**Project Title**

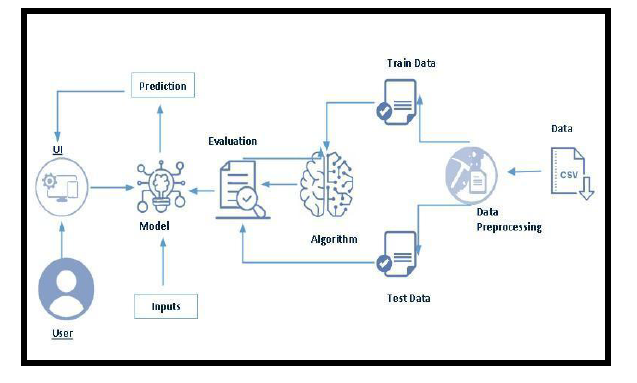
**Predicting Personal Loan Approval Using Machine Learning**

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**



**Project Flow**

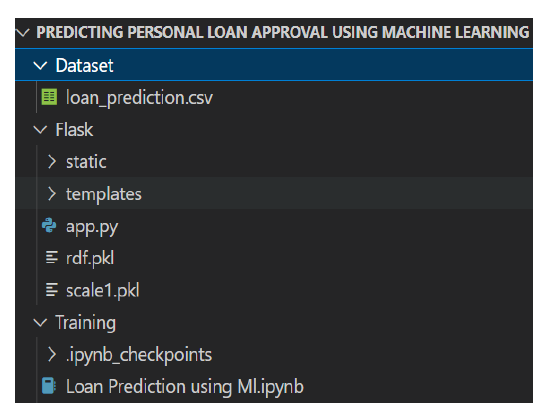
* User interacts with the UI to enter the input.
* Entered input is analysed by the model which is integrated.
* Once model analyses the input the prediction is showcased on the UI

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

1. Define Problem / Problem Understanding
   1. Specify the business problem
   2. Business requirements
   3. Literature Survey
   4. Social or Business Impact
2. Data Collection & Preparation
   1. Collect the dataset
   2. Data Preparation
3. Exploratory Data Analysis
   1. Descriptive statistical
   2. Visual Analysis
4. Model Building
   1. Training the model in multiple algorithms
   2. Testing the model
5. Performance Testing & Hyperparameter Tuning
   1. Testing model with multiple evaluation metrics
   2. Comparing model accuracy before & after applying hyperparameter tuning
6. Model Deployment
   1. Save the best model
   2. Integrate with Web Framework
7. Project Demonstration & Documentation
   1. Record explanation Video for project end to end solution
   2. Project Documentation-Step by step project development procedure

**Project Structure**

Create the Project folder which contains files as shown below



* We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
* rdf.pkl is our saved model. Further we will use this model for flask integration.
* Training folder contains a model training file.

**Milestone 1: Define Problem / Problem Understanding**

**Activity 1: Specify the business problem**

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.

**Activity 2: Business requirements**

The business requirements for a machine learning model to predict personal loan approval include the ability to accurately predict loan approval based on applicant information, Minimise the number of false positives (approved loans that default) and false negatives (rejected loans that would have been successful).Provide an explanation for the model's decision, to comply with regulations and improve transparency.

**Activity 3: Literature Survey**

As the data is increasing daily due to digitization in the banking sector, people want to apply for loans through the internet. Machine Learning (ML), as a typical method for information investigation, has gotten more consideration increasingly. Individuals of various businesses are utilising ML calculations to take care of the issues dependent on their industry information. Banks are facing a significant problem in the approval of the loan. Daily there are so many applications that are challenging to manage by the bank employees, and also the chances of some mistakes are high.Most banks earn profit from the loan, but it is risky to choose deserving customers from the number of applications.There are various algorithms that have been used with varying levels of success. Logistic regression, decision tree, random forest, and neural networks have all been used and have been able to accurately predict loan defaults. Commonly used features in these studies include credit score, income, and employment history, sometimes also other features like age, occupation, and education level.

**Activity 4: Social or Business Impact.**

**Social Impact:-** Personal loans can stimulate economic growth by providing individuals with the funds they need to make major purchases, start businesses, or invest in their education.

**Business Model/Impact:-** Personal loan providers may charge fees for services such as loan origination, processing, and late payments.Advertising the brand awareness and marketing to reach out to potential borrowers to generate revenue.

**Milestone 2: Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

**Activity 1: Collect the dataset**

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: <https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset>

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

visualisation techniques and some analysing techniques.

**Note:** There are a number of techniques for understanding the data. But here we have used some

of it. In an additional way, you can use multiple techniques.

**Activity 1.1: Importing the libraries**

Import the necessary libraries as shown in the image. Here we have used visualisation

style as fivethirtyeight.

|  |
| --- |
| #Importing the Libraries  import numpy as np  import pandas as pd  import pickle  import matplotlib.pyplot as plt  %matplotlib.pylot as plot  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 |

**Activity 1.2: Read the Dataset**

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.

#Data Collection and Preparation

#Read The Data Set

df=pd.read\_csv("E:\\NMDS\pl\_train.csv")

df.head()

#Data Collection and Preparation

#Read The Data Set

df1=pd.read\_csv("E:\\NMDS\pl\_test.csv")

df1.head()

**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: 43.2+ KB

**df1.info()**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 367 entries, 0 to 366

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Loan\_ID 367 non-null object

1 Gender 356 non-null object

2 Married 367 non-null object

3 Dependents 357 non-null object

4 Education 367 non-null object

5 Self\_Employed 344 non-null object

6 ApplicantIncome 367 non-null int64

7 CoapplicantIncome 367 non-null int64

8 LoanAmount 362 non-null float64

9 Loan\_Amount\_Term 361 non-null float64

10 Credit\_History 338 non-null float64

11 Property\_Area 367 non-null object

dtypes: float64(3), int64(2), object(7)

memory usage: 24.4+ KB

**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

**df1.isnull().sum()**

Loan\_ID 0

Gender 11

Married 0

Dependents 10

Education 0

Self\_Employed 23

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 5

Loan\_Amount\_Term 6

Credit\_History 29

Property\_Area 0

dtype: int64

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

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

df1['Gender']=df1['Gender'].fillna(df1['Gender'].mode()[0])

df1['Married']=df1['Married'].fillna(df1['Married'].mode()[0])

#replacing + with space for filling the nan values

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])

#replacing + with space for filling the nan values

df1['Dependents']=df1['Dependents'].str.replace('+','')

df1['Dependents']=df1['Dependents'].fillna(df1['Dependents'].mode()[0])

df1['Self\_Employed']=df1['Self\_Employed'].fillna(df1['Self\_Employed'].mode()[0])

df1['LoanAmount']=df1['LoanAmount'].fillna(df1['LoanAmount'].mode()[0])

df1['Loan\_Amount\_Term']=df1['Loan\_Amount\_Term'].fillna(df1['Loan\_Amount\_Term'].mode()[0])

df1['Credit\_History']=df1['Credit\_History'].fillna(df1['Credit\_History'].mode()[0])

#changing the data type of each float column to int

from numpy import int64

df['Gender']=df['Gender'].astype('int64')

df['Married']=df['Married'].astype(int64)

df['Dependents']=df['Dependents'].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)

#changing the data type of each float column to int

from numpy import int64

df1['Gender']=df1['Gender'].astype('int64')

df1['Married']=df1['Married'].astype(int64)

df1['Dependents']=df1['Dependents'].astype(int64)

df1['Dependents']=df1['Dependents'].astype(int64)

df1['Self\_Employed']=df1['Self\_Employed'].astype(int64)

df1['CoapplicantIncome']=df1['CoapplicantIncome'].astype(int64)

df1['LoanAmount']=df1['LoanAmount'].astype(int64)

df1['Loan\_Amount\_Term']=df1['Loan\_Amount\_Term'].astype(int64)

df1['Credit\_History']=df1['Credit\_History'].astype(int64)

#dividing the dataset into dependent and independent y and x respectively

from imbalance.combine import SMOTETomek

smote=SMOTETomek(0.90)

y=df['Loan\_Status']

x=df.drop(columns=['Loan\_Status'],axis=1)

#creating a new x and y variables for the balanced set

x\_bal,y\_bal=smote.fit\_resample(x,y)

#printing the values of y before balancing the data and after

print(y.value\_counts())

print(y\_bal.value\_counts())

Y 422

N 192

Name: Loan\_Status, dtype: int64

df.describe()

| **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** |
| --- | --- | --- | --- | --- |
| **count** | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 |
| **mean** | 5403.459283 | 1621.245798 | 145.465798 | 342.410423 | 0.855049 |
| **std** | 6109.041673 | 2926.248369 | 84.180967 | 64.428629 | 0.352339 |
| **min** | 150.000000 | 0.000000 | 9.000000 | 12.000000 | 0.000000 |
| **25%** | 2877.500000 | 0.000000 | 100.250000 | 360.000000 | 1.000000 |
| **50%** | 3812.500000 | 1188.500000 | 125.000000 | 360.000000 | 1.000000 |
| **75%** | 5795.000000 | 2297.250000 | 164.750000 | 360.000000 | 1.000000 |
| **max** | 81000.000000 | 41667.000000 | 700.000000 | 480.000000 | 1.000000 |

df1.describe()

ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term Credit\_History

count 367.000000 367.000000 367.000000 367.000000 367.000000

mean 4805.599455 1569.577657 136.321526 342.822888 0.839237

std 4910.685399 2334.232099 60.967295 64.658402 0.367814

min 0.000000 0.000000 28.000000 6.000000 0.000000

25% 2864.000000 0.000000 101.000000 360.000000 1.000000

50% 3786.000000 1025.000000 126.000000 360.000000 1.000000

75% 5060.000000 2430.500000 157.500000 360.000000 1.000000

max 72529.000000 24000.000000 550.000000 480.000000 1.000000

#Data Visualization using distplot

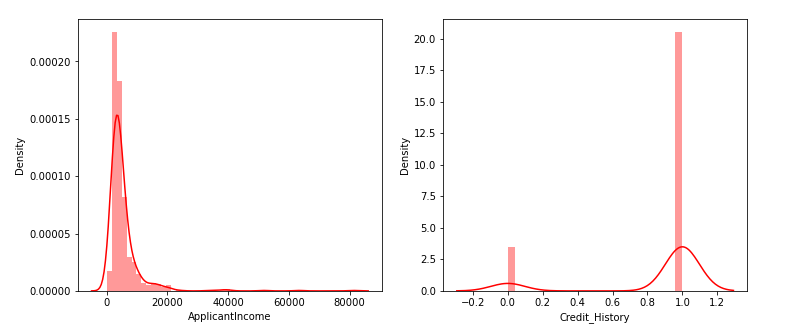
plt.figure(figsize = (12,5))

plt.subplot(121)

sns.distplot(df['ApplicantIncome'],color='r')

plt.subplot(122)

sns.distplot(df['Credit\_History'],color='r')



#Bivariate analysis

#Data Visualization using countplot

plt.figure(figsize = (18,4))

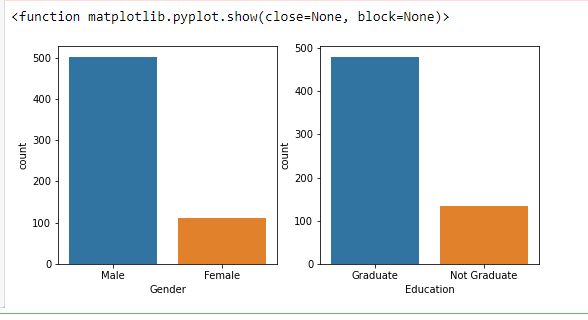
plt.subplot(1,4,1)

sns.countplot(df['Gender'])

plt.subplot(1,4,2)

sns.countplot(df['Education'])

plt.show



#Data Visualization using countplot

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

plt.subplot(131)

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

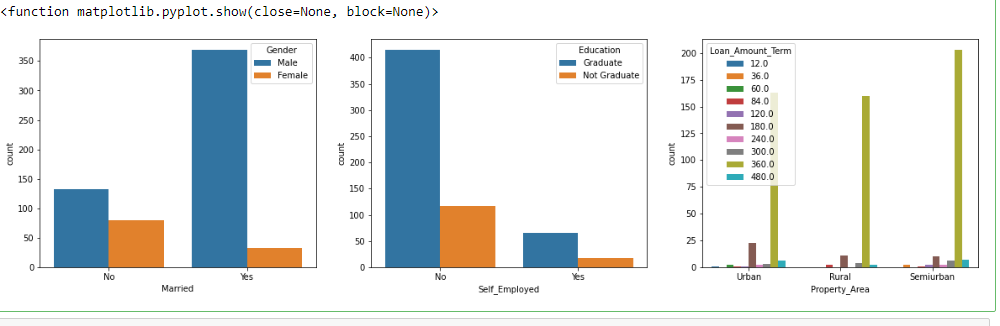
plt.subplot(132)

sns.countplot(df['Self\_Employed'],hue=df['Education'])

plt.subplot(133)

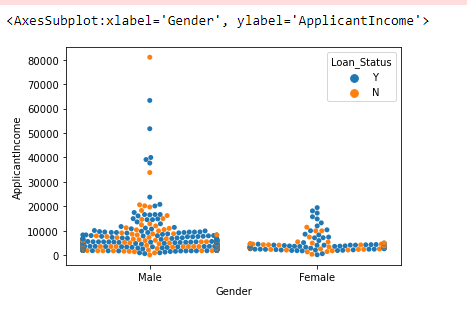
sns.countplot(df['Property\_Area'],hue=df['Loan\_Amount\_Term'])

plt.show



#visualized based gender and income what would be the application status

sns.swarmplot(df['Gender'],df['ApplicantIncome'],hue=df['Loan\_Status'])



Sclaing the Data

#performing feature scaling operation using standard scaller an x part of the dataset

#because there different types of values in the columns

sc=StandardScalaer()

x\_bal=sc.fit\_transform(x\_bal)

x\_bal=pd.df(x\_bal,columns=names)

#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)

# Create Decision Tree classifer object

clf = DecisionTreeClassifier()

# Train Decision Tree Classifer

clf = clf.fit(X\_train,y\_train)

#Predict the response for test dataset

y\_pred = clf.predict(X\_test)

# Model Accuracy, how often is the classifier correct?

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred)

from sklearn.tree import export\_graphviz

from sklearn.externals.six import StringIO

from IPython.display import Image

import pydotplus

dot\_data = StringIO()

export\_graphviz(clf, out\_file=dot\_data,

filled=True, rounded=True,

special\_characters=True,feature\_names = feature\_cols,class\_names=['0','1'])

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_png('diabetes.png')

Image(graph.create\_png())

from sklearn.ensemble import RandomForestClassifier

from sklearn.datasets import make\_classification

X, y = make\_classification(n\_samples=1000, n\_features=4,n\_informative=2, n\_redundant=0,random\_state=0, shuffle=False)

clf = RandomForestClassifier(max\_depth=2, random\_state=0)

clf.fit(X, y)

RandomForestClassifier()

print(clf.predict([[0, 0, 0, 0]]))

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import sklearn

dff=pd.read\_csv("E:\\NMDS\pl\_train.csv")

X = dff.iloc[:, [1, 2, 3]].values

y = dff.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Training the K-NN model on the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

ac = accuracy\_score(y\_test, y\_pred)