ARIGNAR ANNA GOVERNMENT ARTS COLLEGE

VILLUPURAM – 605 602.

DEPARTMENT OF COMPUTER APPLICATIONS

MACHINE LEARNING WITH PYTHON

Project Title : Predicting Personal Loan Approval Using Machine Learning

Team Id : NM2023TMID18759

Team Leader : KEERTHIVASAN A (73CD56066E3DF63B72A425055D5A0965)

Team member: PASUPATHI P (5070694B08EC52291454049B072F23EE)

Team member: PRASANNA D (197330FCA72FF44883F4D4C366F64403)

Team member: PRAVEEN RAJ A (AC7DE7FE0539DE1965343E67B7FB4676)

1. INTRODUCTION

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

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

1.1 Overview:

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

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

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

1.2 Purpose:

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

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

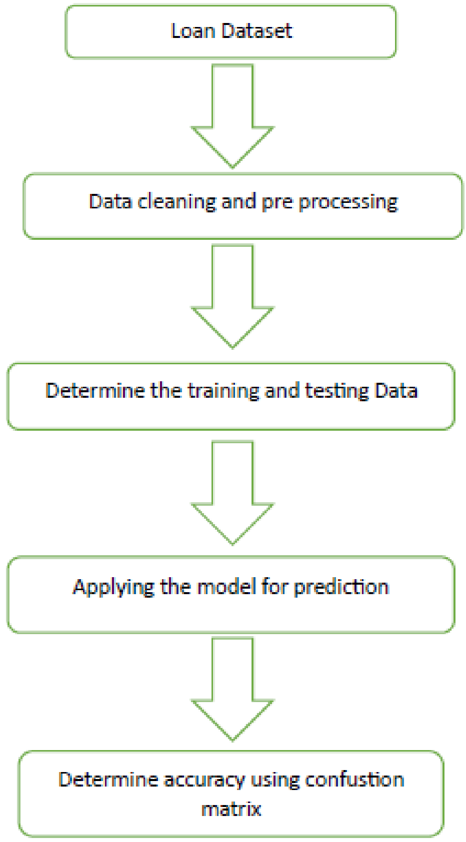
1.3. PREDICTIVE MODELLING

Predictive modeling is used to analyze the data and predict the outcome. Predictive modeling used to predict the unknown event which may occur in the future. In this process, we are going to create, test and validate the model.
There are different methods in predictive modeling. They are learning, artificial intelligence and statistics. Once we create a model, we can use many times, to determine the probability of outcomes.

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

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

1.4.PROCESS MODE USED:



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

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

1.5.Project Flow:

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

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

✓Define Problem / Problem Understanding

✓Data Collection & Preparation

✓Exploratory Data Analysis

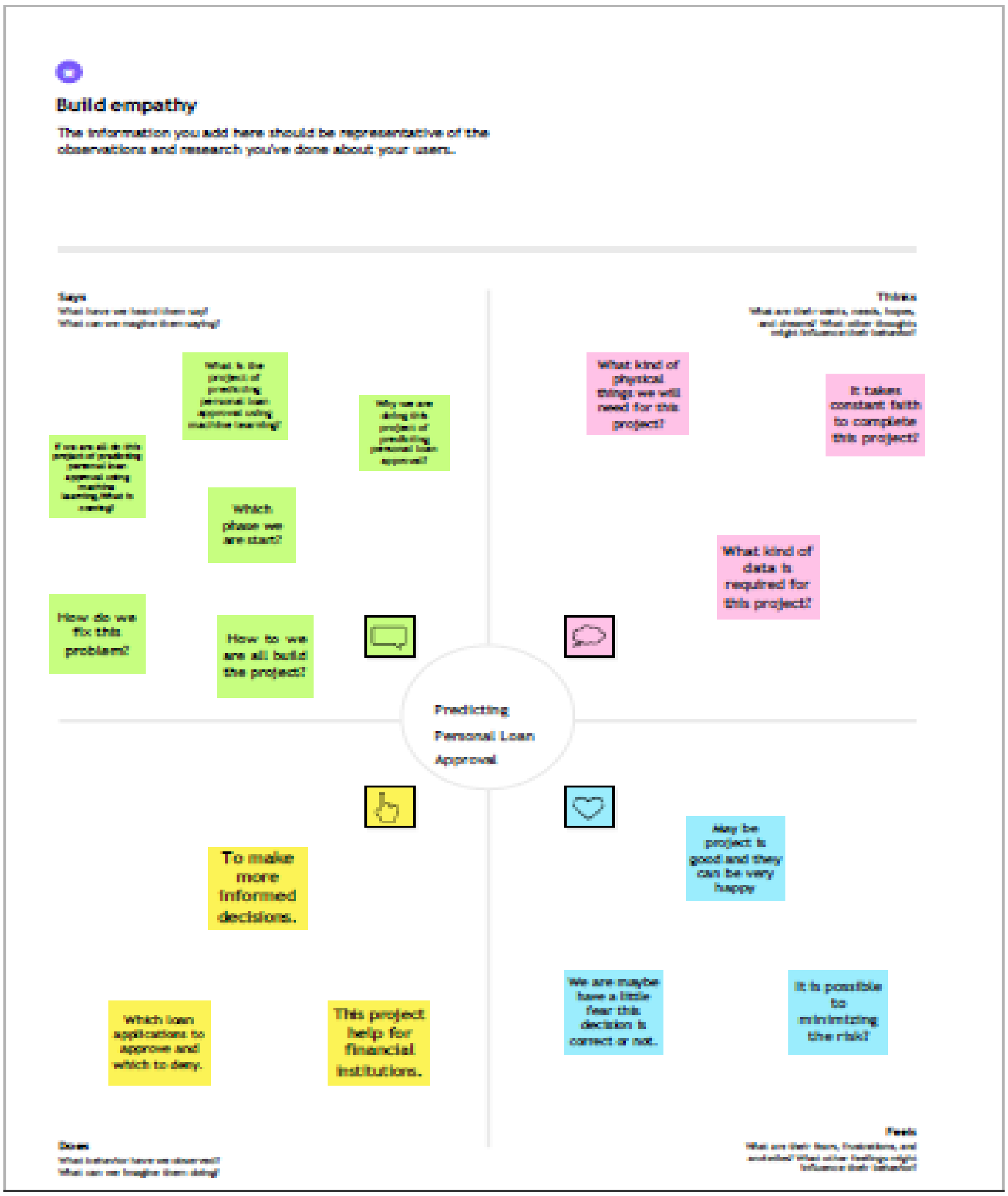
✓Model Building

✓Performance Testing & Hyperparameter Tuning

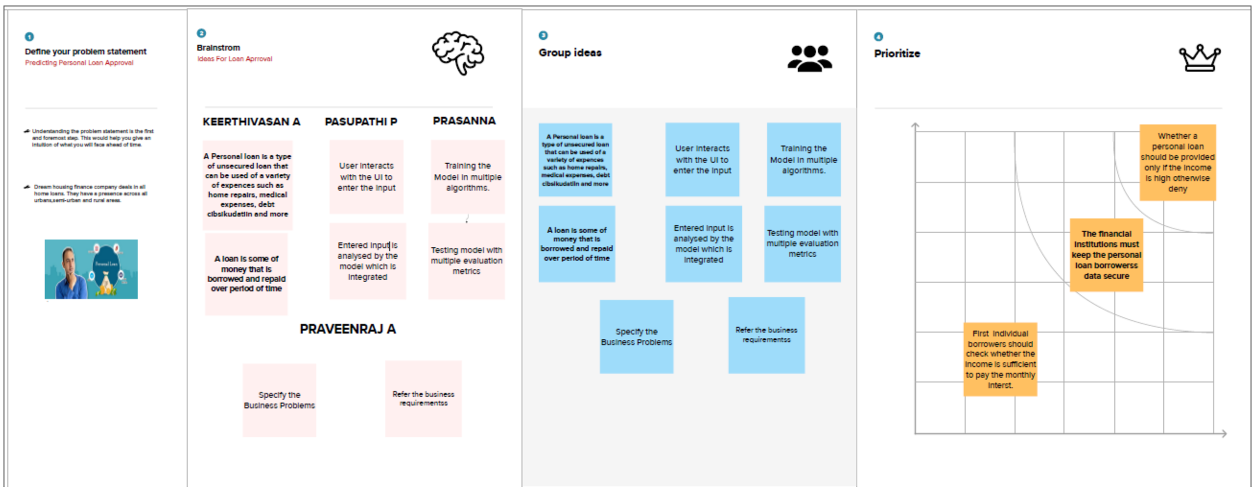
✓Model Deployment

2. PROBLEM DEFINITION AND DESIGN THINKING

2.1 Empathy Map:



2.2 Ideation & Brainstorming Map:



3. RESULT


Loan Approval Model
File | E:/Predicting%20Personal%20Loan%20Approval%20Using%20Machine%20Learning/Flask/home.html
Loan Approval Model
This model is created to predict if you qualify to get a certain amount of loan or not.
Gender Male Female
Are you married? Yes No
Number of dependents 1
Are you graduated? Yes No
Are you self-employed? Yes No
Applicant Income 3245
Coapplicant Income 212
Loan Amount 234
Loan Amount Term 213
Credit History All Debts paid
Property Area : Urban
Submit


File | E:/Predicting%20Personal%20Loan%20Approval%20Using%20Machine%20Learning/Flask/home.html
This model is created to predict if you qualify to get a certain amount of loan or not.
Gender Male Female
Are you married? Yes No
Number of dependents 1
Are you graduated? Yes No
Are you self-employed? Yes No
Applicant Income 3245
Coapplicant Income 212
Loan Amount 234
Loan Amount Term 213
Credit History All Debts paid
Property Area : Semiurban
Submit
"sorry! you are not eligible for loan"

4. ADVANTAGES AND DISADVANTAGES

ADVANTAGES :

✓The nonfunctional requirements ensure the software system follow legal and compliance rules.

✓They ensure the reliability, availability, and performance of the software system

✓They ensure good user experience and ease of operating the software.

✓They help in formulating security policy of the software system.

DISADVANTAGES :

✓None functional requirement may affect the various high-level software subsystem.

✓Technically you must be have a hardware requirements and that's ram almost 8 GB RAM.

5. APPLICATIONS

Here we will be using a declared constructor to route to the HTML page which we have created earlier. In, ‘/’ URL is bound with the home.html function. Hence, when the home pageof the web server is opened in the browser, the html page will be rendered. Whenever you enterthe values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

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

6. CONCLUSION

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

7. FUTURE SCOPE

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

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

8. APPENDIX

A. Source Code:

Milestone 2: Data Collection & Preparation

Importing the libraries:

```
#Importing the libraries
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
import warnings
warnings.filterwarnings('ignore')
```

Read the Dataset:

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

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	L
0	LP001002	Male	No	0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	
...	...	...	...	...	...	...	...	...	
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	

✓ 0s completed at 6:50 AM

Handling missing values:

```
df.info()
```

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

```
df.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
```

✓ 0s completed at 6:50 AM

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

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

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

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

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

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

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

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

Handling Categorical Values:

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

Handling Imbalance Data:

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

```
[12] smote=SMOTETomek()
```

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

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

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

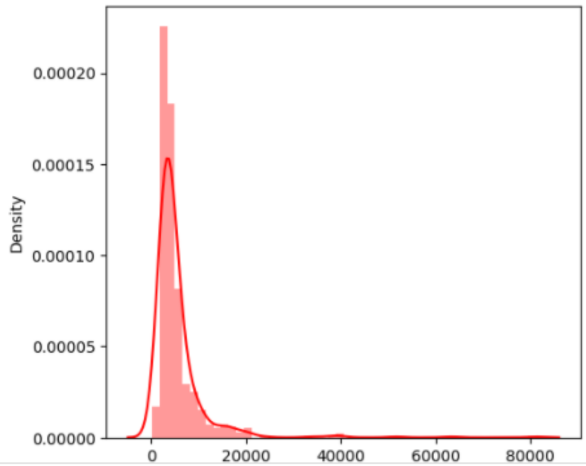
```
1    422
0    192
Name: Loan_Status, dtype: int64
1    352
0    352
Name: Loan_Status, dtype: int64
```

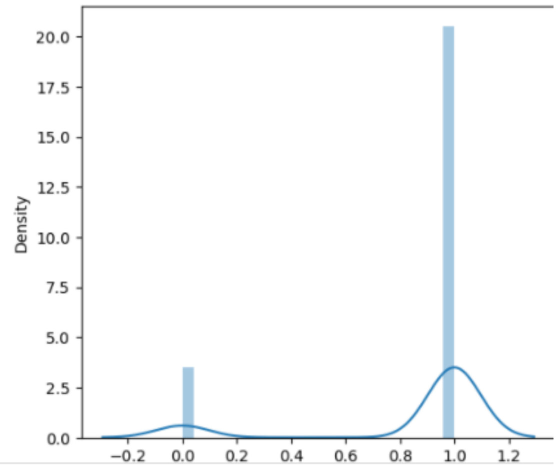
Exploratory Data Analysis:

```
df.describe()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Lo
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.00000	61
mean	0.182410	0.653094	0.744300	0.781759	0.133550	5403.459283	1621.24430	14
std	0.386497	0.476373	1.009623	0.413389	0.340446	6109.041673	2926.24876	8
min	0.000000	0.000000	0.000000	0.000000	0.000000	150.000000	0.00000	
25%	0.000000	0.000000	0.000000	1.000000	0.000000	2877.500000	0.00000	10
50%	0.000000	1.000000	0.000000	1.000000	0.000000	3812.500000	1188.50000	12
75%	0.000000	1.000000	1.000000	1.000000	0.000000	5795.000000	2297.25000	16
max	1.000000	1.000000	3.000000	1.000000	1.000000	81000.000000	41667.00000	70

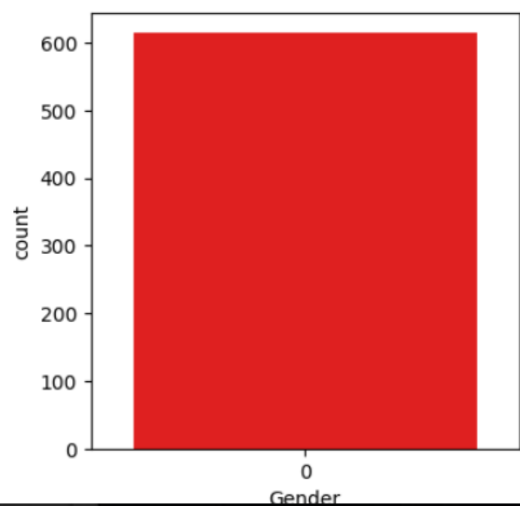
```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['ApplicantIncome'], color='r')
plt.subplot(122)
sns.distplot(df['Credit_History'])
plt.show()
```

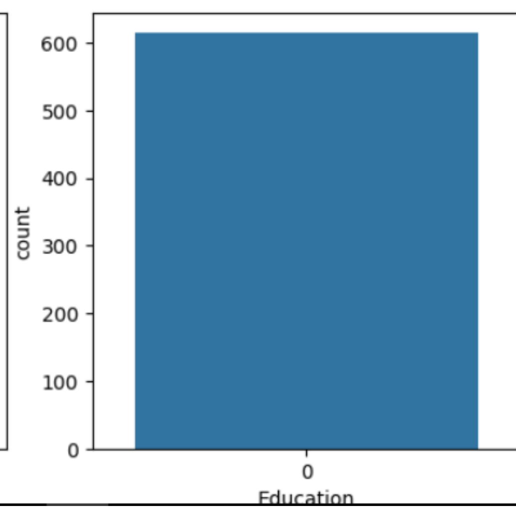




✓ 0s completed at 6:50 AM

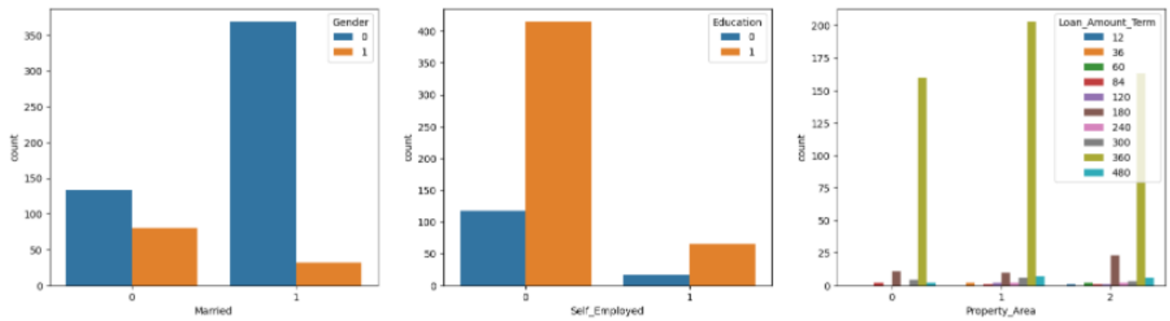
```
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(df['Gender'],color='r')
plt.xlabel('Gender')
plt.subplot(1,4,2)
sns.countplot(df['Education'])
plt.xlabel('Education')
plt.show()
```





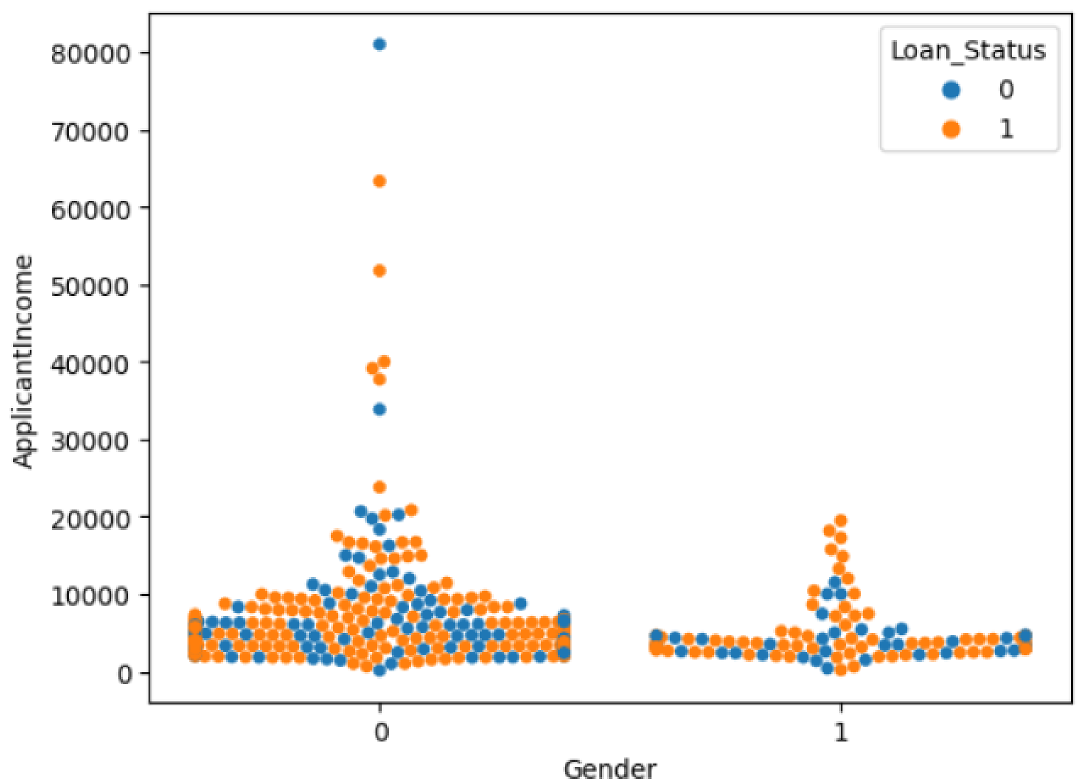
```python
plt.figure(figsize=(20,5))
plt.subplot(1,3,1)
sns.countplot(x = 'Married', hue="Gender", data = df)
plt.subplot(1,3,2)
sns.countplot(x = 'Self_Employed', hue="Education", data=df)
plt.subplot(1,3,3)
sns.countplot(x='Property_Area', hue="Loan_Amount_Term", data=df)
```

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




```python
sns.swarmplot(x = 'Gender', y='ApplicantIncome', hue="Loan_Status", data=df)
```

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



Scalling the Data:

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

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

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	-0.406894	-1.140462	-0.692132	0.618954	-0.316228	0.108147	-0.565620	-0.285487
1	-0.406894	0.876838	0.385879	0.618954	-0.316228	-0.109484	-0.011726	-0.177388
2	-0.406894	0.876838	-0.692132	0.618954	3.162278	-0.381608	-0.565620	-1.015154
3	-0.406894	0.876838	-0.692132	-1.615629	-0.316228	-0.453292	0.300483	-0.285487
4	-0.406894	-1.140462	-0.692132	0.618954	-0.316228	0.134105	-0.565620	-0.001727

```
[ ] x_train,x_test,y_train,y_test = train_test_split(x_bal,y_bal, test_size=0.33, random_state=42)
```

```
[ ] x_train.shape

(471, 11)
```

Model Building:

1.1 Decision Tree Model:

1.2 Random Forest Model:

```
#Milestone 4: Model Building
#decision tree model
def RandomForest(x_train,x_test,y_train,y_test):
  model = RandomForestClassifier()
  model.fit(x_train,y_train)
  y_tr = model.predict(x_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(x_test)
  print(accuracy_score(yPred,y_test))
  y_tr = model.predict(x_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(x_test)
  print(accuracy_score(yPred,y_test))
```

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

```
1.0
0.8412017167381974
1.0
0.8412017167381974
```

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

1.3 KNN model:

1.4 Xgboost model:

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

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

    0.8365180467091295
    0.7854077253218884
```

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

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

    0.9532908704883227
    0.8412017167381974
```

1.5 ANN model:

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

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

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

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

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

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

```
classifier.fit(x_train,y_train,batch_size=100, validation_split=0.2, epochs=1(
```

```
Epoch 73/100
4/4 [==============================] - 0s 21ms/step - loss: 0.2510 - accuracy: 0.8856 - val_loss: 0.4533
Epoch 74/100
4/4 [==============================] - 0s 25ms/step - loss: 0.2509 - accuracy: 0.8910 - val_loss: 0.4438
Epoch 75/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2480 - accuracy: 0.8963 - val_loss: 0.4565
Epoch 76/100
4/4 [==============================] - 0s 12ms/step - loss: 0.2455 - accuracy: 0.8883 - val_loss: 0.4630
Epoch 77/100
4/4 [==============================] - 0s 15ms/step - loss: 0.2456 - accuracy: 0.8910 - val_loss: 0.4579
Epoch 78/100
4/4 [==============================] - 0s 12ms/step - loss: 0.2419 - accuracy: 0.8963 - val_loss: 0.4687
Epoch 79/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2402 - accuracy: 0.8856 - val_loss: 0.4715
Epoch 80/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2395 - accuracy: 0.8803 - val_loss: 0.4761
Epoch 81/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2371 - accuracy: 0.8910 - val_loss: 0.4654
Epoch 82/100
4/4 [==============================] - 0s 19ms/step - loss: 0.2354 - accuracy: 0.8963 - val_loss: 0.4694
Epoch 83/100
4/4 [==============================] - 0s 14ms/step - loss: 0.2343 - accuracy: 0.8963 - val_loss: 0.4770
Epoch 84/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2318 - accuracy: 0.8856 - val_loss: 0.4782
Epoch 85/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2307 - accuracy: 0.8883 - val_loss: 0.4759
Epoch 86/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2287 - accuracy: 0.8936 - val_loss: 0.4815
Epoch 87/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2273 - accuracy: 0.8963 - val_loss: 0.4936
```

Testing the model:

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

array([0])
```

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

▾ RandomForestClassifier
RandomForestClassifier()
```

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

              precision    recall  f1-score   support

           0       0.77      0.78      0.77       114
           1       0.79      0.77      0.78       119

    accuracy                           0.78       233
   macro avg       0.78      0.78      0.78       233
weighted avg       0.78      0.78      0.78       233
```

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

array([0])
```

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

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

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

```
▾ GradientBoostingClassifier
GradientBoostingClassifier()
```

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

```
              precision    recall  f1-score   support

           0       0.77      0.78      0.77       114
           1       0.79      0.77      0.78       119

    accuracy                           0.78       233
   macro avg       0.78      0.78      0.78       233
weighted avg       0.78      0.78      0.78       233
```

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

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

```
y_pred = classifier.predict(x_test)
```

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

```
y_pred
```

```
[3.4216109e-01],
[8.6666912e-01],
[3.5088898e-03],
[4.5950827e-01],
[6.5072119e-01],
[8.7764539e-02],
[7.4654257e-01],
[2.3753060e-01],
[2.1284377e-05],
[9.3427205e-01],
[6.0837412e-01],
[6.9507974e-04],
[8.9078701e-01],
[8.7701297e-01],
[9.6149690e-04],
[3.4735101e-01],
[4.7861549e-01],
[9.7111917e-01],
[2.2641063e-04],
[5.7968706e-01],
[1.6246670e-03],
[7.9318714e-01],
[2.3815326e-01],
[4.0762681e-01],
```

```
def predict_exit(sample_value):
  sample_value = np.array(sample_value)
  sample_value = sample_value.reshape(1, -1)
  sample_value = sc.transform(sample_value)
  return classifier.predict(sample_value)
```

```
[ ] sample_value = [[1,1,0,1,1,4276,1542,145,240,0,1]]
    if predict_exit(sample_value)>0.5:
      print('Prediction: High Chance of Loan Approval!')
    else:
      print('Prediction: Low Chance of Loan Approval!')
```

```
1/1 [==============================] - 0s 62ms/step
Prediction: Low Chance of Loan Approval!
```

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

```
1/1 [==============================] - 0s 21ms/step
Prediction: High Chance of Loan Approval!
```

Perfomance Testing & Hyperparameter Tuning:

1.1: Compare the model:

```
#Milestone 5: Performance Testing & Hyperparameter Tuning
def compareModel(x_train,x_test,y_train,y_test):
  decisionTree(x_train,x_test,y_train,y_test)
  print('-'*100)
  RandomForest(x_train,x_test,y_train,y_test)
  print('-'*100)
  XGB(x_train,x_test,y_train,y_test)
  print('-'*100)
  KNN(x_train,x_test,y_train,y_test)
  print('-'*100)
```

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

```
1.0
0.8454935622317596
1.0
0.8454935622317596
              precision    recall  f1-score   support

           0       0.77      0.78      0.77       114
           1       0.79      0.77      0.78       119

    accuracy                           0.78       233
   macro avg       0.78      0.78      0.78       233
weighted avg       0.78      0.78      0.78       233
```

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

```
0.9532908704883227
0.8412017167381974
              precision    recall  f1-score   support

           0       0.77      0.78      0.77       114
           1       0.79      0.77      0.78       119

    accuracy                           0.78       233
   macro avg       0.78      0.78      0.78       233
weighted avg       0.78      0.78      0.78       233
```

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

```
0.8365180467091295
0.7854077253218884
              precision    recall  f1-score   support

           0       0.77      0.78      0.77       114
           1       0.79      0.77      0.78       119

    accuracy                           0.78       233
   macro avg       0.78      0.78      0.78       233
weighted avg       0.78      0.78      0.78       233
```

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

```
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
```

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

```
8/8 [==============================] - 0s 2ms/step
0.4892703862660944
ANN Model
Confusion_Matrix
[[114   0]
 [119   0]]
Classification Report
              precision    recall  f1-score   support

           0       0.49      1.00      0.66       114
           1       0.00      0.00      0.00       119

    accuracy                           0.49       233
   macro avg       0.24      0.50      0.33       233
weighted avg       0.24      0.49      0.32       233
```

```
from sklearn.model_selection import cross_val_score
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
ypred = rf.predict(x_test)
```

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

```
0.8420857465954698
```

```
[ ] cv = cross_val_score(rf,x,y,cv=5)
```

```
[ ] np.mean(cv)
```

```
0.7784886045581768
```

```
8/8 [==============================] - 0s 2ms/step
0.4892703862660944
XGB
Confusion_Matrix
[[114   0]
 [119   0]]
Classification Report
              precision    recall  f1-score   support

           0       0.49      1.00      0.66       114
           1       0.00      0.00      0.00       119

    accuracy                           0.49       233
   macro avg       0.24      0.50      0.33       233
weighted avg       0.24      0.49      0.32       233
```

Model Deployment:

```
import pickle
pickle.dump(KNN,open("rdf.pkl",'wb'))
model=pickle.load(open('rdf.pkl','rb'))
```

Building Html pages:

Build Python code:

rminal Help • app.py - Predicting Personal Loan Approval Using Machine Learning - Visual Studio Code

app.py 2 • home.html

flask > app.py > ...

```python
from flask import Flask, render_template, request
import pickle
import numpy as np
model = pickle.load(open('rdf.pkl','rb'))  # opening pickle file in read mode
app = Flask(__name__, template_folder='template')  # initializing Flask app
@app.route("/")
def hello():
    return render_template('home.html')
@app.route("/predict", methods=['POST'])
def predict():
     if request.method == 'POST':
         d1 = request.form['Married']
         if d1 == 'No':
             d1 = 0
         else:
             d1 = 1
         d2 = request.form['Gender']
         if (d2 == 'Male'):
             d2 = 1
         else:
             d2 = 0
         d3 = request.form['Education']
         if (d3 == 'Graduate'):
             d3 = 1
         else:
             d3 = 0
         d4 = request.form['Self_Employed']
         if (d4 == 'No'):
             d4 = 0
         else:
             d4 = 1
         d5 = request.form['ApplicantIncome']
```

Help • app.py - Predicting Personal Loan Approval Using Machine Learning - Visual Studio Code

p.py 2 ● home.html

app.py > ...

```python
        d5 = request.form['ApplicantIncome']
        d6 = request.form['CoapplicantIncome']
        d7 = request.form['LoanAmount']
        d8 = request.form['Loan_Amount_Term']
        d9 = request.form['Credit_History']
        if (d9 == 'All debts paid'):
            d9 = 1
        else:
            d9 = 0
        d10 = request.form['Property_Area']
        if (d10 == 'Urban'):
            d10 = 2
        elif (d10 == 'Rural'):
            d10 = 0
        else:
            d10 = 1
        d11 = request.form['Dependents']
        if (d11 == '3+'):
            d11 = 3
        elif (d11=='2'):
            d11 = 2
        elif (d11=='1'):
            d11 = 1
        else:
            d11 = 0
        arr = np.array([[d1, d2, d3, d4, d5, d6, d7, d8, d9,d10,d11]])
        pred = model.predict(arr)
        if pred == 0:
            ans = "Sorry! You are not eligible for Loan."
        else:
            ans = "Congrats! You are eligible for Loan."
```

```python
        return render_template('home.html', prediction_text = ans)
app.run(host="0.0.0.0")            # deploy
app.run(debug=True)                # run on local system
if __name__ == '__main__':
    app.run(debug = True)
```

home.html:

```html
<!DOCTYPE html>

<html>
<style>
    body
    {
        background-image: url('https://wallpaperaccess.com/download/loan-4105994');
    }
</style>
<head>

    <meta charset="UTF-8">

    <title>Loan Approval Model</title>

</head>

<body>

<div class="login">

    <div class="banner">

        <h1 style="text-align: center;">Loan Approval Model</h1>

    </div>

    <p style="text-align: center;"> This model is created to predict if you qualify to get a certain amount of l

    </p>

    <br> <br>
```

app.py   home.html

lask > home.html > html > head > meta

```html

    <form action="{{ url_for('predict')}}" method="post">

        <label for="Gender">Gender </label>

        <label class="radio-inline">

            <input type="radio" name="Gender">Male

        </label>

        <label class="radio-inline">

            <input type="radio" name="Gender">Female

        </label>

        <br><br>

        <label for="Married"> Are you married? </label>

        <label class="radio-inline">

            <input type="radio" name="Married">Yes

        </label>

        <label class="radio-inline">

            <input type="radio" name="Married">No

        </label>
```

app.py home.html

flask > home.html > html > head > meta

```html
        <br><br>

        <label for="Dependents"> Number of dependents </label>

        <select id="Dependents" name="Dependents">

            <option disabled selected value> -- Select an option -- </option>

            <option value="0">0</option>

            <option value="1">1</option>

            <option value="2">2</option>

            <option value="3+">3+</option>

        </select>

        <br><br>

        <label for="Education"> Are you graduated? </label>

        <label class="radio-inline">

            <input type="radio" name="Education">Yes

        </label>

        <label class="radio-inline">

            <input type="radio" name="Education">No

```

app.py home.html

flask > home.html > html > head > meta

```html
</label>

<br><br>

<label for="Self_Employed"> Are you self-employed? </label>

<label class="radio-inline">

    <input type="radio" name="Self_Employed">Yes

</label>

<label class="radio-inline">

    <input type="radio" name="Self_Employed">No

</label>

<br><br>

<label for="ApplicantIncome">Applicant Income</label>

<input type="text" name="ApplicantIncome" required="required" />

<br><br>

<label for="CoapplicantIncome">Coapplicant Income</label>

<input type="text" name="CoapplicantIncome" required="required" />

<br><br>
```

app.py · home.html

flask > home.html > html > head > meta

```html
            <label for="Loan_Amount_Term">Loan Amount Term</label>

            <input type="text" name="Loan_Amount_Term" placeholder="Term in days" required="required" />

            <br><br>

            <label for="Credit_History">Credit History</label>

            <select id="Credit_History" name="Credit_History">

                <option disabled selected value> -- Select an option -- </option>

                <option value="All Debts paid">All Debts paid</option>

                <option value="Not paid">Not paid</option>

            </select>

            <br><br>

            <label for="Property_Area">Property Area</label> ;

            <select id="Property_Area" name="Property_Area">

                <option disabled selected value> -- Select an option -- </option>

                <option value="Rural">Rural</option>

                <option value="Semiurban">Semiurban</option>

                <option value="Urban">Urban</option>

            </select>

        <br><br>

            <input type="submit", value = "Submit">

        </form>
        <h3>{{ prediction_text }}</h3>

    </div>

</body>

</html>
```

Output:


Loan Approval Model
E:/Predicting%20Personal%20Loan%20Approval%20Using%20Machine%20Learning/Flask/home.html
Loan Approval Model
This model is created to predict if you qualify to get a certain amount of loan or not.
Gender Male Female
Are you married? Yes No
Number of dependents 1
Are you graduated? Yes No
Are you self-employed? Yes No
Applicant Income 3245
Coapplicant Income 212
Loan Amount 234
Loan Amount Term 213
Credit History All Debts paid
Property Area : Urban
Submit


E:/Predicting%20Personal%20Loan%20Approval%20Using%20Machine%20Learning/Flask/home.html
This model is created to predict if you qualify to get a certain amount of loan or not.
Gender Male Female
Are you married? Yes No
Number of dependents 1
Are you graduated? Yes No
Are you self-employed? Yes No
Applicant Income 3245
Coapplicant Income 212
Loan Amount 234
Loan Amount Term 213
Credit History All Debts paid
Property Area : Semiurban
Submit
"sorry! you are not eligible for loan"