PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNINE

1.INTRODUCTION

1.1Overview

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

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

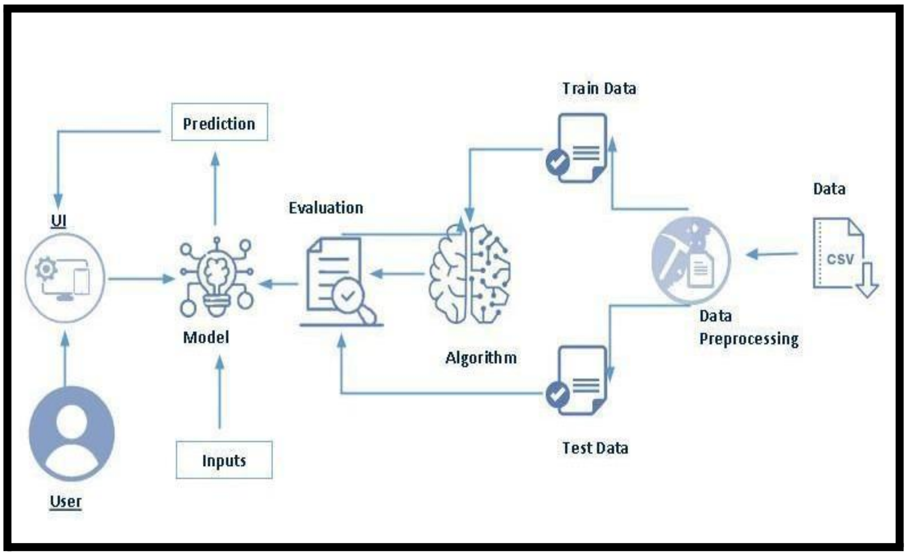
Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

1.2Purpose

Several collections of data from past loan applicants use different features to decide the loan status. A machine learning model can look at this data, which could be static or time-series, and give a probability estimate of whether this loan will be approved.

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..

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

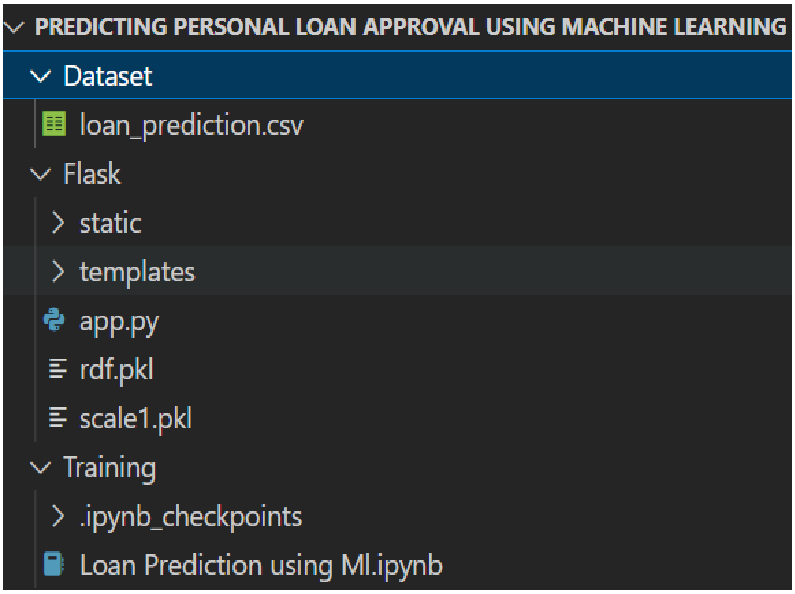
showcased on the UITo accomplish this, we have to

complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - 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:

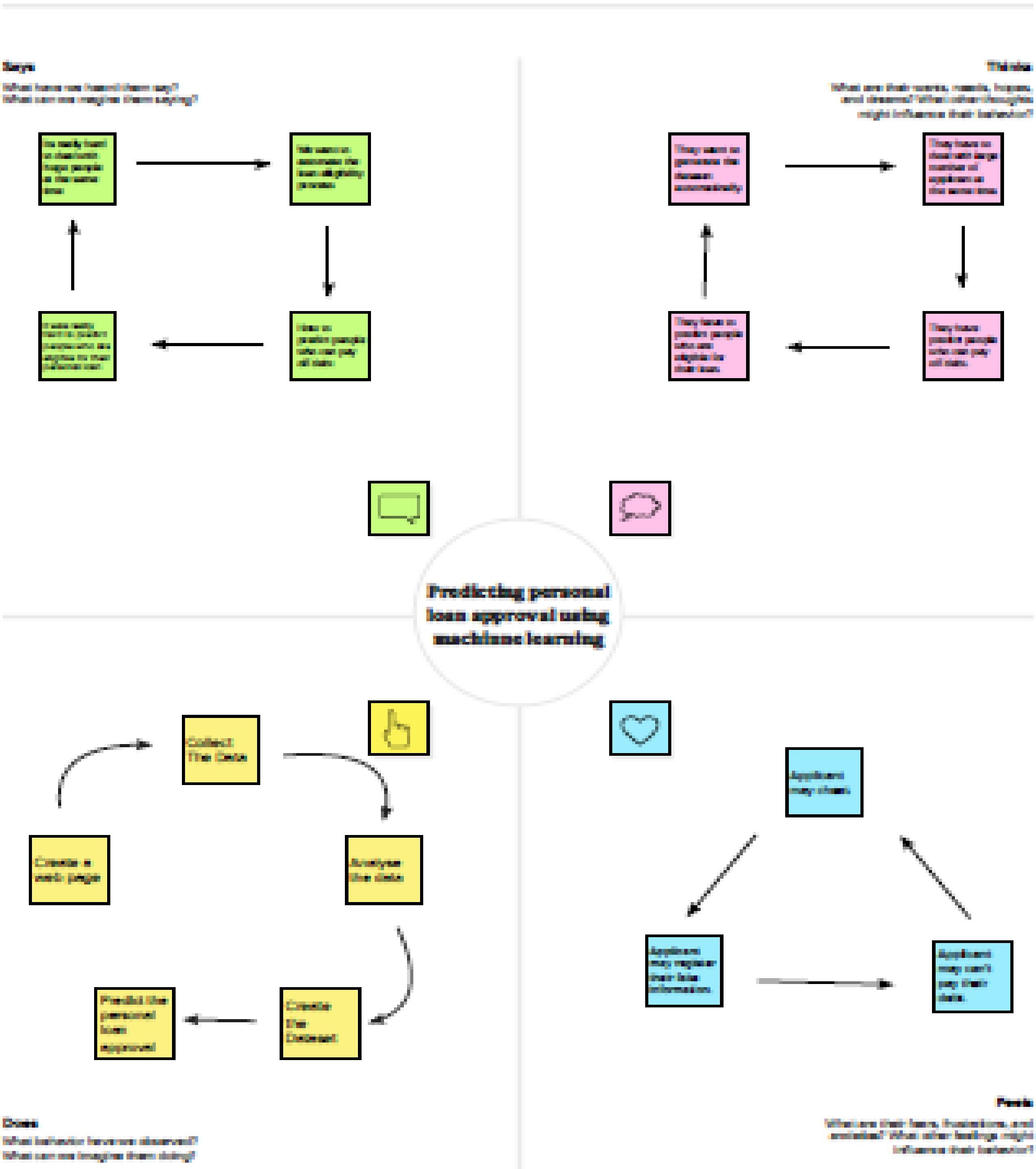
Create the Project folder which contains files as shown below



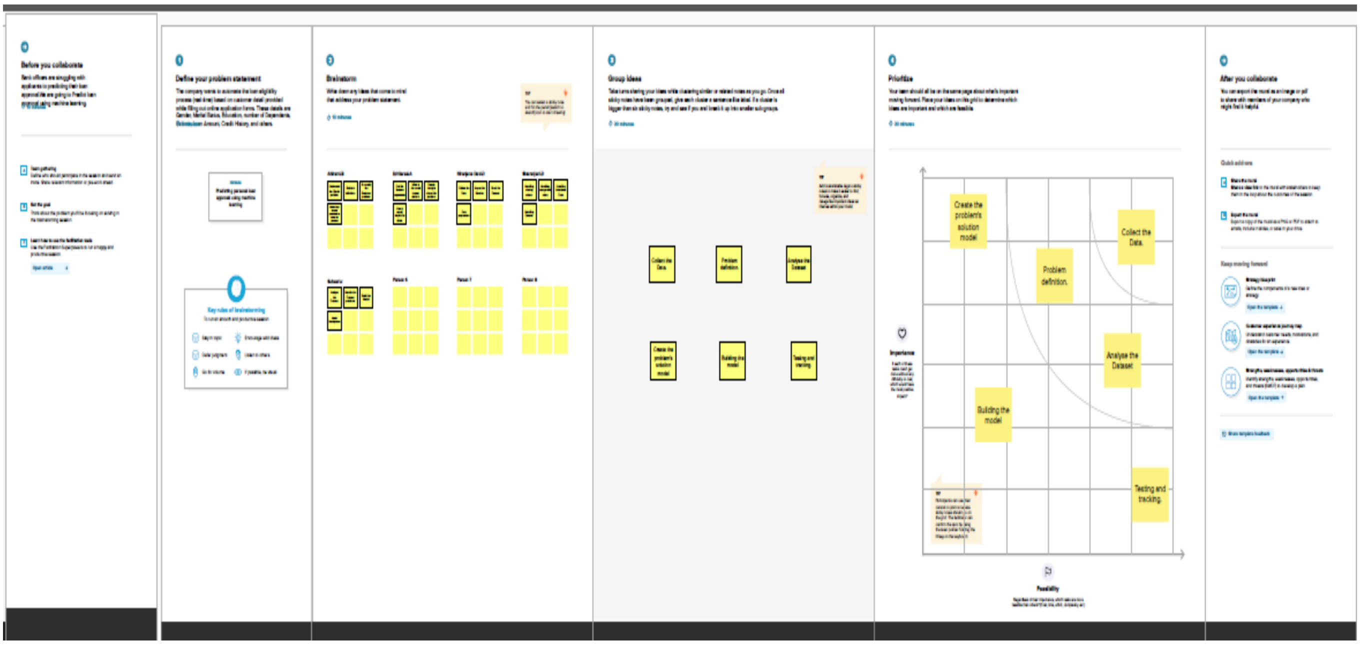
- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.

2.PROBLEM DEFINITION & DESIGN THINKING

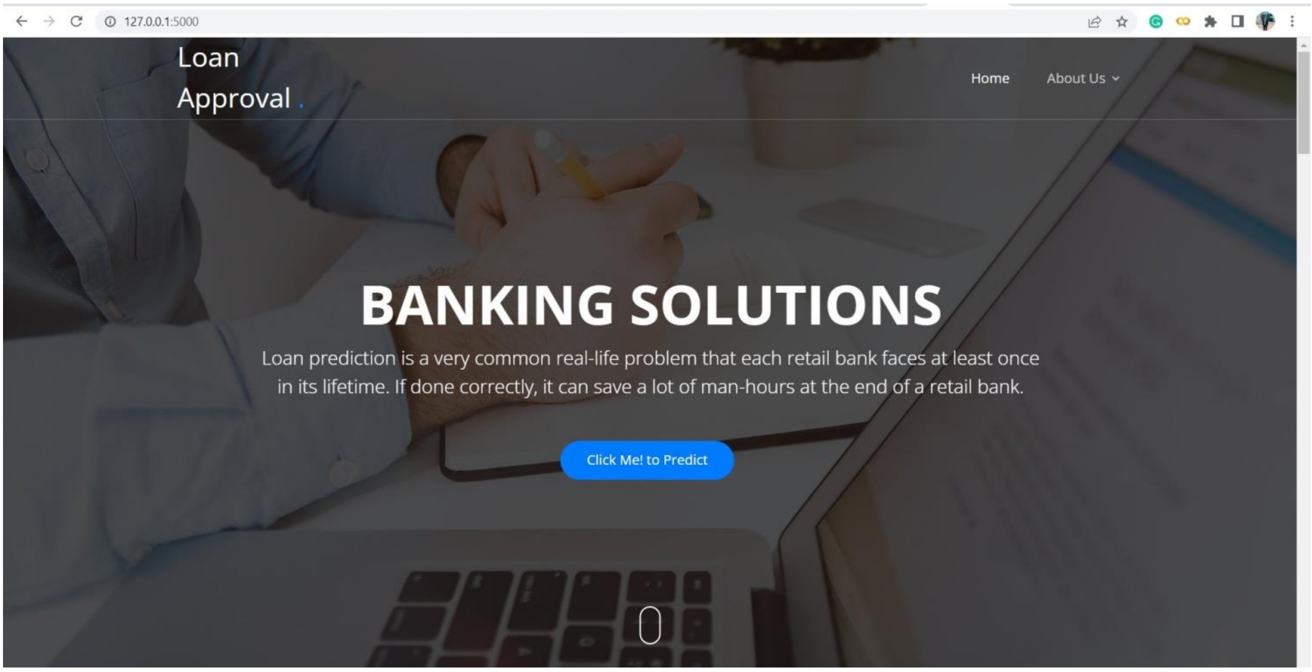
2.1 EMPATHY MAP



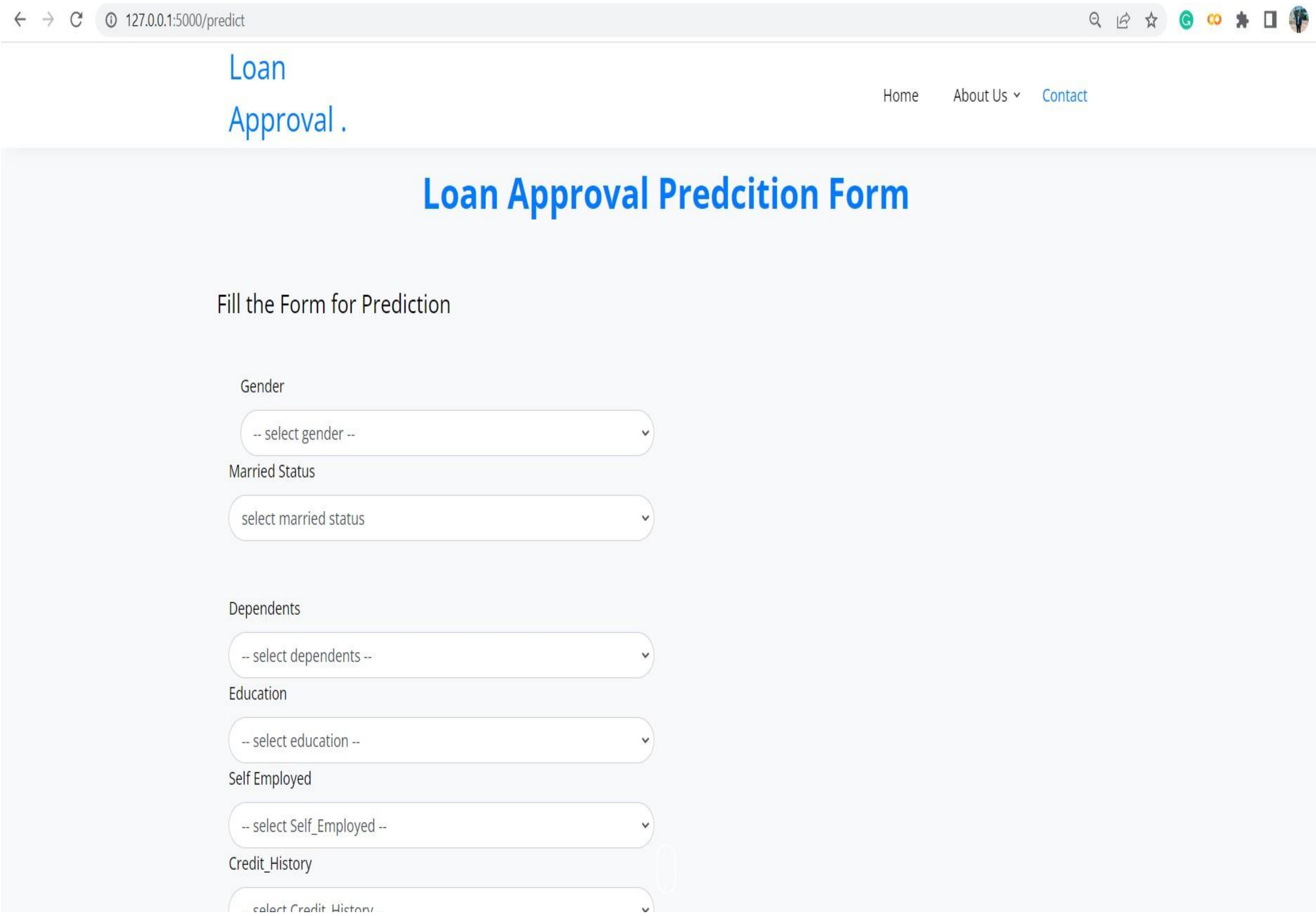
2.2 IDEATION AND BRAINSTRROMIUNG MAP



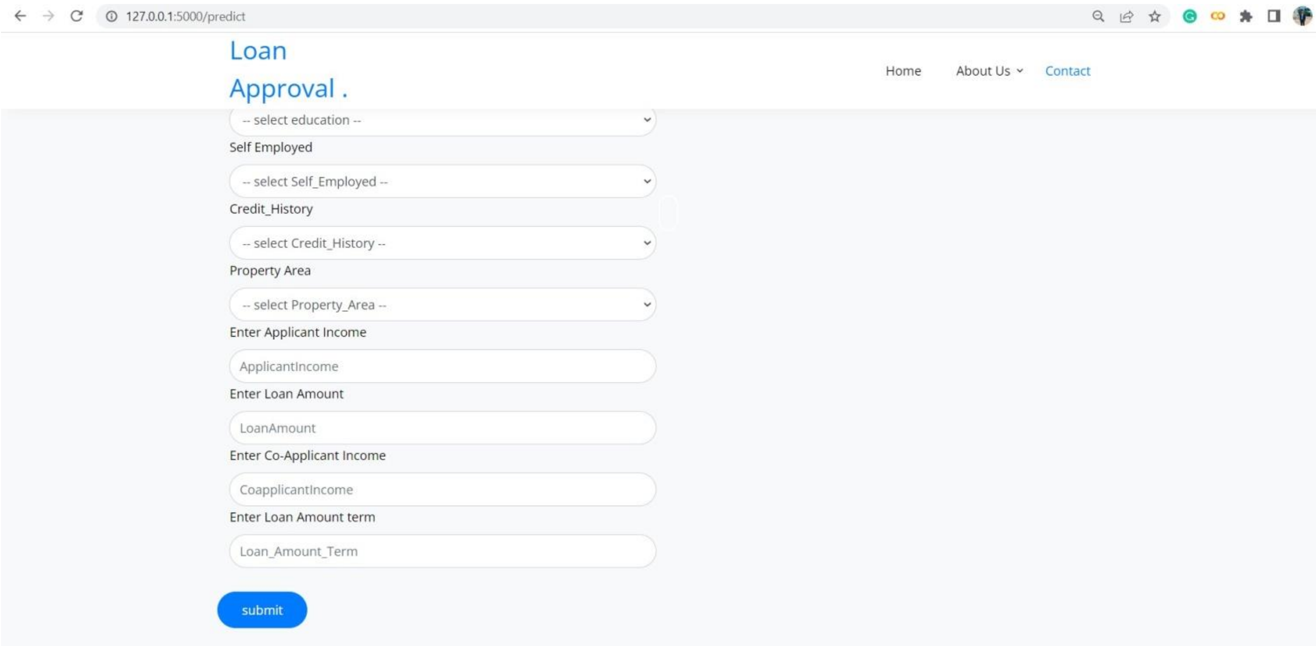
3.RESULT


127.0.0.1:5000
Loan
Approval .
Home
About Us
BANKING SOLUTIONS
Loan prediction is a very common real-life problem that each retail bank faces at least once in its lifetime. If done correctly, it can save a lot of man-hours at the end of a retail bank.
Click Me! to Predict

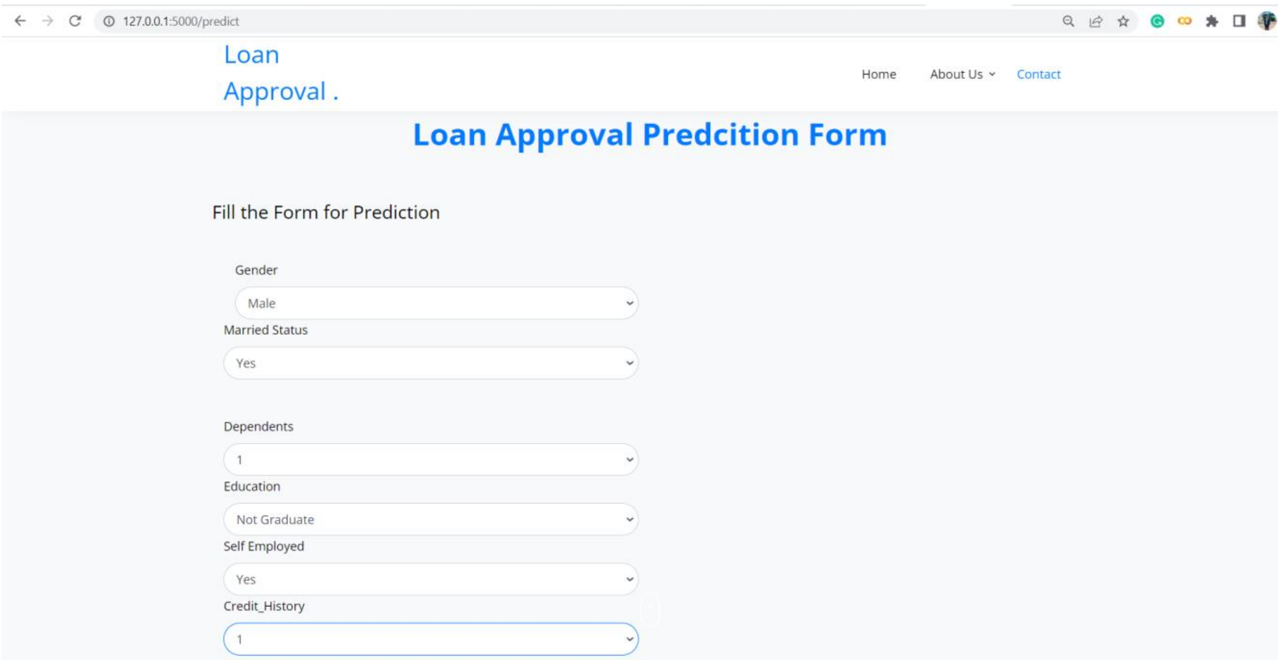
When you click on click me to predict the button from the banner you will get redirected to the prediction page.

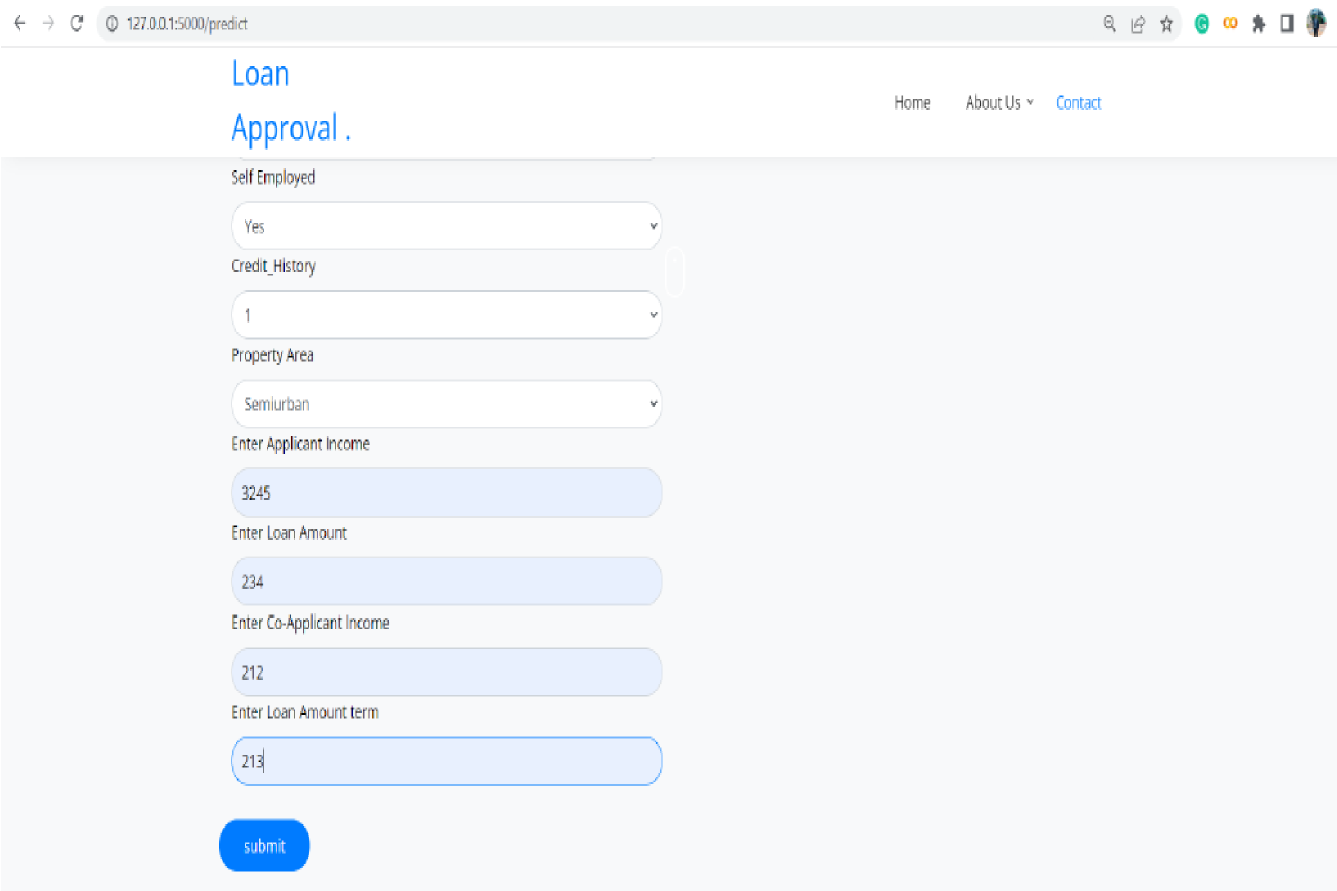


Input 1- Now, the user will give inputs to get the predicted result after clicking onto the submit button.

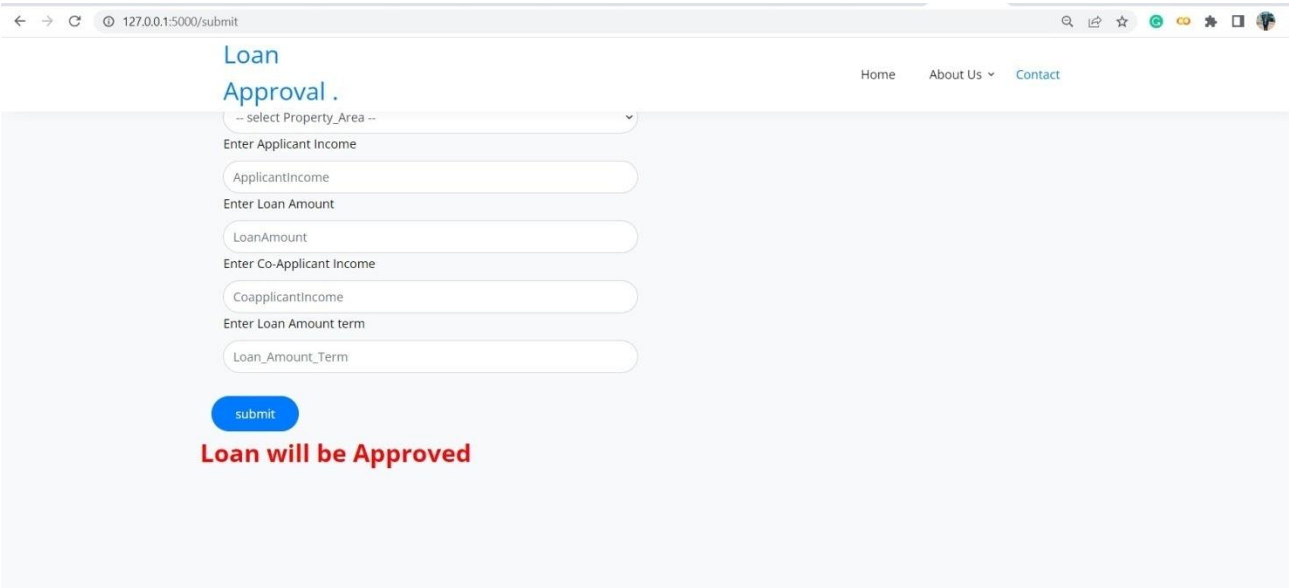


Input 1- Now, the user will give inputs to get the predicted result after clicking onto the submit button.




127.0.0.1:5000/predict
Loan
Approval .
Home
About Us
Contact
Self Employed
Yes
Credit_History
1
Property Area
Semiurban
Enter Applicant Income
3245
Enter Loan Amount
234
Enter Co-Applicant Income
212
Enter Loan Amount term
213
submit

Now when you click the submit button,you will get the result on the same page.



4.ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

Accuracy—one of the primary benefits of using machine learning for credit scoring is its accuracy. Unlike human manual processing, ML-based models are automated and less likely to make mistakes. This means that loan processing becomes not only faster but more accurate, too, cutting costs on the whole.

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.

DISADVANTAGES:

The disadvantage of this model is that it emphasize different weights to each factor but in real life sometime loan can be approved on the basis of single strong factor only, which is not possible through this system. Loan Prediction is very helpful
for employee of banks as well as for the applicant also.

5.APPLICATION:

Application of the predicting personal loan approval given below….

The goal of this project is to give an idea that how Reinforcement learning can be applied and how it can be used in Real-world applications such as self-driving cars (eg: AWS DeepRacer), training robots in the assembly line, and many more…

6.CONCLUSION:

So here, it can be concluded with confidence that the Naïve Bayes model is extremely efficient and gives a better result when compared to other models. It works correctly and fulfills all requirements of bankers. This system properly and accurately calculate the result. It predicts the loan is approve or reject to loan applicant or customer very accuratly.

7.FUTURE SCOPE:

This Project Work Can Be Extended To Higher Level In Future. For Example, A Predictive Model For Loans That Uses Machine Learning Algorithms, Where The Results From Each Graph Of The Project Can Be Taken As Individual Criteria For The Machine Learning Algorithm Can be Created. Also, A Risk Score Can Be Generated Based On Applicant To Predict Loan Default Rate.

8.APPENDIX

Source code:

DATA COLLECTION AND PREPARATION:

```
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, RandomForest
Classifier

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, confusi
on_matrix, f1_score
```

```
data = pd.read_csv('loan_prediction.csv')
data
```

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

```
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

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

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	0.0	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	0.0	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	0.0	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	0.0	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	0.0	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

```
data['Property_Area']=data['Property_Area'].map({'Urban':2,'Semiurban':1,'Rural':0})
data.head()
data['Married']=data['Married'].map({'Yes':1,'No':0})
data.head()
data['Education']=data['Education'].map({'Graduate':1,'Not Graduated':0})
data.head()
data['Self_Employed']=data['Self_Employed'].map({'Yes':1,'No':0})
data.head()
data['Loan_Status']=data['Loan_Status'].map({'Yes':1,'No':0})
data.head()
data.isnull().sum()
```

```
Gender                13
Married                3
Dependents            15
Education            134
Self_Employed         32
ApplicantIncome        0
CoapplicantIncome      0
LoanAmount            22
Loan_Amount_Term      14
Credit_History        50
Property_Area          0
Loan_Status          614
dtype: int64
```

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

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

```
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status        614
dtype: int64
```

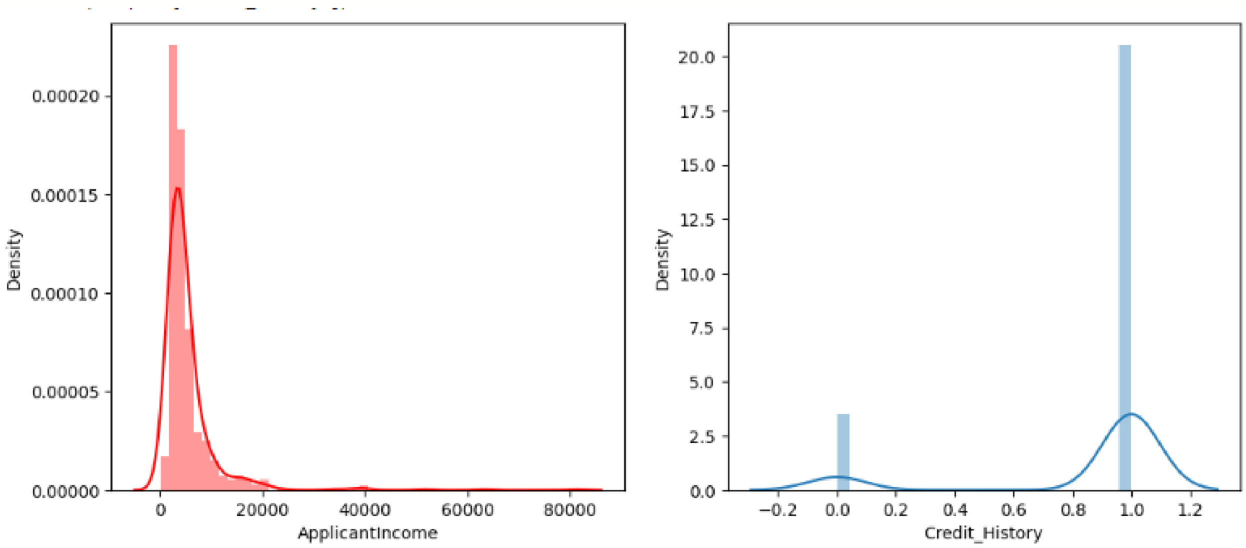
```
data.info()
```

```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             614 non-null    float64
 1   Married            614 non-null    float64
 2   Dependents         614 non-null    object
 3   Education          614 non-null    float64
 4   Self_Employed      614 non-null    float64
 5   ApplicantIncome    614 non-null    int64
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         614 non-null    float64
 8   Loan_Amount_Term   614 non-null    float64
 9   Credit_History     614 non-null    float64
 10  Property_Area      614 non-null    int64
 11  Loan_Status        0 non-null      float64
dtypes: float64(9), int64(2), object(1)
memory usage: 57.7+ KB
```

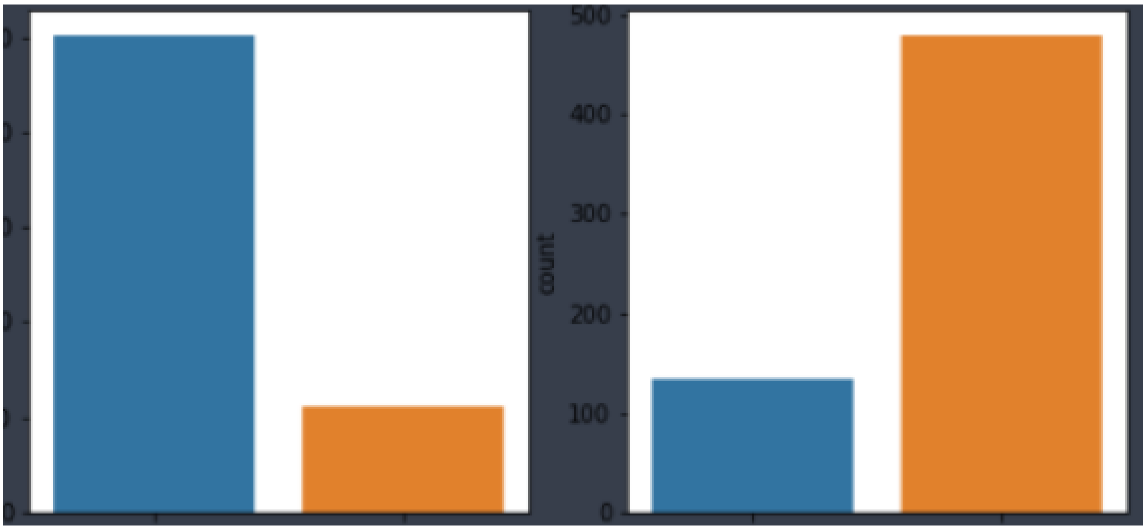
```
data['Gender']=data['Gender'].astype('int64')

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')
```

VISUAL ANALYSIS:

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




```
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()
```

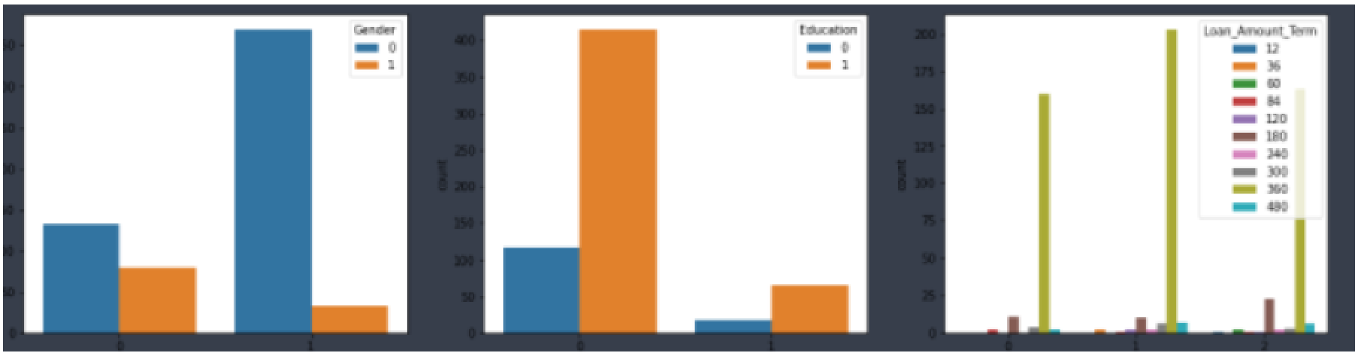


plt.figure(figsize=(20,5))

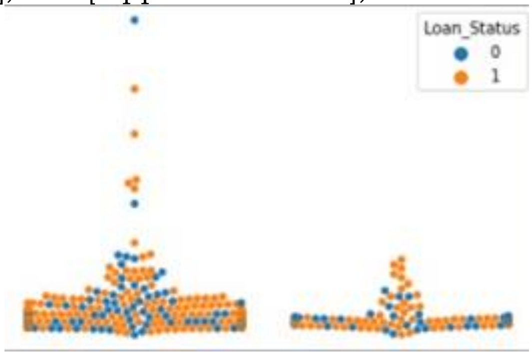
sns.countplot(data['Gender'], hue=data['Married'])

plt.subplot(133)

sns.countplot(data['Property_Area'], hue=data['Loan_Amount_Term'])



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



MODEL BUILDING:

```
def decisionTree(x_train, x_test, y_train, y_test)
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
y Pred = dt.predict(x_test)
print('***DecisionTreeClassifier***')
print('Confusionmatrix')
print(confusion_matrix(y_test,yPred))
print(classification_report(y_test,yPred))
print('Classification report')

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))

def KNN(x_train, x_test, y_train, y_test):
knn = KNeighborsClassifier()
yPred = knn.predict(x_test)
knn.fit(x_train,y_train)
print('***KNeighborsClassifier***')
print('Confusion matrix')
print(confusion_matrix(y_test,yPred))
print('Classification report')
print(classification_report(y_test,yPred))

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))

import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
classifier = Sequential()
classifier.add(Dense (units=100, activation='relu', input_dim=11))
classifier.add(Dense(units=50, activation='relu'))
classifier.add(Dense (units=1, activation='sigmoid'))
classifier.compile(optimizer='adam',                loss='binary_crossentropy',
metrics=['accuracy'])
```

Performance Testing & Hyperparameter Tuning

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

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

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
```