Project 3

Loan Approval System:

In a loan approval system, the typical setup involves using various factors or features to predict whether a loan application should be approved or denied.

**Code Snippet for data processing**:

import pandas as pd

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

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import OneHotEncoder

from sklearn.impute import SimpleImputer

# Load the dataset

data = pd.read\_csv('/content/sample\_data/Loan\_Train.csv')

# Display the first few rows of the dataset

print("First few rows of the dataset:")

print(data.head())

# Encoding categorical variables

label\_encoder = LabelEncoder()

data['Gender'] = label\_encoder.fit\_transform(data['Gender'])

data['Married'] = label\_encoder.fit\_transform(data['Married'])

data['Education'] = label\_encoder.fit\_transform(data['Education'])

data['Self\_Employed'] = label\_encoder.fit\_transform(data['Self\_Employed'])

data['Property\_Area'] = label\_encoder.fit\_transform(data['Property\_Area'])

data['Loan\_ID'] = label\_encoder.fit\_transform(data['Loan\_ID'])

data['Dependents'] = label\_encoder.fit\_transform(data['Dependents'])

X = data.drop('Loan\_Status', axis=1)

y = data['Loan\_Status']

# Handling missing values for numerical columns

numeric\_columns = X.select\_dtypes(include=['number']).columns

numeric\_imputer = SimpleImputer(strategy='mean')

X[numeric\_columns] = numeric\_imputer.fit\_transform(X[numeric\_columns])

# Handling missing values for categorical columns

categorical\_columns = X.select\_dtypes(include=['object']).columns

categorical\_imputer = SimpleImputer(strategy='most\_frequent')

# Encoding categorical variables using one-hot encoding

encoder = OneHotEncoder(sparse=False, handle\_unknown='ignore')

X\_encoded = pd.DataFrame(encoder.fit\_transform(X[categorical\_columns]))

# Concatenate the one-hot encoded features with the remaining numerical features

X\_encoded = pd.concat([X\_encoded, X.select\_dtypes(exclude=['object'])], axis=1)

# Splitting the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Display the shape of the training and testing sets

print("\nShape of training set (features):", X\_train.shape)

print("Shape of testing set (features):", X\_test.shape)

print("Shape of training set (target):", y\_train.shape)

print("Shape of testing set (target):", y\_test.shape)

le = LabelEncoder()

for column in data.columns:

if data[column].dtype == object:

data[column] = le.fit\_transform(data[column])

clf = RandomForestClassifier()

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

# Make predictions on the test set

predictions = clf.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, predictions)

print("Accuracy:", accuracy)

**Output:**

First few rows of the dataset:

Loan\_ID Gender Married Dependents Education Self\_Employed \

0 LP001002 Male No 0 Graduate No

1 LP001003 Male Yes 1 Graduate No

2 LP001005 Male Yes 0 Graduate Yes

3 LP001006 Male Yes 0 Not Graduate No

4 LP001008 Male No 0 Graduate No

ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term \

0 5849 0.0 NaN 360.0

1 4583 1508.0 128.0 360.0

2 3000 0.0 66.0 360.0

3 2583 2358.0 120.0 360.0

4 6000 0.0 141.0 360.0

Credit\_History Property\_Area Loan\_Status

0 1.0 Urban Y

1 1.0 Rural N

2 1.0 Urban Y

3 1.0 Urban Y

4 1.0 Urban Y

Shape of training set (features): (491, 12)

Shape of testing set (features): (123, 12)

Shape of training set (target): (491,)

Shape of testing set (target): (123,)

Accuracy: 0.7804878048780488

**Explanation:**

**Loading the Dataset**: It loads a CSV file named 'Loan\_Train.csv' using Pandas.

**Displaying the First Few Rows of the Dataset**: It prints the first few rows of the dataset to inspect the data structure.

*“print(data.head())”*

**Encoding Categorical Variables**: It encodes categorical variables into numerical values using Label Encoding. Each unique category in the categorical variables is assigned a unique integer.

First handling missing values separately for numeric and categorical columns using SimpleImputer.

Then encode categorical variables using one-hot encoding.

After that, concatenating the one-hot encoded features with the remaining numerical features.

**Separating Features and Target Variable**: It separates the features (independent variables) from the target variable (dependent variable).

Some Feature Variables (Input):

1. **Credit Score**: A measure of the creditworthiness of the applicant
2. **Loan Amount**: The amount of money the applicant is requesting as a loan.
3. **Employment History**: Length of employment and stability in the job.
4. **Education Level**: Education level of the applicant.
5. **Property Type**: Type of property the applicant intends to purchase (if applicable).
6. **Number of Dependents**: Number of dependents the applicant has.

In this case, the code is considering :

11 feature variables (Input):

Gender, Married, Dependents, Education, Self\_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term, Credit\_History, Property\_Area.

Target Variable (Output):

* **Loan \_Status**: A binary variable indicating whether the loan application was approved (1) or denied (0).

**Splitting the Dataset**: It splits the dataset into training and testing sets using the train\_test\_split function from scikit-learn. The training set contains 80% of the data, while the testing set contains 20% of the data.

**Displaying the Shape of the Sets**: It prints the shapes of the training and testing sets to verify the dimensions of the data.

**Instantiate the Random Forest Classifier**: This initializes the random forest model with default hyperparameters.

“*clf = RandomForestClassifier*()”

**Train the Classifier**: Here the fit() method is used to train the classifier on the training data (X\_train, y\_train). This means the model learns patterns and relationships in the training data that allow it to make predictions.

“*clf.fit(X\_train, y\_train)”*

**Make Predictions**: The predict() method is used to make predictions on the test set (X\_test). The model applies the learned patterns to the test data and predicts the target variable (Loan\_Status in this case).

“*predictions = clf.predict(X\_test)”*

**Calculate Accuracy**: After making predictions, compare the predicted labels (predictions) with the actual labels from the test set (y\_test). The accuracy\_score function from scikit-learn calculates the accuracy of the model on the test set.

“*accuracy = accuracy\_score(y\_test, predictions)”*

**Print Accuracy**: Print the accuracy of the model on the test set to the console.

“*print("Accuracy:", accuracy)*”

Displaying the shape of training and testing sets:

This provides valuable information about the sizes of the training and testing datasets and their respective feature matrices and target variables. It helps ensure that the data has been properly split and provides insights into the dimensions of the datasets being used for training and evaluation of the machine learning model.

E.g.: *“Shape of training set (target): (491,)*

*Shape of testing set (target): (123,)”*

Here,

"(491,)" indicates that there are 491 elements in the training set's target variable (y\_train).

"(123,)" indicates that there are 123 elements in the testing set's target variable (y\_test).

The output *"Accuracy: 0.7804878048780488"* indicates that the accuracy of the Random Forest classifier on the test set is approximately 0.7805(round off) or 78.05%.

Put differently, out of all the loan applications in the test set, the model correctly predicted the approval or denial status of approximately 78.05% of them.