
Decision Tree
Confusion_Matrix
[[83 24]
 [25 93]]
Classification Report
              precision    recall  f1-score   support

           0       0.77      0.78      0.77       107
           1       0.79      0.79      0.79       118

    accuracy                           0.78       225
   macro avg       0.78      0.78      0.78       225
weighted avg       0.78      0.78      0.78       225

-----------------------------------------------------------
```



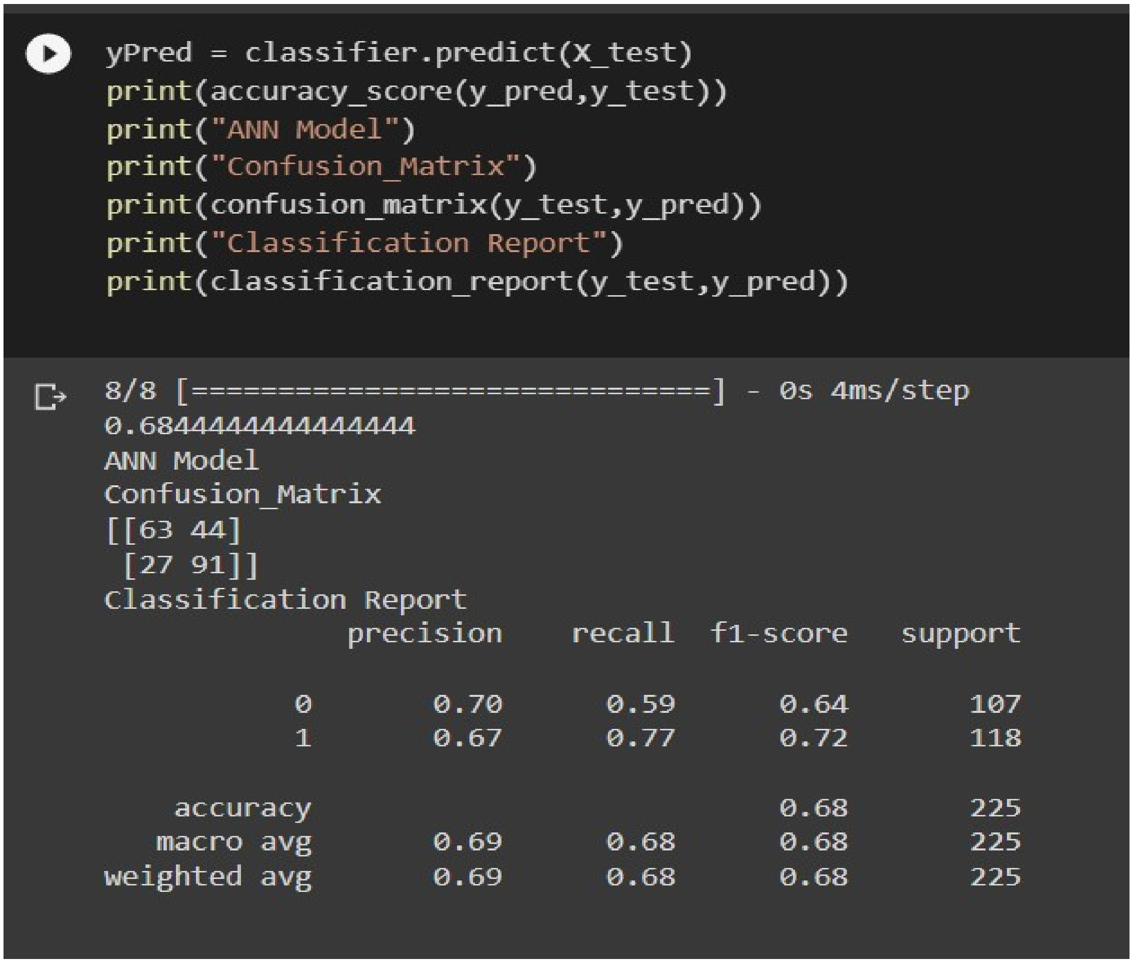
```
-----------------------------------------------------------
1.0
0.8088888888888889
Random Forest
Confusion_Matrix
[[ 78  29]
 [ 14 104]]
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.73      0.78       107
           1       0.78      0.88      0.83       118

    accuracy                           0.81       225
   macro avg       0.81      0.81      0.81       225
weighted avg       0.81      0.81      0.81       225


-----------------------------------------------------------
```



```
0.933920704845815
0.8222222222222222
XGBoost
Confusion_Matrix
[[ 78  29]
 [ 11 107]]
Classification Report
              precision    recall  f1-score   support

           0       0.88      0.73      0.80       107
           1       0.79      0.91      0.84       118

    accuracy                           0.82       225
   macro avg       0.83      0.82      0.82       225
weighted avg       0.83      0.82      0.82       225
```



```
0.7665198237885462
0.6666666666666666
KNN
Confusion_Matrix
[[60 47]
 [28 90]]
Classification Report
              precision    recall  f1-score   support

           0       0.68      0.56      0.62       107
           1       0.66      0.76      0.71       118

    accuracy                           0.67       225
   macro avg       0.67      0.66      0.66       225
weighted avg       0.67      0.67      0.66       225
```



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

```
8/8 [==============================] - 0s 4ms/step
0.6844444444444444
ANN Model
Confusion_Matrix
[[63 44]
 [27 91]]
Classification Report
              precision    recall  f1-score   support

           0       0.70      0.59      0.64       107
           1       0.67      0.77      0.72       118

    accuracy                           0.68       225
   macro avg       0.69      0.68      0.68       225
weighted avg       0.69      0.68      0.68       225
```

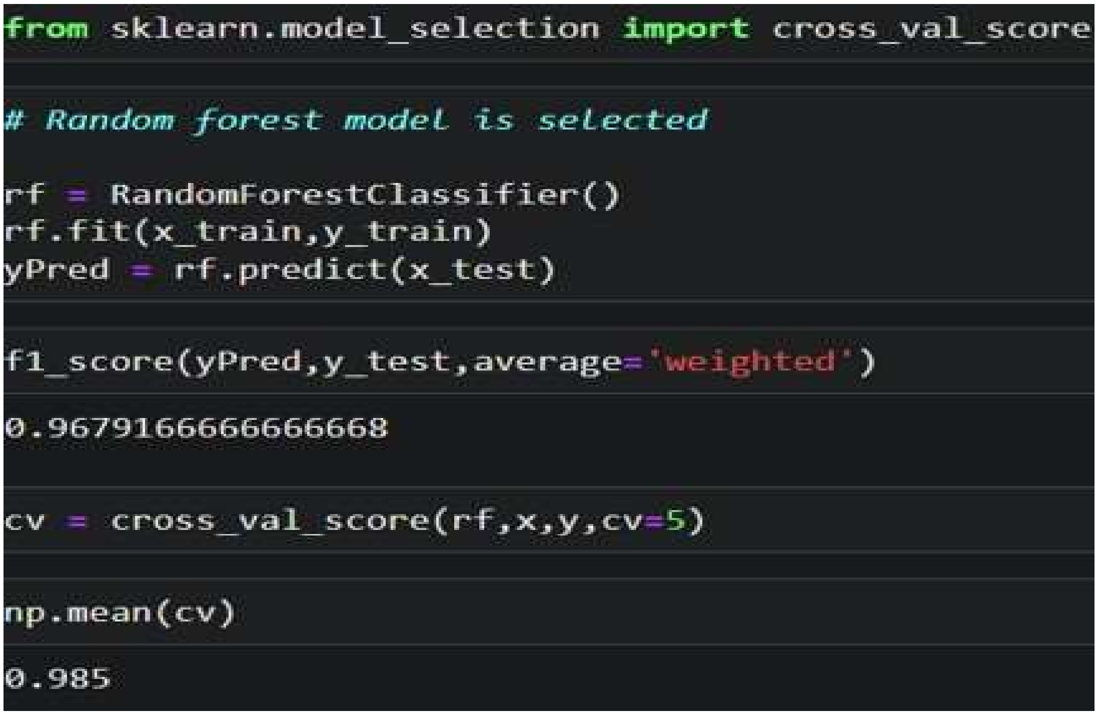
After calling the function, the results of models are displayed as output.

From the five models Xgboost is performing well. From the below image, We can see the accuracy of the model. Xgboost is giving the accuracy of 93.39% with training data , 82.2% accuracy for the testing data.

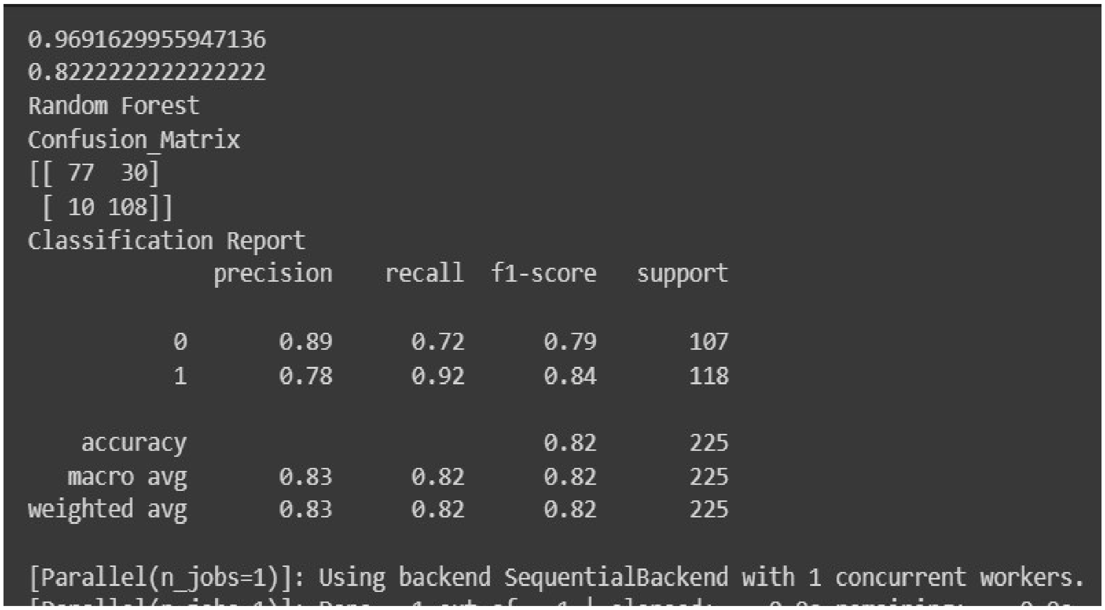
Comparing model accuracy before & after applying hyperparameter tuning:

Evaluating performance of the model From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

Note: To understand cross validation, refer to this link



```
from sklearn.model_selection import cross_val_score

# Random forest model is selected

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

f1_score(yPred,y_test,average='weighted')

0.9679166666666668

cv = cross_val_score(rf,x,y,cv=5)

np.mean(cv)

0.985
```

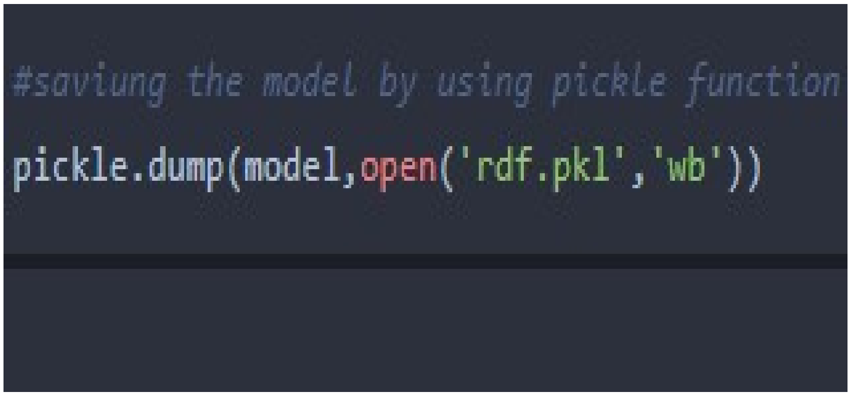


```
0.9691629955947136
0.8222222222222222
Random Forest
Confusion_Matrix
[[ 77  30]
 [ 10 108]]
Classification Report
              precision    recall  f1-score   support

           0       0.89      0.72      0.79       107
           1       0.78      0.92      0.84       118

    accuracy                           0.82       225
   macro avg       0.83      0.82      0.82       225
weighted avg       0.83      0.82      0.82       225

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

Model Deployment:

Save the best model:

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.



```
#saviung the model by using pickle function
pickle.dump(model,open('rdf.pkl','wb'))
```

Integrate with Web Framework:

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script
- Run the web application

Building Html Pages:

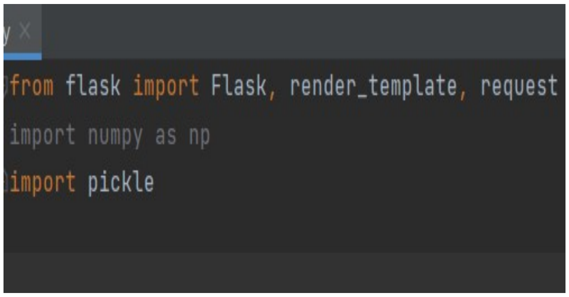
For this project create two HTML files namely

- home.html
- predict.html

and save them in the templates folder.

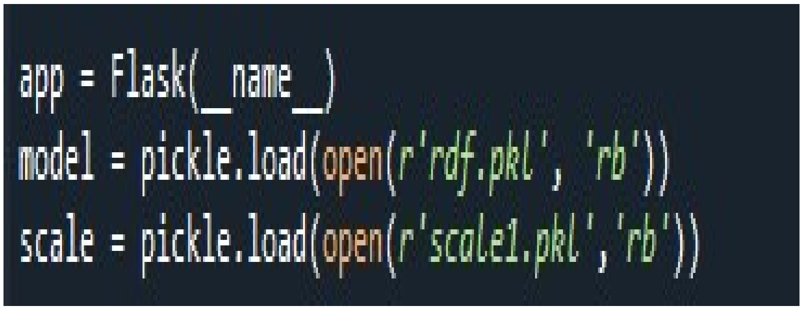
Build Python code:

Import the libraries



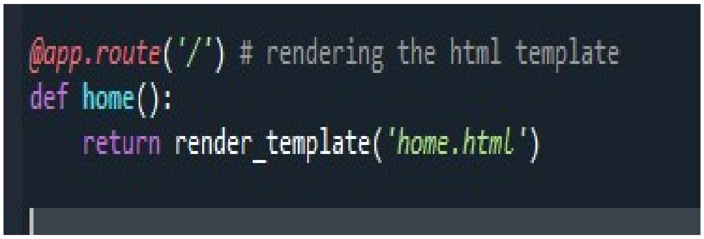
```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (_name__) as argument.



```
app = Flask(__name__)
model = pickle.load(open(r'rdf.pkl', 'rb'))
scale = pickle.load(open(r'scale1.pkl','rb'))
```

Render HTML page:



```
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

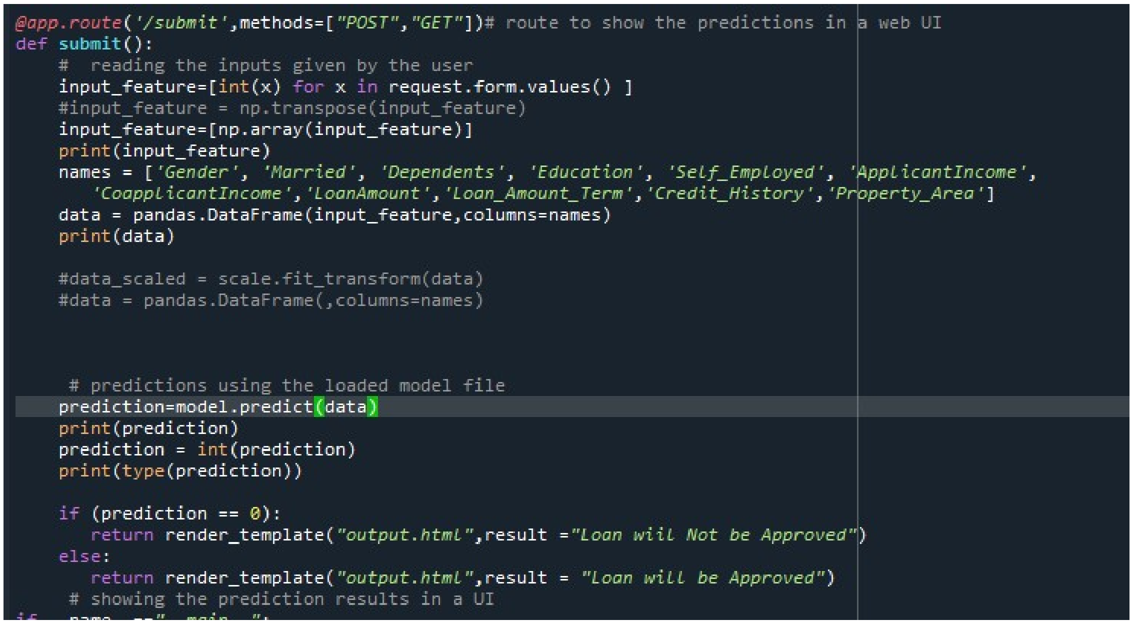
which we have created earlier.

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

Here we will be using a declared constructor to route to the HTML page

.

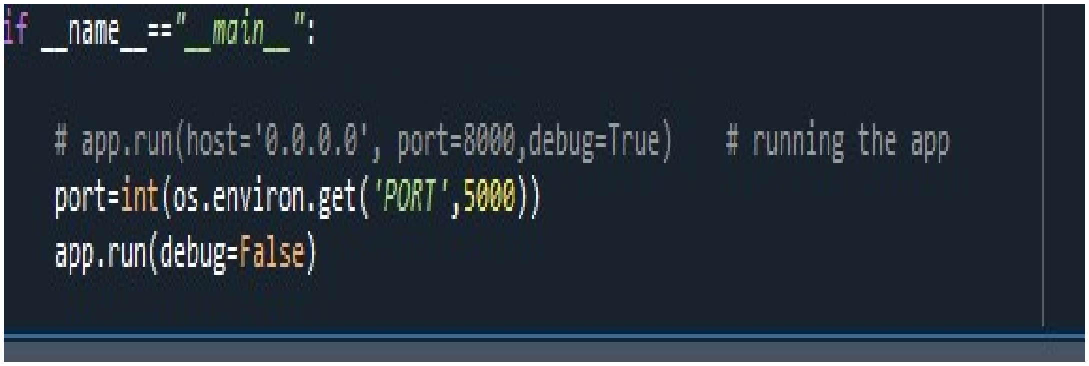
Retrieves the value from UI:



```
@app.route('/submit',methods=["POST","GET"])# route to show the predictions in a web UI
def submit():
    #  reading the inputs given by the user
    input_feature=[int(x) for x in request.form.values() ]
    #input_feature = np.transpose(input_feature)
    input_feature=[np.array(input_feature)]
    print(input_feature)
    names = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'ApplicantIncome',
       'CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','Property_Area']
    data = pandas.DataFrame(input_feature,columns=names)
    print(data)

    #data_scaled = scale.fit_transform(data)
    #data = pandas.DataFrame(,columns=names)



     # predictions using the loaded model file
    prediction=model.predict(data)
    print(prediction)
    prediction = int(prediction)
    print(type(prediction))

    if (prediction == 0):
       return render_template("output.html",result ="Loan wiil Not be Approved")
    else:
       return render_template("output.html",result = "Loan will be Approved")
     # showing the prediction results in a UI
```

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

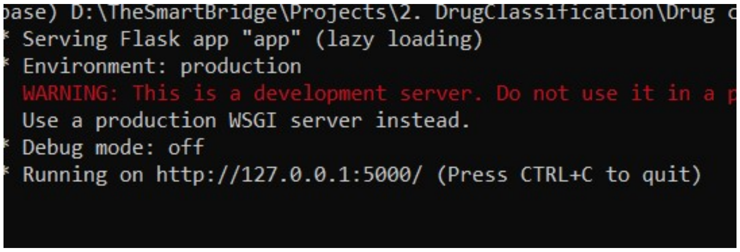
Main Function:



```
if __name__=="__main__":

    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(debug=False)
```

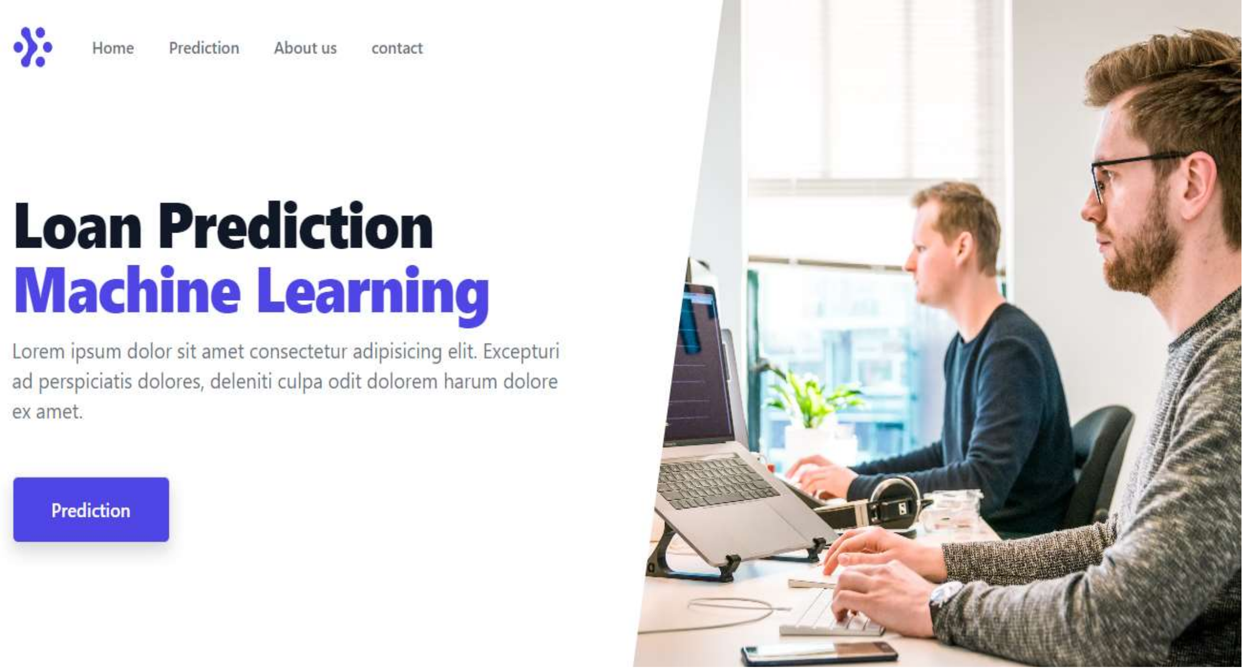
Run the web application:

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

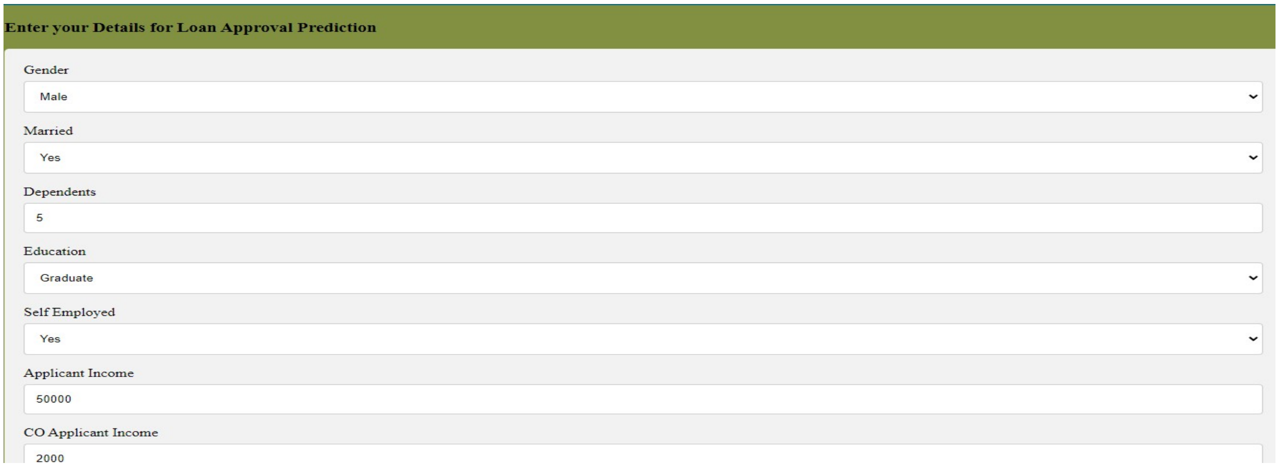
Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result



Loan Prediction
Machine Learning

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

Now,when you click on click me to predict the button from the banner you will get redirected to the prediction page.



Enter your Details for Loan Approval Prediction

Gender

Male

Married

Yes

Dependents

5

Education

Graduate

Self Employed

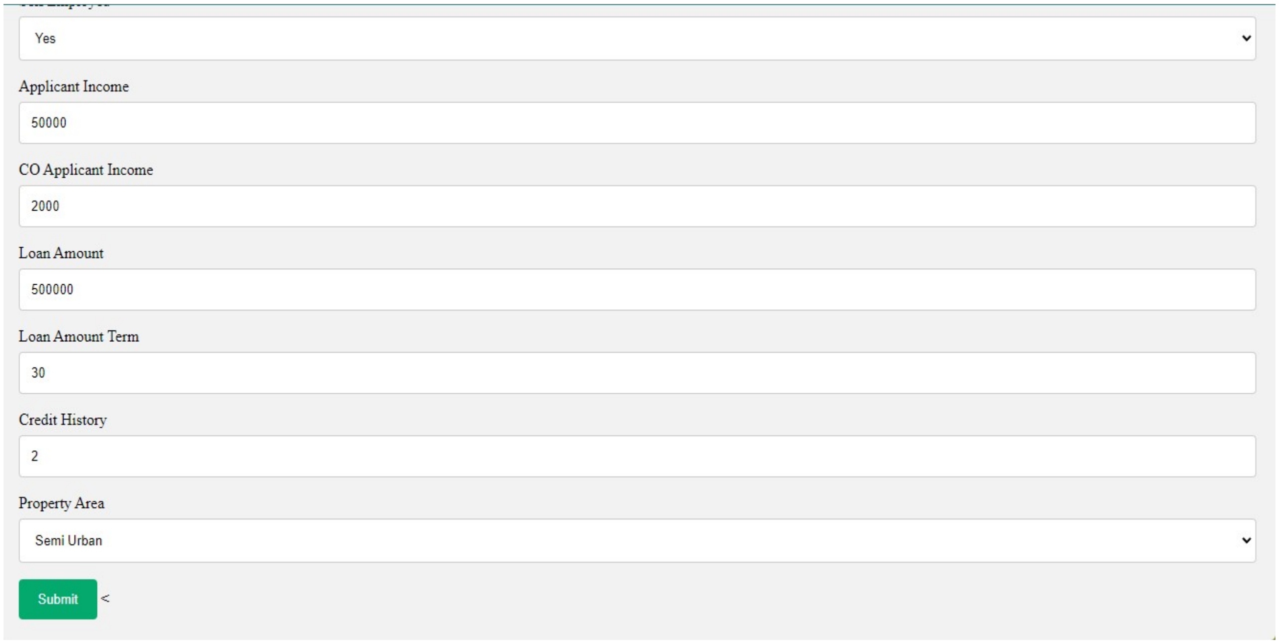
Yes

Applicant Income

50000

CO Applicant Income

2000



Yes

Applicant Income

50000

CO Applicant Income

2000

Loan Amount

500000

Loan Amount Term

30

Credit History

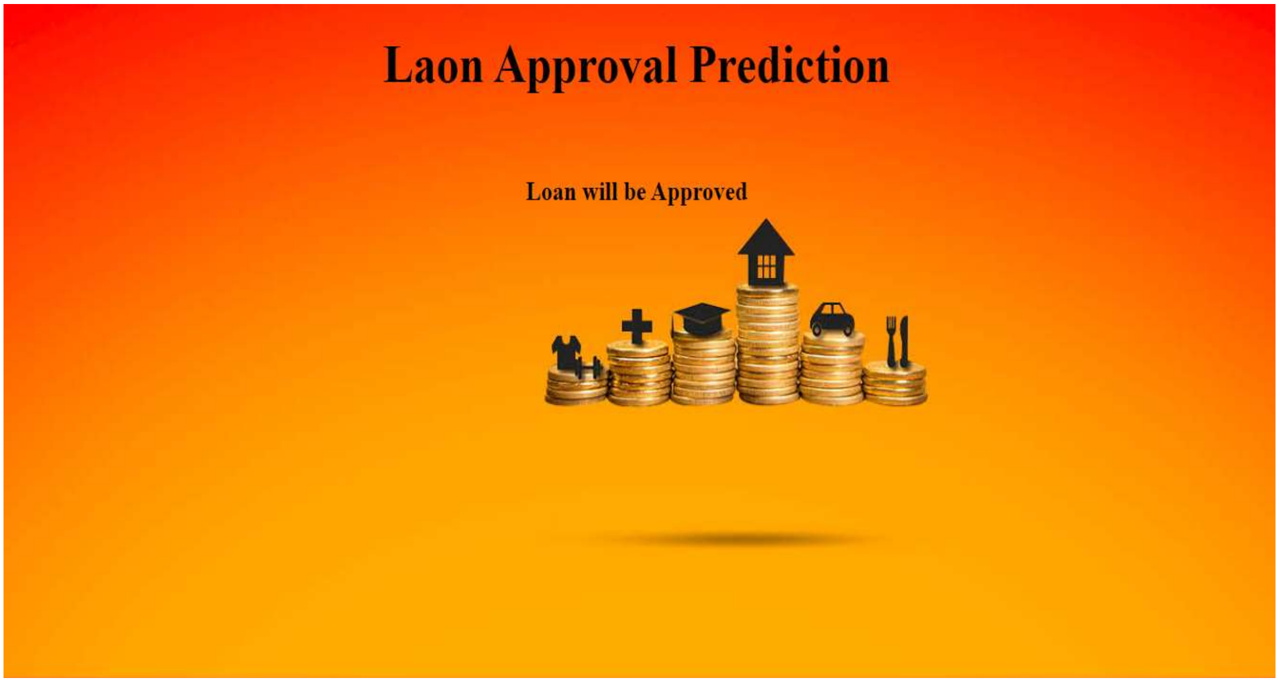
2

Property Area

Semi Urban

Input 1- Now, the user will give inputs to get the predicted result after clicking onto the submit button.

You will get the output.



Laon Approval Prediction

Loan will be Approved

CONCLUSION:

The predictive models based on Decision Tree and Random Forest, give the accuracy as 80.945%, 93.648% and 83.388% whereas the cross-validation is found to be 80.945%, 72.213% and 80.130% respectively. This shows that for the given dataset, the accuracy of model based on decision tree is highest but random forest is better at generalization even though it’s cross validation is not much higher than other models.