PROJECT REPORT

CONTENT

1. INTRODUCTION

1.1 OVERVIEW

The project of predicting personal loan approval using machine learning aims to develop a model that can accurately predict the probability of loan approval based on a set of parameters. The project will use a dataset of past loan applicants to train and test the model. The dataset will be preprocessed and cleaned to remove any inconsistencies or irrelevant information. Then, various machine learning algorithms such as logistic regression, decision trees, and random forest will be applied to determine the best-performing model. The chosen model will be evaluated based on its accuracy, precision, recall, and F1 score. Once the optimal model has been identified, it will be deployed to a web application to provide users with a user-friendly interface to check their eligibility for a personal loan. The application will take user inputs such as age, income, credit score, and employment status, and return the probability of loan approval. The final aim of this project is to reduce the risk of default for lenders and streamline the loan approval process by providing accurate and reliable applicant evaluations.

1.2 PURPOSE

The purpose of predicting personal loan approval is to develop a model that can accurately predict the probability of loan approval based on various applicant parameters. The project aims to address the following objectives:

1. Improve the accuracy of loan approval decisions: By using machine learning algorithms, the project seeks to reduce the risk of default for lenders by improving the accuracy of loan approval decisions.
2. Streamline the loan approval process: The project aims to make the loan approval process faster and more efficient by providing a model that can quickly assess the probability of loan approval for an applicant.
3. Provide a user-friendly interface for loan eligibility: The web application developed as part of the project will provide users with a user-friendly interface to check their eligibility for a personal loan.

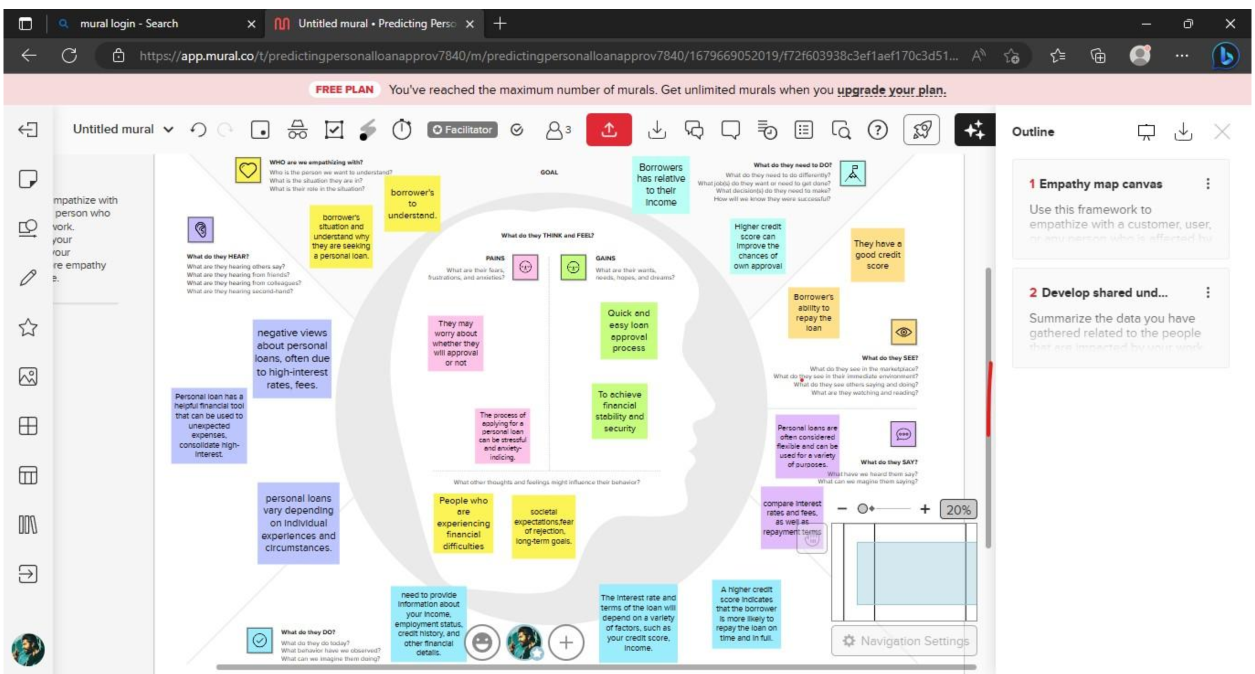
2. PROBLEM DEFINITION & DESIGN THINKING

<u>PROBLEM DEFINITION:</u>

The problem definition of predicting personal loan approval using machine learning is to develop a model that can accurately predict the probability of loan approval based on various applicant parameters. The problem arises because the loan approval process can be time-consuming and may involve a lot of paperwork. Also, lenders may face difficulties in assessing the creditworthiness of an applicant due to insufficient data or inaccurate credit history. These factors can result in delayed or inaccurate loan approval decisions, which can cause inconvenience to borrowers and increase the risk of default for lenders.

<u>DESIGN THINKING:</u>

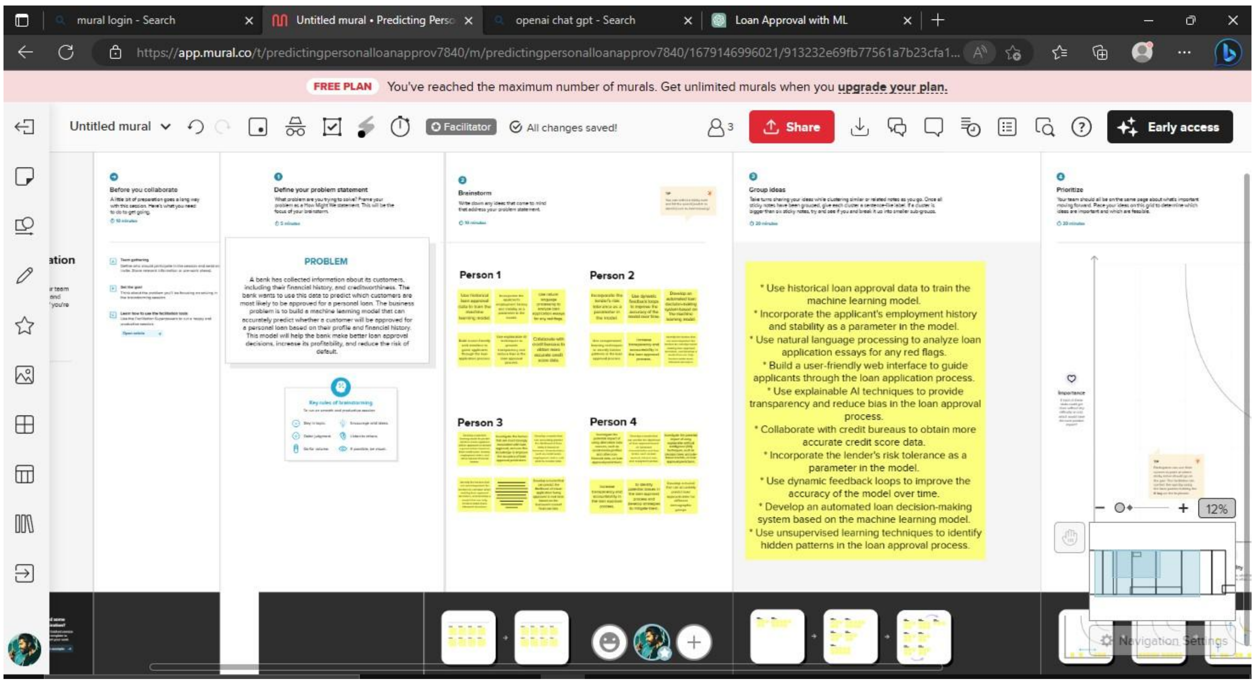
2.1 EMPATHY MAP



Empathy mapping is a technique that helps understand users' perspectives and experiences in a particular context. In the context of predicting personal loan approval using machine learning, the empathy map can help understand the users' emotions, attitudes, behaviors, and pain points related to the loan approval process.

Problem Statement: Develop a machine learning model that can accurately predict personal loan approval based on various applicant parameters.

2.2 IDEATION & BRAINSTOMING MAP



Ideation and brainstorming map is a technique to generate and organize ideas related to a particular topic. In the context of predicting personal loan approval using machine learning, ideation and brainstorming can help generate ideas for improving the loan approval process and developing a more accurate machine learning model.

3 RESULT

Result :

```
data=pd.read_csv("/content/drive/MyDrive/Dataset/train_u6lujuX_CVtuZ9i.csv")
data.head()
```

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

DATA PREPROCESSING

handling Categorical values

```
data.head()
```

Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0.0	0.0	0	1	0.0	5849	0.0	NaN	360.0	1.0	2	1
0.0	1.0	1	1	0.0	4583	1508.0	128.0	360.0	1.0	0	0
0.0	1.0	0	1	1.0	3000	0.0	66.0	360.0	1.0	2	1
0.0	1.0	0	0	0.0	2583	2358.0	120.0	360.0	1.0	2	1
0.0	0.0	0	1	0.0	6000	0.0	141.0	360.0	1.0	2	1

Handling Missing values

```
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          0
dtype: int64
```

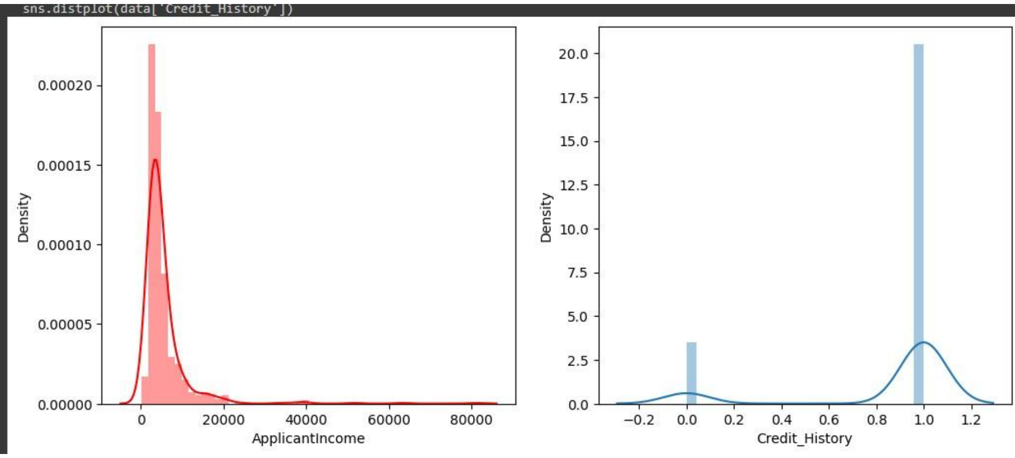
Handling Categorical values

```
data.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    int64
 1   Married            614 non-null    int64
 2   Dependents         614 non-null    int64
 3   Education          614 non-null    int64
 4   Self_Employed      614 non-null    int64
 5   ApplicantIncome    614 non-null    int64
 6   CoapplicantIncome  614 non-null    int64
 7   LoanAmount         614 non-null    int64
 8   Loan_Amount_Term   614 non-null    int64
 9   Credit_History     614 non-null    int64
 10  Property_Area      614 non-null    int64
 11  Loan_Status        614 non-null    int64
dtypes: int64(12)
memory usage: 57.7 KB
```

Data Visualization

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



Balancing the dataset

```
x_bal.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	0	0	0	1	0	5849	0	120	360	1	2
1	0	1	1	1	0	4583	1508	128	360	1	0
2	0	1	0	1	1	3000	0	66	360	1	2
3	0	1	0	0	0	2583	2358	120	360	1	2
4	0	0	0	1	0	6000	0	141	360	1	2

Scalling the dataset

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

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	0.0	-1.159502	-0.692472	0.599248	-0.328042	0.088761	-0.505149	-0.344427	0.275037	0.626689	1.362793
1	0.0	0.862439	0.362485	0.599248	-0.328042	-0.137238	-0.054573	-0.246405	0.275037	0.626689	-1.245364
2	0.0	0.862439	-0.692472	0.599248	3.048390	-0.419827	-0.505149	-1.006075	0.275037	0.626689	1.362793
3	0.0	0.862439	-0.692472	-1.668758	-0.328042	-0.494267	0.199399	-0.344427	0.275037	0.626689	1.362793
4	0.0	-1.159502	-0.692472	0.599248	-0.328042	0.115717	-0.505149	-0.087119	0.275037	0.626689	1.362793

```
#splitting the dataset in train and test on balanced dataset
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
```

```
(565, 11)
```

```
[ ] X_test.shape
```

```
(279, 11)
```

```
[ ] y_train.shape, y_test.shape
```

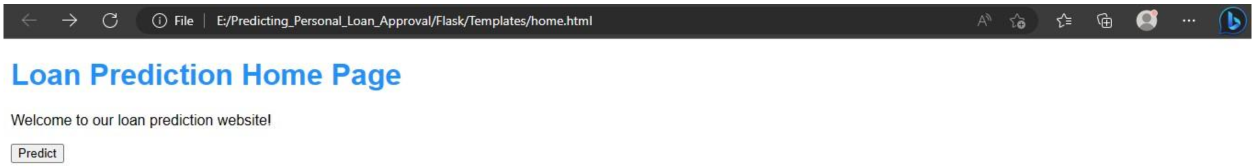
```
((565,), (279,))
```

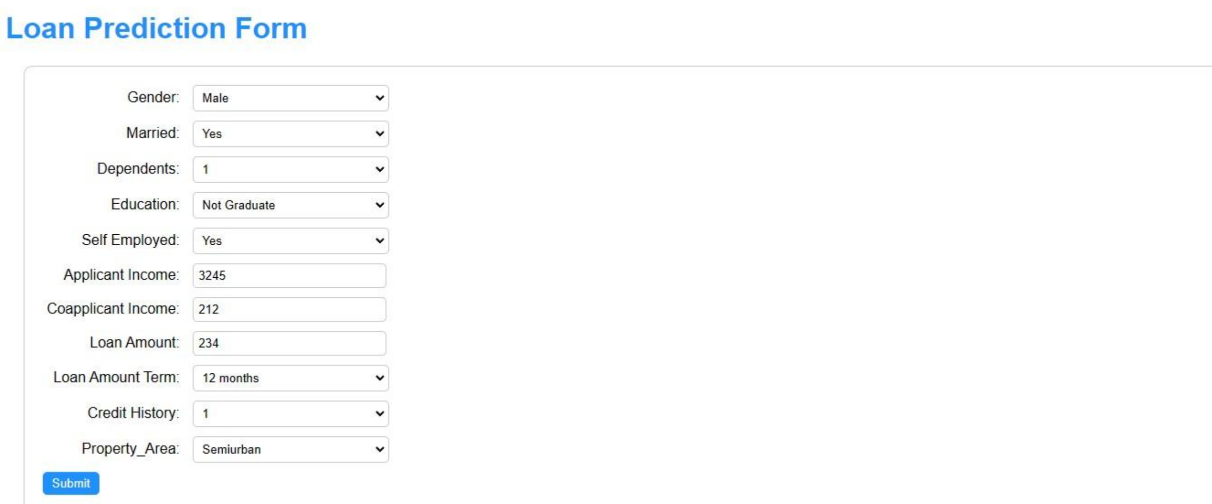
Model Building

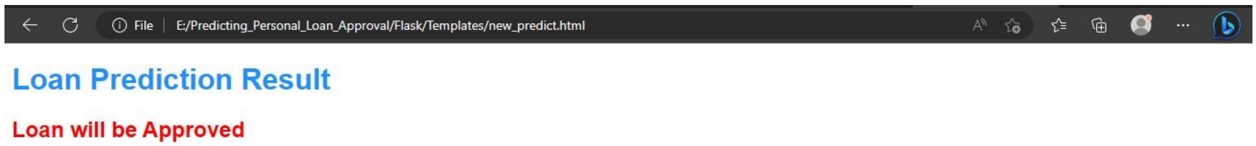
```
#printing the train accuracy and test accuracy respectively
RandomForest(X_train,X_test,y_train,y_test)
```

```
1.0
0.7992831541218638
```

Web Framework







4. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Improved accuracy: Machine learning models can analyze large amounts of data and identify patterns that humans may not be able to detect. This can lead to more accurate loan approval predictions.
- Faster decision-making: Machine learning algorithms can process data quickly and make loan approval decisions in real time, which can improve the speed and efficiency of the loan approval process.
- Reduced bias: Machine learning models can be trained to eliminate bias in loan approval decisions, which can help ensure that loan approval decisions are fair and equitable.
- Increased transparency: Machine learning models can be designed to provide explanations for their loan approval decisions, which can increase transparency and help build trust in the loan approval process.
- Cost savings: By automating the loan approval process using machine learning, lenders can reduce costs associated with manual loan processing and improve operational efficiency.

DISADVANTAGES:

- Lack of interpretability: Some machine learning models can be difficult to interpret, which can make it difficult to understand why loan approval decisions are being made.

- Overfitting: Machine learning models can be prone to overfitting, which occurs when a model is too complex and performs well on the training data but poorly on new data.
- Lack of data quality: Machine learning models rely on high-quality data to make accurate predictions, so if the data used to train the model is incomplete or inaccurate, the predictions may be unreliable.
- Ethical concerns: Machine learning models can be trained on biased data or perpetuate biases, which can lead to discriminatory loan approval decisions.
- Security risks: Machine learning models may be vulnerable to attacks from malicious actors, which can compromise the security of borrower data and put lenders at risk.

5. APPLICATIONS

Predicting personal loan approval using machine learning has a wide range of applications across the financial industry, including:

1. Banks and financial institutions: Banks and other financial institutions can use machine learning to automate the loan approval process and make more accurate loan approval decisions. This can lead to faster loan processing times, reduced costs, and improved customer satisfaction.
2. Peer-to-peer lending platforms: Peer-to-peer lending platforms can use machine learning to evaluate borrower creditworthiness and make loan approval decisions. This can help ensure that loans are being made to creditworthy borrowers and reduce the risk of default.
3. Credit scoring companies: Credit scoring companies can use machine learning to develop more accurate credit scoring models, which can be used by lenders to make loan approval decisions. This can help improve access to credit for underserved populations and reduce the risk of default.
4. Insurance companies: Insurance companies can use machine learning to assess the risk of lending to borrowers and make more accurate loan approval decisions. This can help reduce the risk of default and improve the profitability of insurance products.
5. Fintech startups: Fintech startups can use machine learning to develop innovative loan approval products and services, such as microloans and instant loan approvals. This can help improve access to credit for underserved populations and reduce the risk of default.

6. Government agencies: Government agencies can use machine learning to develop more effective loan programs and improve the efficiency of the loan approval process. This can help improve access to credit for individuals and small businesses and promote economic growth.

Overall, predicting personal loan approval using machine learning has the potential to transform the way that loans are approved and processed, leading to faster decision-making, reduced costs, and improved access to credit for individuals and small businesses.

6. CONCLUSION

In conclusion, predicting personal loan approval using machine learning has the potential to significantly improve the loan approval process, leading to faster decision-making, reduced costs, and improved access to credit for individuals and small businesses. Machine learning models can analyze large amounts of data, identify patterns, and make loan approval decisions in real time, which can improve the speed and efficiency of the loan approval process. By automating the loan approval process using machine learning, lenders can reduce costs associated with manual loan processing and improve operational efficiency. However, there are also potential disadvantages, such as lack of interpretability and data quality issues, that need to be addressed to ensure that loan approval decisions are fair, transparent, and unbiased. Overall, predicting personal loan approval using machine learning has a wide range of applications across the financial industry, and has the potential to transform the way that loans are approved and processed.

7. FUTURE SCOP

The future scope of predicting personal loan approval using machine learning is very promising. As the financial industry continues to become more data-driven, machine learning models are expected to play an increasingly important role in the loan approval process. Here are some potential future developments:

1. Use of more advanced machine learning techniques: As machine learning techniques continue to evolve, more advanced algorithms and models may be developed that can improve loan approval predictions even further. For example, deep learning techniques may be used to analyze unstructured data, such as borrower social media activity, to improve loan approval predictions.
2. Integration with blockchain technology: Blockchain technology has the potential to improve the security and transparency of the loan approval process, and may be integrated with machine learning models to further improve loan approval predictions.
3. Collaboration between lenders: Lenders may collaborate to share data and develop more accurate loan approval models. This could lead to more consistent loan approval decisions across different lenders, and could help reduce the risk of default.
4. Increased use of alternative data sources: Machine learning models may be trained on alternative data sources, such as mobile phone usage data or utility bill payment history, to improve loan approval predictions. This could help improve access to credit for underserved populations who may not have traditional credit histories.
5. Expansion to other types of loans: The use of machine learning to predict loan approvals may expand to other types of loans, such as business loans, mortgage loans, and car loans.

Overall, the future scope of predicting personal loan approval using machine learning is very promising, and is likely to lead to continued improvements in the speed, efficiency, and accuracy of the loan approval process.

8. APPENDIX

8.1 SOURCE CODE

Importing 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, Ra
ndomForestClassifier
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
```

Load the dataset

```
data=pd.read_csv("/content/drive/MyDrive/Dataset/train_u6lu
juX_CVtuZ9i.csv")
data.head()
```

```
#dropping the Loan id columns because there is no use it fo
r the model building
data.drop(['Loan_ID'],axis=1,inplace=True)
```

Data Preprocessing

```
#hadling categorical features

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

Handling Missing values

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

```
Gender               125
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
```

```
data['Gender'] = data['Gender'].fillna(data['Gender'].mode(
)[0])
data['Married'] = data['Married'].fillna(data['Married'].mo
de()[0])
#replacing + with space for filling the nan values
data['Dependents']=data['Dependents'].str.replace('+','')
#1 1 2 3+ ---3
```

```
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])
```

Handling Categorical values

```
#getting the total info of the data after performing categorical to numerical and replacing missing values
data.info()


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

Data Visualization

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


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

#visualizing two collumns againist each other
plt.figure(figsize=(20, 5))
plt.subplot(131)
sns.countplot(data['Gender'], hue=data['Married'])
plt.subplot(132)
sns.countplot(data['Self_Employed'], hue=data['Education'])
plt.subplot(133)
sns.countplot(data['Property_Area'], hue=data['Loan_Amount_
Term'])

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

Balancing the Dataset

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

# Separate the features and target variable
X = data.drop(columns= ['Loan_Status'], axis=1)
y = data['Loan_Status']

# Create an instance of the SMOTE algorithm
smote = SMOTE()

# Fit and transform the dataset using SMOTE
x_bal, y_bal = smote.fit_resample(X, y)


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


names = x_bal.columns
x_bal.head()
```

Scalling the dataset

```
from sklearn.preprocessing import StandardScaler
#performing feature scaling operation using standard scalle
r on x part of the dataset bacause
#there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)


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


#splitting the dataset in train and test on balanced datase
t
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
X_test.shape
y_train.shape, y_test.shape
```

Model Building

```
#importing and building the random forest 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))

#printing the train accuracy and test accuracy respectively
RandomForest(X_train,X_test,y_train,y_test)
```

Hyper Parameter tuning

```
rf = RandomForestClassifier()

#giving some parameters that can be used in randized search
 cv
parameters = {
                'n_estimators': [1,20,30,55,68, 74,90,120,1
15],
                'criterion': ['gin', 'entropy'],
                'max_features': ["auto", "sqrt", "log2"],
          'max_depth': [2,5,8,10], 'verbose':[1,2,3,4,6,8,9
,10]
}

#performing the randomized cv
RCV = RandomizedSearchCV(estimator=rf, param_distributions=
parameters, cv=10, n_iter=4)

RCV.fit(X_train,y_train)


#getting the best parameters from the giving list and best
score from them
bt_params = RCV.best_params_
bt_score = RCV.best_score_

bt_params
bt_score


#training and test xg boost model on the best parameters go
r from the randomized cv
def RandomForest(X_train,X_test,y_train,y_test):
  model = RandomForestClassifier(verbose= 9, n_estimators=
55, max_features= 'auto', max_depth=2, criterion='entropy')
```

```
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print("Training Accuracy")
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print("Testing Accuracy")
  print(accuracy_score(yPred, y_test))

#printing the train and test accuracy after hyper parameter
 tuning
RandomForest(X_train,X_test,y_train,y_test)


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

web Framework

app.py

```
import pandas as pd

import os

import numpy as np

import pickle

from flask import Flask, render_template, request


app=Flask(__name__)

model = pickle.load(open(r'rdf.pkl', 'rb'))

scale = pickle.load(open(r'scale.pkl','rb'))
```

```
@app.route('/') #rendering the html template

def home():

    return render_template('predict.html')


@app.route('/predict',methods=["POST", "GET"]) #rendering the
html

def predict():

    return render_template('predict.html')


@app.route('/submit', methods=["POST", "GET"])  #rout to show
the predictions in a web UI

def submit():

    #redaing the inputs given by the user

    input_feature=[int(x) for x in request.form.values()]

    #input_feature = np.transpose (input_feature)

    input_feature =[np.array(input_feature)]

    print(input_feature)

    names = ['Gender', 'Married', 'Dependents', 'Education',
'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome',
'LoanAmount', 'Loan_Amount_Term', 'Creadit_History',
'Property_Area']

    data = pandas.DataFrame(input_feature, colums=names)

    print(data)


    data_scaled = scale.fit_transform(data)

    data = pandas.DataFrame(data,columns=names)


    #predictions using the loaded model file

    prediction = model.predict(data)

    print(prediction)
```

```
    prediction = int(prediction)

    print(type(prediction))


    if (prediction == 0):

        return render_template('predict.html', result="Loan will
Not be Approved")

    else:

        return render_template('predict.html', result = "Loan
will be Approved")


if __name__ =="__main__":

    app.run(debug=True) #running the app
```