Smart Lender-Applicant Credibility Prediction For Loan Approval UsingMachine Learning

1 Category: Machine Learning

2 Skills Required:

2.1 Python, Python for Data Analysis, Python For Data Visualization, Data PreprocessingTechniques, Machine Learning, IBM Watson

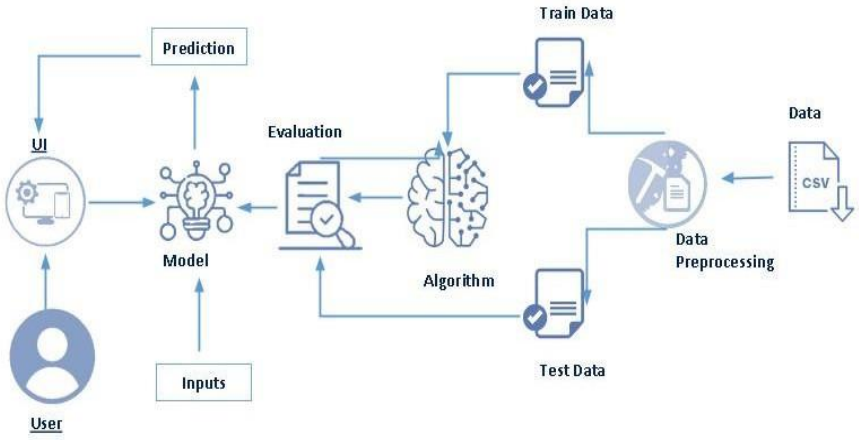
3 Project Description:

3.1 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 is very 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.

3.2 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.

3.3 We will be using classification algorithms such as Decision tree, Random forest, KNN, and 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 IBM deployment

4 Technical Architecture:



5 Pre-Requisites:

5.1 To complete this project, you must require the following software, concepts, and packages

- Anaconda

navigator:

6 Prior Knowledge

6.1 You must have prior knowledge of the following topics to complete this project.

ML Concepts

Supervised learning:

7 Project Objectives

7.1 Write what are all the technical aspects that students would get if they complete this project.

1. Knowledge of Machine Learning Algorithms.
2. Knowledge of Python Language with Machine Learning
3. You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
4. You will be able to know how to pre-process/clean the data using different data pre-processing techniques.

8 Project Flow

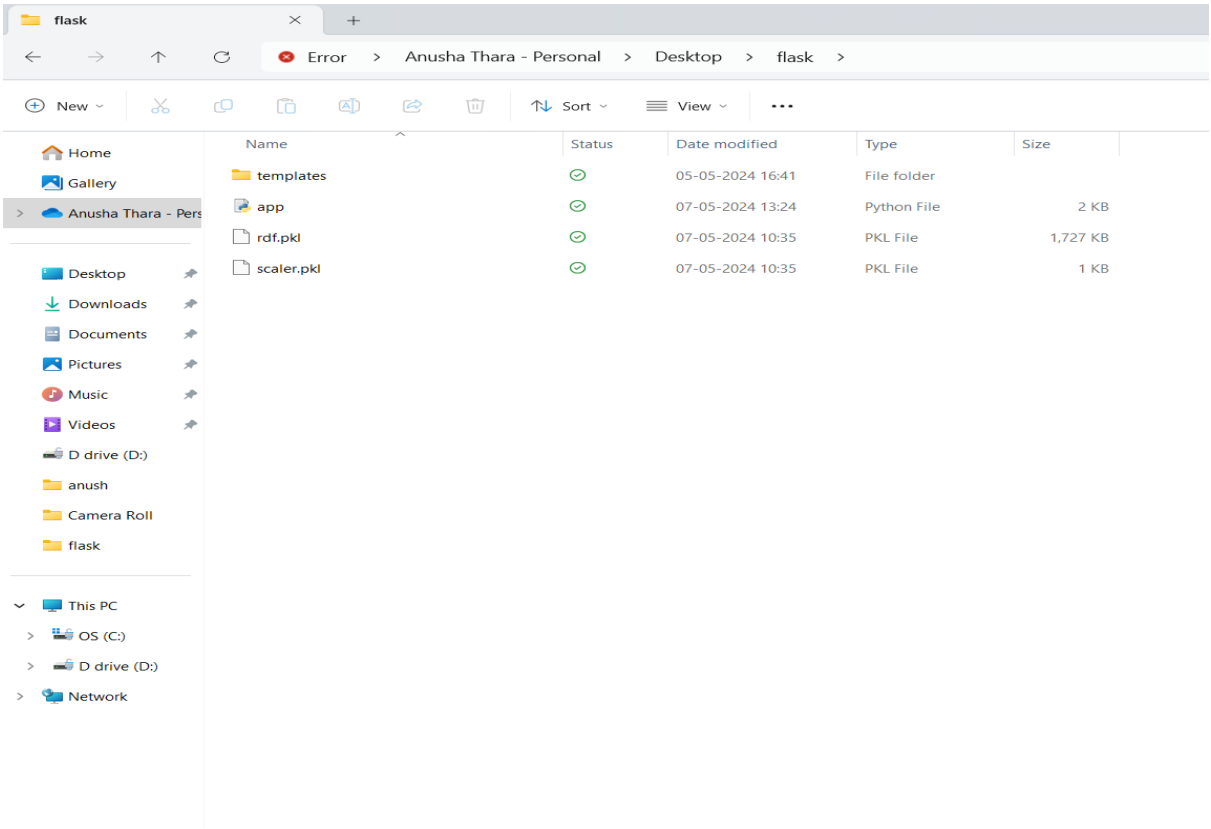
8.1 Install Required Libraries.
Data Collection.

. Collect the dataset or Create the dataset
Data Preprocessing.
.· Import the Libraries.
· Importing the dataset.
· Understanding Data Type and Summary of features.
· Take care of missing data
· Data Visualization.
· Drop the column from DataFrame & replace the missing value.
· Splitting the Dataset into Dependent and Independent variables
· Splitting Data into Train and Test.

8.2 Model Building
· Training and testing the model
· Evaluation of Model
· Saving the Model
Application Building
· Create an HTML file
· Build a Python Code
8.3 Final UI
· Dashboard Of the flask app.

9 Project Structure

9.1 Create a Project folder that contains files as shown below



10 Data Collection

10.1 ML depends heavily on data, without data, it is impossible for an "AI" to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

Download The Dataset

Duration: 0.1 Hrs

Skill Tags:

Download the dataset from the below link.

- You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc.
- Here we are using a data set which you can find in the below link and you can download it from the link: Link

11 Visualizing And Analyzing The Data

11.1 As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There is n 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

Duration: 0.1 Hrs

Skill Tags:

Import the necessary libraries as shown in the image

Import the required libraries for the model to run. The first step is usually importing the libraries that will be needed in the program

ort the required libraries for the model to run. The first step is usually importing the libraries that will be needed in the program.

```
#importing the necessary 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
import os
```

```
[ ] #loading the datset
df=pd.read_csv('/content/loan_prediction.csv')
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Statu
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	

Reading The Dataset

Duration: 0.1 Hrs

Skill Tags:

- 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 to give the directory of the CSV file.

```
#loading the datset
df=pd.read_csv('/content/loan_prediction.csv')
df
```

	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

614 rows × 13 columns

Uni-Variate Analysis

Duration: 0.5 Hrs

Skill Tags:

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

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

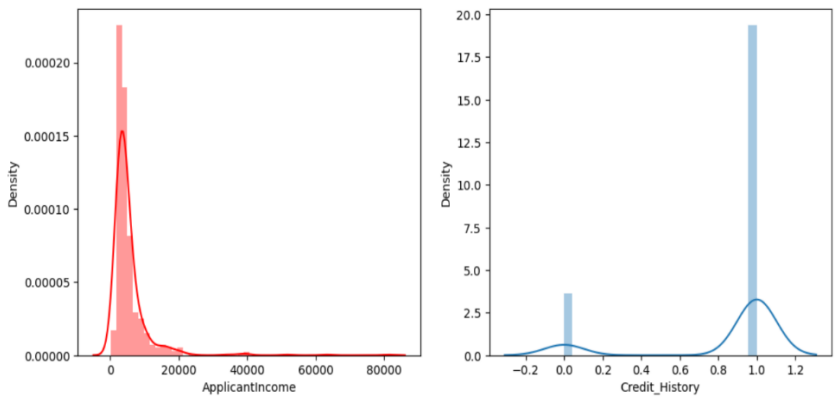
```
sns.distplot(df['ApplicantIncome'],color='r')
<ipython-input-8-5d5f89002da6>:6: UserWarning:

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

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

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

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



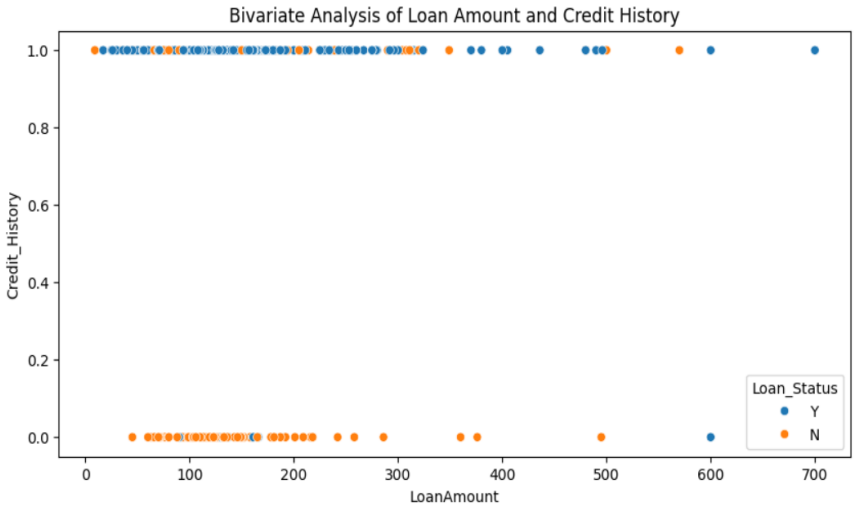
- In our dataset, we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot, we have plotted the below graph.
- From the plot we came to know, Applicants' income is skewed towards the left side, whereas credit history is categorical with 1.0 and 0.0

Bivariate Analysis

Duration: 0.5 Hrs

Skill Tags:

```
plt.figure(figsize=(10, 5))
sns.scatterplot(x='LoanAmount', y='Credit_History', data=df, hue='Loan_Status')
plt.title('Bivariate Analysis of Loan Amount and Credit History')
plt.show()
```

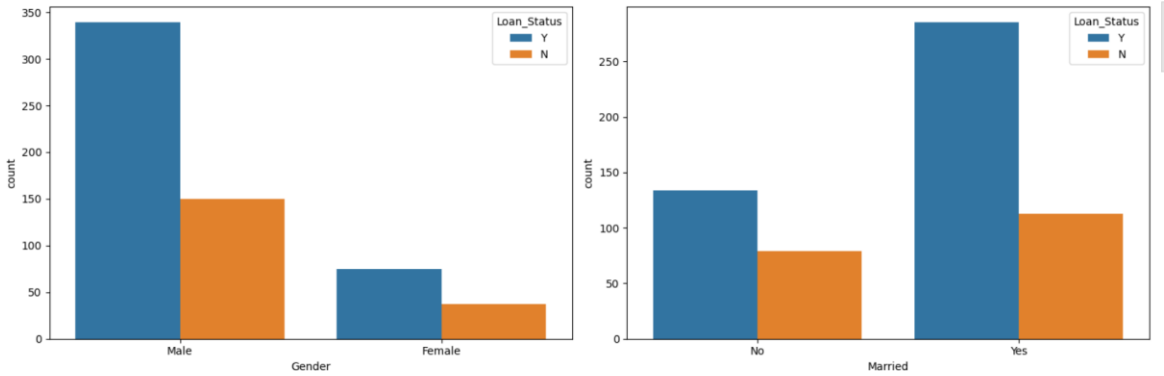


Countplot:-

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

```
#plotting the count plot
plt.subplots(figsize=(15, 5))
for i, col in enumerate(['Gender', 'Married']):
    plt.subplot(1, 2, i+1)
    sns.countplot(data=df, x=col, hue='Loan_Status')
plt.tight_layout()
plt.show()
```

```
<ipython-input-5-e6918b950db3>:4: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly
  plt.subplot(1, 2, i+1)
```



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.
- The loan amount term is based on the property area of a person holding

Multivariate Analysis

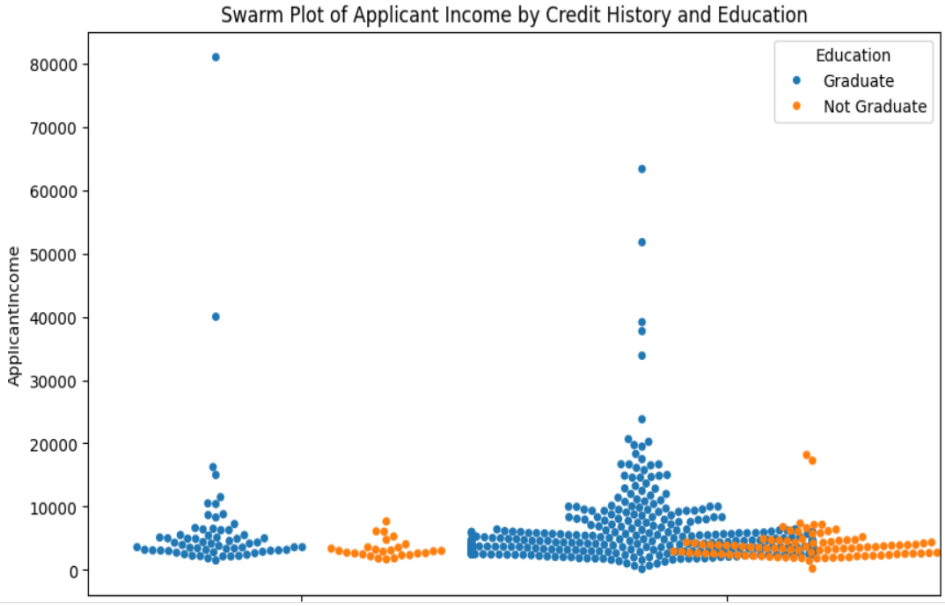
Duration: 0.5 Hrs

Skill Tags:

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

```
plt.figure(figsize=(10, 6))
sns.swarmplot(x='Credit_History', y='ApplicantIncome', hue='Education', data=df, dodge=True)
plt.title('Swarm Plot of Applicant Income by Credit History and Education')
plt.legend(title='Education')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 23.7% of the points cannot be placed; you may want to de
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 31.8% of the points cannot be placed; you may want to de
  warnings.warn(msg, UserWarning)
```



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

Descriptive Analysis

Duration: 0.5 Hrs

Skill Tags:

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have 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, maxand percentile values of continuous features.

```
[8] df.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

12 Data Pre-Processing

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 of 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 outliers
- Scaling Techniques
- Splitting dataset into training and test set

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 of 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 outliers
- Scaling Techniques
- Splitting dataset into training and test set

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 of 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 outliers
- Scaling Techniques
- Splitting dataset into training and test set

Checking For Null Values

Duration: 0.1 Hrs

Skill Tags:

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

```
#checking for null values
df.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 to it. From the below image we found that there are no null values present in our dataset. So we can skip the handling of the missing values step.

```
df.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
```

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

```
[ ] df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
    df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
    df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mean())
```

```
#fill the missing values for categorical terms - mode
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.isnull().sum()
```

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

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

Handling Categorical Values

Duration: 0.5 Hrs

Skill Tags:

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.

```
[ ] df['Gender']=df['Gender'].map({'Female':1,'Male':0})
    df['Property_Area']=df['Property_Area'].map({'Urban':2,'Semiurban':1,'Rural':0})
    df['Married']=df['Married'].map({'Yes':1,'No':0})
    df['Self_Employed']=df['Self_Employed'].map({'Yes':1,'No':0})
    df['Education']=df['Education'].map({'Graduate':1,'Not Graduate':0})
    df['Loan_Status']=df['Loan_Status'].map({'Y':1,'N':0})
```

converting string datatype variables into integer data type

```
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')
```

Balancing The Dataset

Duration: 0.5 Hrs

Skill Tags:

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 biased results, which means our model is able to predict only one class element

For Balancing the data we are using 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.

```
[ ] from imblearn.combine import SMOTETomek
```

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

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

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

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

```
Loan_Status
1    422
0    192
Name: count, dtype: int64
Loan_Status
1    347
0    347
Name: count, dtype: int64
```

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

Duration: 0.5 Hrs

Skill Tags:

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 biased results, which means our model is able to predict only one class element

For Balancing the data we are using 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.

```
[ ] from imblearn.combine import SMOTETomek

[ ] smote=SMOTETomek()

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

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

[ ] print(y.value_counts())
    print(y_bal.value_counts())

    Loan_Status
    1    422
    0    192
    Name: count, dtype: int64
    Loan_Status
    1    347
    0    347
    Name: count, dtype: int64
```

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

Scaling The Data

Duration: 0.1 Hrs

Skill Tags:

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.

```
[ ]  from sklearn.preprocessing import StandardScaler
     sc=StandardScaler()
     x_bal=pd.DataFrame(sc.fit_transform(x_bal),columns=x_bal.columns)
```

We will perform scaling only on the input values

Once the dataset is scaled, it will be converted into array and we need to convert it back to dataframe.

Splitting Data Into Train And Test

Duration: 0.1 Hrs

Skill Tags:

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 the x variable, df is passed by 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, and random_state.

```
x_train,x_test,y_train,y_test=train_test_split(x_bal,y_bal,test_size=0.2,random_state=42)
```

13 Model Building

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 saved based on its performance.

Decision Tree Model

Duration: 0.1 Hrs

Skill Tags:

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

```
[ ] def DecisionTree(x_train,x_test,y_train,y_test):
        dt_model= DecisionTreeClassifier()
        dt_model.fit(x_train, y_train)
        dt_pred=dt_model.predict(x_test)
        print("***Decision Tree Classifier")
        print('confusion atrix')
        print(confusion_matrix(y_test,dt_pred))
        print('classification report')
        print(classification_report(y_test,dt_pred))
```

Random Forest Model

Duration: 0.1 Hrs

Skill Tags:

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialized 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 are done.

```
def RandomForest(x_train,x_test,y_train,y_test):
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
    rf_model.fit(x_train, y_train)
    rf_pred=rf_model.predict(x_test)
    print("***RandomForestClassifier***")
    print('confusion matrix')
    print(confusion_matrix(y_test,rf_pred))
    print('classification report')
    print(classification_report(y_test,rf_pred))
```

KNN Model

Duration: 0.1 Hrs

Skill Tags:

A function named KNN is created and train and test data are passed as the parameters. Inside the function, the KNeighborsClassifier algorithm is initialized 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 KNN(x_train,x_test,y_train,y_test):
        knn_model = KNeighborsClassifier(n_neighbors=5)
        knn_model.fit(x_train, y_train)
        knn_pred=knn_model.predict(x_test)
        print("***KNeighborsClassifier***")
        print('confusion matrix')
        print(confusion_matrix(y_test,knn_pred))
        print('classification report')
        print(classification_report(y_test,knn_pred))
```

Compare The Model

Duration: 0.5 Hrs

Skill Tags:

For comparing the above four models compareModel function is defined.

```
XGBoost Accuracy: 0.84
```

```
[ ] dt_model = DecisionTreeClassifier()
    dt_model.fit(x_train, y_train)
    dt_pred=dt_model.predict(x_test)
```

```
    print(f"Decision Tree Accuracy: {accuracy_score(y_test, dt_pred):.2f}")
```

```
Decision Tree Accuracy: 0.77
```

```
[ ] from sklearn.ensemble import RandomForestClassifier

    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
    rf_model.fit(x_train, y_train)
    rf_pred=rf_model.predict(x_test)
```

```
[ ] print(f"Random Forest Accuracy: {accuracy_score(y_test, rf_pred):.2f}")
```

```
Random Forest Accuracy: 0.86
```

```
[ ] from sklearn.neighbors import KNeighborsClassifier

    knn_model = KNeighborsClassifier(n_neighbors=5)
    knn_model.fit(x_train, y_train)
    knn_pred=knn_model.predict(x_test)
```

```
[ ] print(f"KNN Accuracy: {accuracy_score(y_test, knn_pred):.2f}")
```

```
KNN Accuracy: 0.81
```

Evaluating Performance Of The Model And Saving The Model

Duration: 0.1 Hrs

Skill Tags:

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 themodel by pickle.dump().

Note: To understand cross-validation, refer this link

```
[ ]  from sklearn.model_selection import cross_val_score
     rf_model=RandomForestClassifier()
     rf_model.fit(x_train,y_train)
     rf_pred=xgb.predict(x_test)
```

```
[ ]  f1_score(rf_pred,y_test,average="weighted")

     0.841677445119298
```

```
[ ]  pickle.dump(rf_model,open('rdf.pkl','wb'))
```

```
[ ]  model_loaded=pickle.load(open("/content/rdf.pkl",'rb'))
```

14 Application Building

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 serverside script

Building Html Pages

Duration: 0.5 Hrs

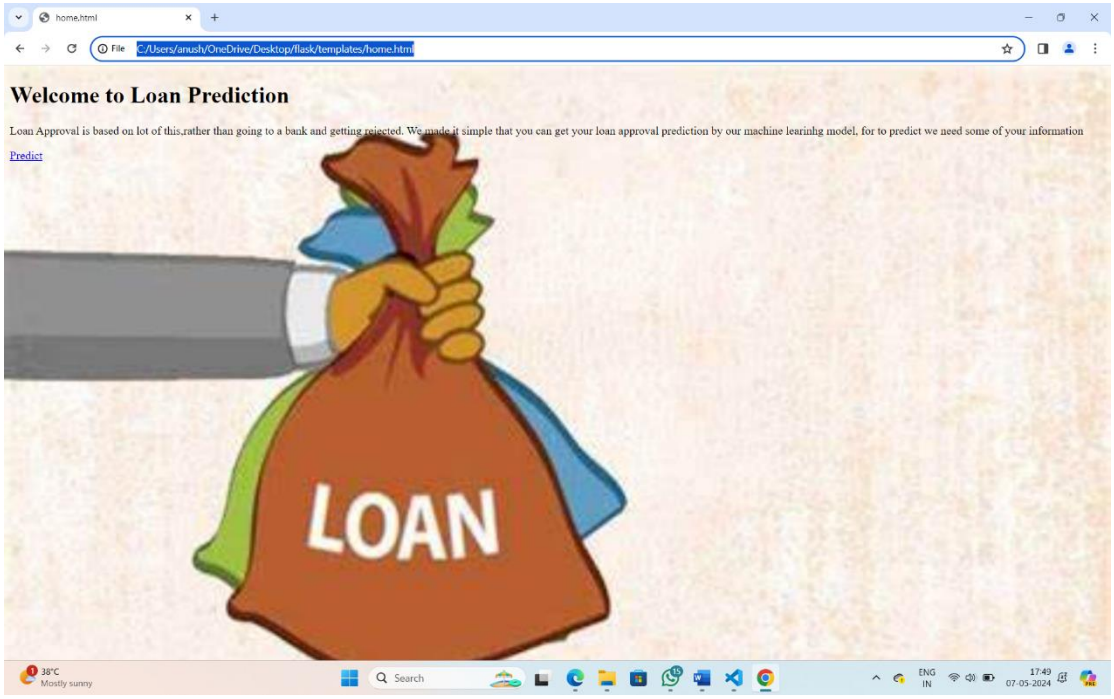
Skill Tags:

For this project create three HTML files namely

- home.html
- predict.html
- submit.html

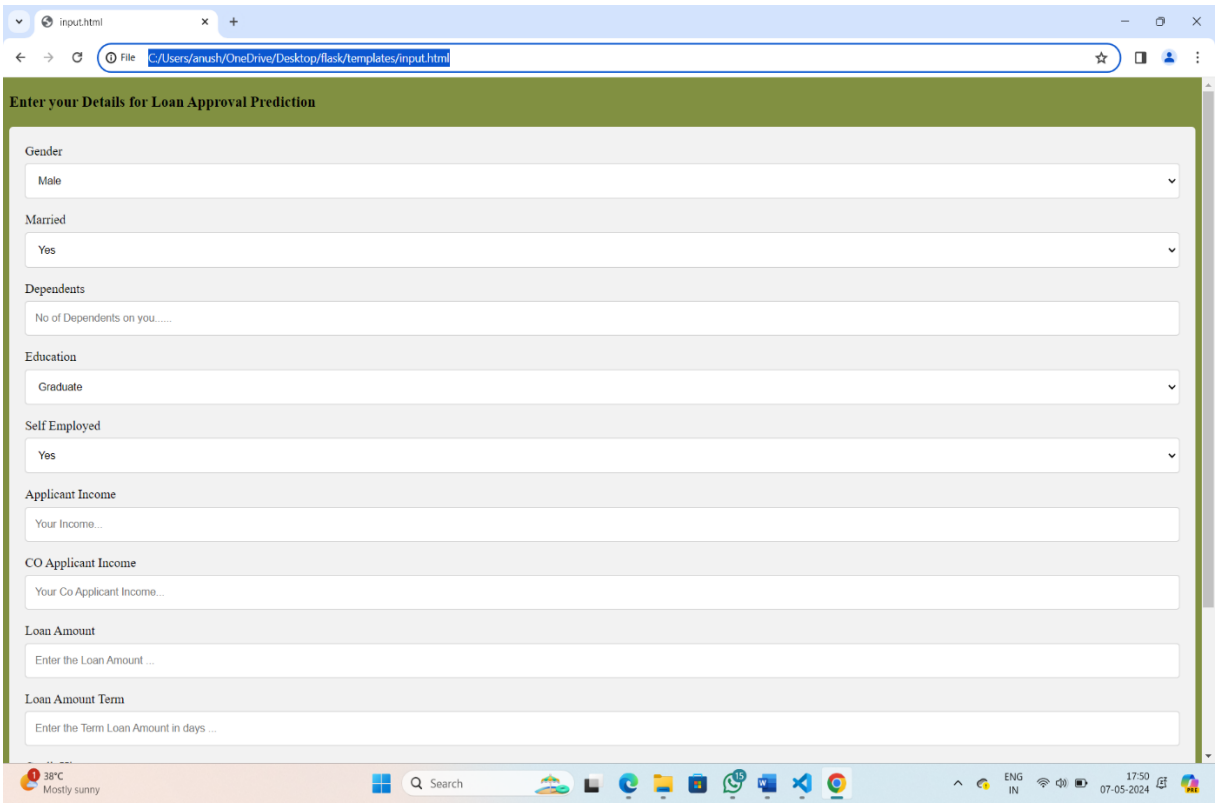
and save them in the templates folder.

Let’s see how our home.html page looks like:



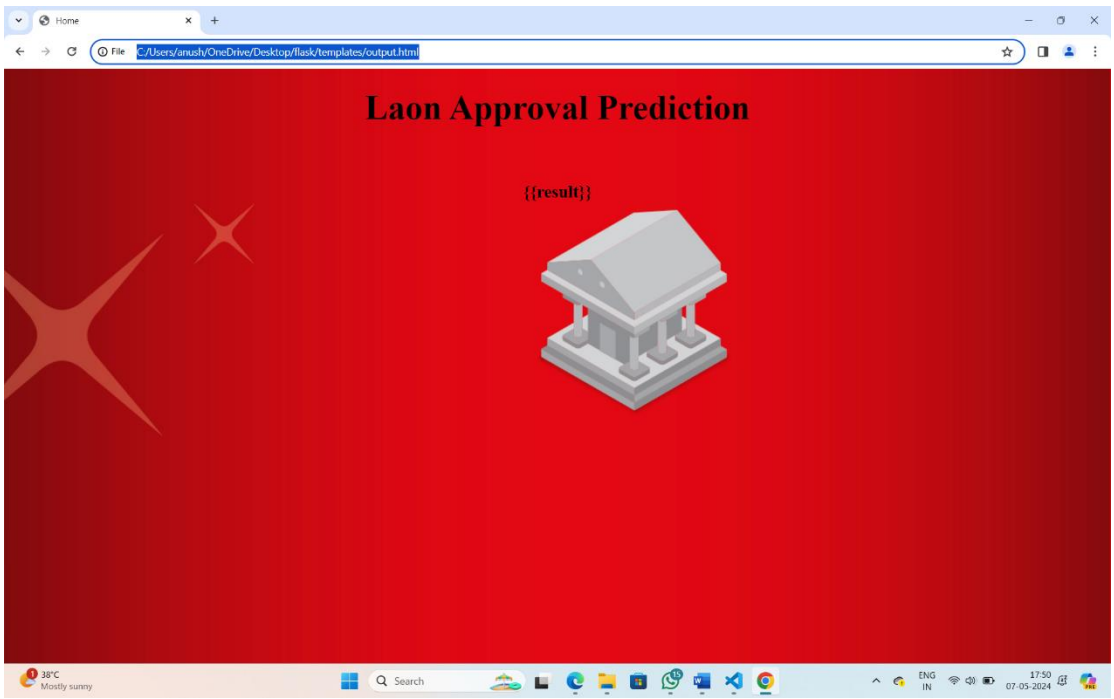
Now when you click on predict button from top right corner you will get redirected to predict.html

Lets look how our predict.html file looks like:



Now when you click on submit button from left bottom corner you will get redirected to submit.html

Lets look how our submit.html file looks like:



Build Python Code

Duration: 0.5 Hrs

Skill Tags:

Import the libraries

```
from flask import Flask,render_template,request
import numpy as np
import pandas as pd
import os
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__) asan argument.

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

Render HTML page:

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

Here we will be using declared constructor to route to the HTML page which we have created earlier.

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

from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```python
@app.route('/')
def home():
    return render_template('home.html')
Codeium: Refactor | Explain | Generate Docstring | X
@app.route('/predict',methods=["POST","GET"])
def predict() :
    return render_template("input.html")
Codeium: Refactor | Explain | Generate Docstring | X
@app.route('/submit',methods=["POST","GET"])
def submit():
    input_feature=[int(x) for x in request.form.values()]
    input_feature=[np.array(input_feature)]
    print(input_feature)
    names=['Gender','Married','Dependents','Education','Self_Employed','ApplicantIncome','CoapplicantIncome','Loan
    data=pd.DataFrame(input_feature,columns=names)
    print(data)
    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 approved")
    else:
        return render_template("output.html",result="loan will be approved")

if __name__=="__main__":
        app.run(debug=False)
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using a 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:

```python
if __name__=="__main__":
        app.run(debug=False)
```

Run The Application

Duration: 0.1 Hrs

Skill Tags: