

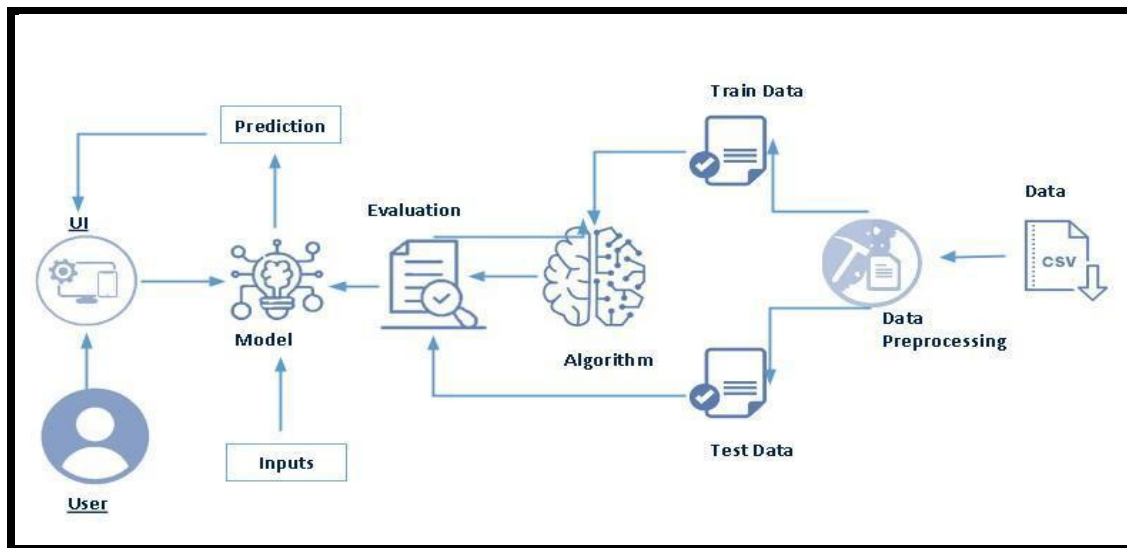
PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING

1.INTRODUCTION

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 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 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.

Technical Architecture



1.2 PURPOSE

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**.

2. PROBLEM DEFINITION AND DESIGN THINKING

2.1 EMPATHY MAP



2.2 IDEATION & BRAINSTORMING MAP

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions as your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 1. Brainstorm concepts
- 2. Prioritize ideas
- 3. Develop a concept

Before you collaborate

1. Make list of population you're trying to solve. Think about what you need to do to get going.

2. Minutes

Define your problem statement

What problem are you trying to solve? Frame your problem as a clear, specific statement. You will be the focus of your brainstorm.

3. Minutes

PROBLEM

Having personal space and privacy while working is essential for productivity and well-being. However, in a shared office environment, it's difficult to find a quiet space to work. This problem statement aims to explore ways to create a more productive and comfortable workspace for individuals in a shared office environment.

Key takeaways from ideation

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

Brainstorm

Write down any ideas that come to mind that address your problem statement.

4. Minutes

Brainstorming ideas

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

Group ideas

Take time during your idea sharing session to cluster ideas as you go. Group all ideas that have been proposed, give each cluster a name for the idea. A cluster is larger than a single idea, but not as big as a group of ideas.

5. Minutes

Grouping ideas

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

Prioritize

You have shared all the ideas you have and it's time to prioritize. Rank each idea based on its impact and feasibility. Rank each idea based on its impact and feasibility.

6. Minutes

Prioritizing ideas

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

After you collaborate

You can expect to reach a consensus on a few ideas and members of your company who might find it helpful.

7. Minutes

After collaborating

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

Brainstorming ideas

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

Prioritizing ideas

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

Developing a concept

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

Final concept

1. Brainstorming ideas

2. Prioritizing ideas

3. Developing a concept

3.RESULT

The image displays two sequential screenshots of a Google Colaboratory notebook interface, showing the process of loading and inspecting a dataset.

Top Screenshot: The notebook is titled "Untitled1.ipynb". The code cell contains the command `data = pd.read_csv('loan_prediction.csv')`. The output shows a preview of the data as a table with 10 columns: Loan_ID, Gender, Married, Dependents, Education, Self_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount, and Loan_Status. The first few rows are visible, showing data for Loan_IDs 0 through 4, followed by an ellipsis, and then rows 609 through 612.

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

Bottom Screenshot: The same notebook is shown, but the code cell now contains `data.info()`. The output provides detailed information about the DataFrame, including the number of entries (614), the total number of columns (13), and the data types for each column. The output also shows the memory usage (62.5+ KB).

```
<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
```

Untitled1.ipynb - Colaboratory

colab.research.google.com/drive/1wqf8ezZrKH0xtStKg3qTdggPipj1_jkv#scrollTo=8RXi8hsxcUvy

To get future Google Chrome updates, you'll need Windows 10 or later. This computer is using Windows 7.

Untitled1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- sample_data
- loan_prediction.csv

Code

```
[15] 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
```

0s completed at 2:09 PM

Untitled1.ipynb - Colaboratory

colab.research.google.com/drive/1wqf8ezZrKH0xtStKg3qTdggPipj1_jkv#scrollTo=KmUP3M-ycBqg

To get future Google Chrome updates, you'll need Windows 10 or later. This computer is using Windows 7.

Untitled1.ipynb

File Edit View Insert Runtime Tools Help

Files

- sample_data
- loan_prediction.csv

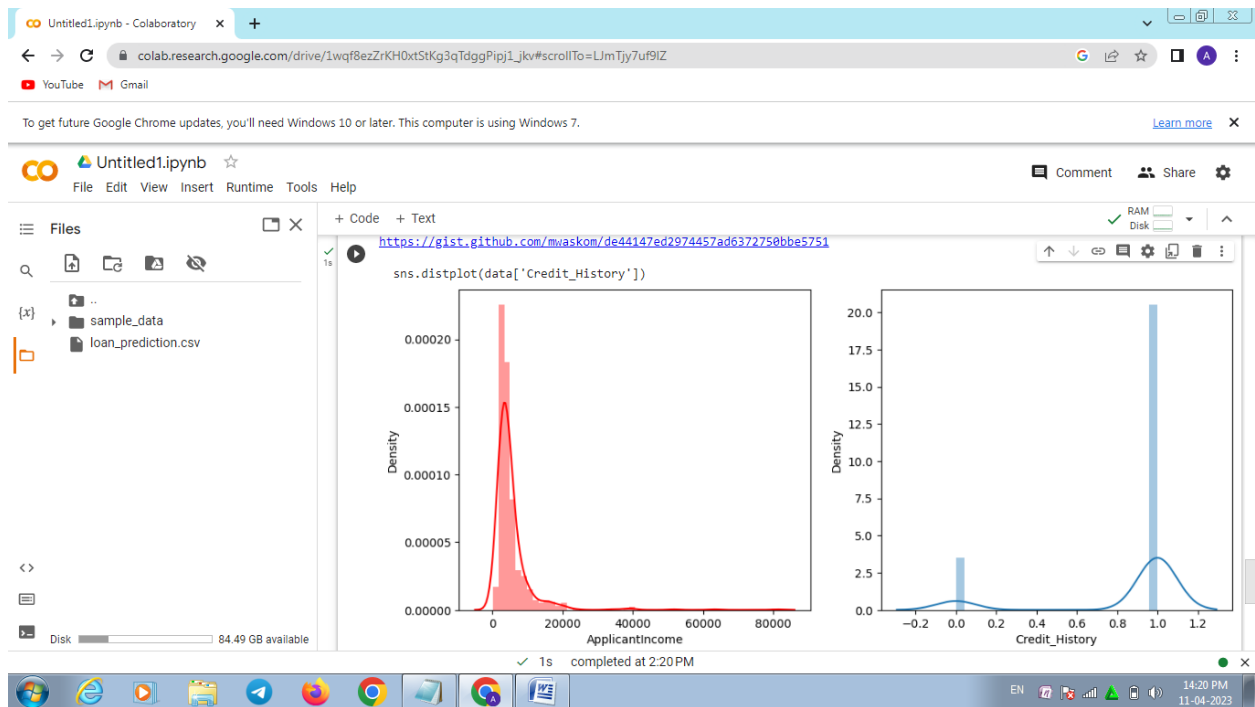
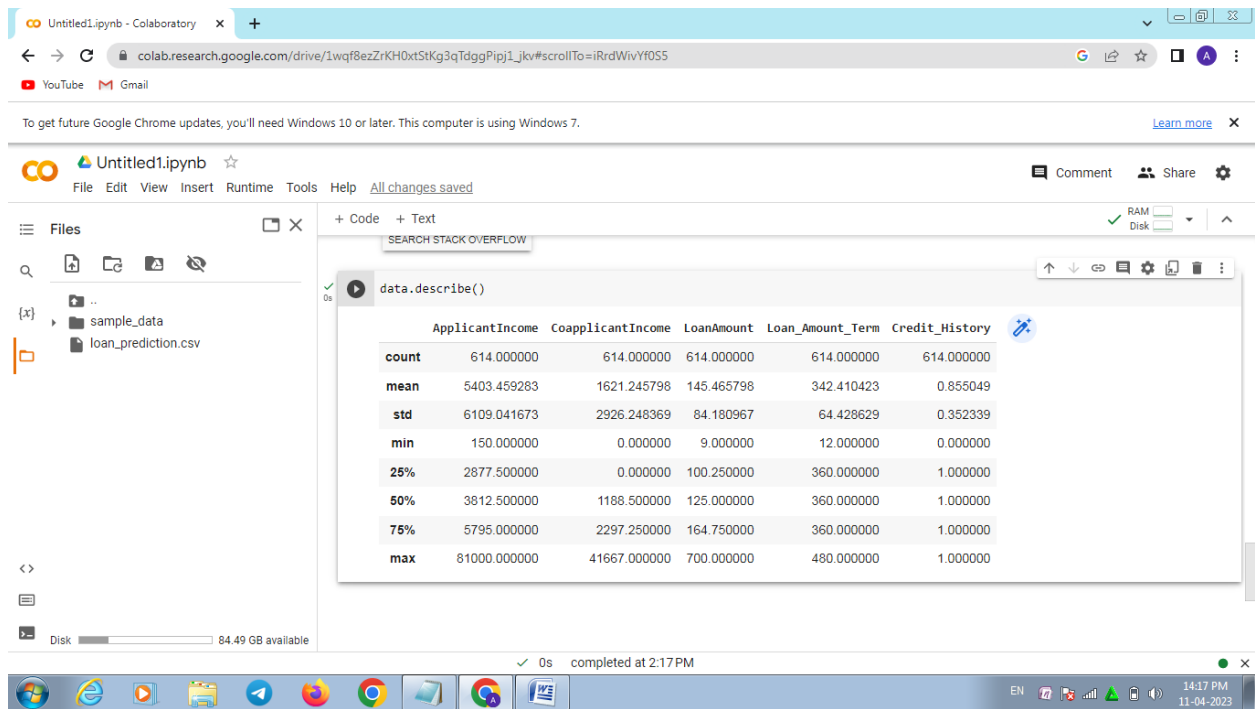
Code

```
[15] 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

data.isnull().sum()

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

0s completed at 2:00 PM



4. ADVANTAGES & 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

- **DISADVANTAGES**

- 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. APPLICATIONS

- In banking sector.
- Co-operate sectors which provides loans to their employees.
- An individual/applicant who wants to know about his capability of taking loans.

6. CONCLUSION

The system approved or rejects the loan applications. Recovery of loans is a major contributing parameter in the financial statements of a bank. It is very difficult to predict the possibility of payment of loan by the customer. Machine Learning (ML) techniques are very useful in predicting outcomes for large amount of data. In our project, three machine learning algorithms, Logistic Regression(LR), Decision Tree(DT) and Random Forest(RF) are applied to predict the loan approval of customers. The experimental results conclude that the accuracy of Random Forest machine algorithm is better than compared to Logistic Regression and decision tree machine learning approaches.

7. FUTURE SCOPE

In near future this module of prediction can be integrate with the module of automated processing system. The system is trained on old training dataset in future software can be made such that new testing date should also take part in training data after some fix time.

8.APPENDIX

A. SOURCE CODE

```
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 ensemble
from sklearn.ensemble import
GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbours 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

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

data.info()

data.isnull().sum()

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

from imblearn.combine import SMOTETomek
smote = SMOTETomek(0.90)
y = data['Loan_Status']
x = data.drop(columns=['Loan_Status'],axis=1)
x_bal,y_bal = smote.fit_resample(x,y)
print(y.value_counts())
print(y_bal.value_counts())

data.describe()

plt.figure(figsize=(12.5))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'],color='r')
plt.subplot(122)
sns.displot(data['credit_History'])
plt.show()

plt.figure(figsize=((18,24)))
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))
plt.subplot(131)
sns.countplot(data['Married'], hue=data['Gender'])
plt.subplot(132)
sns.countplot(data['self_Employed'], hue=data['Education'])
plt.figure(figsize=((18,24)))
plt.subplot(133)
sns.countplot(data['Property_Area'], hue=data['Loan_Amount_Term'])

```

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

```
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
x_bal=pd.DataFrame(x_bal,columns=names)
x_train,x_test,y_train,y_test = train_test_split(
x_bal,y_bal,test_size=0.33,random_state=42)
```

```
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))
```

```
def randomForest(x_train,x_test,y_train,y_test):
rf = RandonForestClassifier()
rf.fit(x_train,y_train)
yPred = rf.predict(x_test)
print('***RandonForestClassifier ***')
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 = KNeighboursClassfier()
knn.fit(x_train,y_train)
yPred = knn.predict(x_test)
print('***KNeighboursClassifier***')
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= GradientBoostingClassfier()
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'])

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

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

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

_classifier.save("loan.h5)
y_pred = classifier.predict(x_test)
[237] y_pred
[238] y_pred = (y_pred > 0.5)
      y_pred

[244] 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.')

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.')
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)

yPred = classifier.predict(x_test)
print(accuracy_score(y_pred,y_test)
print("ANN Model")
print('Confusion matrix')
print(confusion_matrix(y_test,yPred))
print('Classification report')
print(classification_report(y_test,yPred))

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
ypred = rf.predict(x_test)
f1_score(yPred,y_test,average='weighted')
cv = cross_val_score(rf,x,y,cv=5)
np.mean(cv)

pickle.dump(model,open('rdf.pkl','wb'))

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

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

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

@app.route('/submit',methods=["POST","GET"])
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_Term','Credit_History','Property_Area'
    ]
    data = pandas.DataFrame(input_feature,column=names)
    print(data)
    #data_scaled = scale_fit_transform(data)
    #data = pandas.DataFrame(data_scaled,column=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 will Not be Apporved")
else:
    return render_template("output.html"result ="Loan will Not be
Apporved")
#showing the prediction results in a UI

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)
```