**import** pandas **as** pd**import** plotly.graph\_objects **as** go**import** plotly.express **as** px**import** plotly.io **as** piopio**.**templates**.**default **=** "plotly\_white"

data **=** pd**.**read\_csv("C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/Credit Scoring/credit\_scoring.csv")print(data**.**head())

def calculate\_credit\_score(age, income, debt, credit\_history, employment\_status):

# This function takes various parameters as inputs and calculates a credit score.

# Define weights for each parameter

age\_weight = 0.2

income\_weight = 0.3

debt\_weight = 0.2

credit\_history\_weight = 0.2

employment\_status\_weight = 0.1

# Calculate the weighted sum

weighted\_sum = (age \* age\_weight) + (income \* income\_weight) - (debt \* debt\_weight) + (credit\_history \* credit\_history\_weight) + (employment\_status \* employment\_status\_weight)

# Map the weighted sum to a credit score scale (e.g., 350 to 800)

min\_score = 350

max\_score = 800

# Ensure the calculated score is within the defined range

credit\_score = max(min\_score, min(max\_score, weighted\_sum))

return credit\_score

# Sample usage

applicant\_age = 30

applicant\_income = 60000

applicant\_debt = 10000

applicant\_credit\_history = 3 # Years of credit history

applicant\_employment\_status = 1 # Employed (1) or unemployed (0)

# Get credit score

result = calculate\_credit\_score(applicant\_age, applicant\_income, applicant\_debt, applicant\_credit\_history, applicant\_employment\_status)

# Print the result

print("Credit Score: {result}")

import pandas as pd

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

# Load the dataset

credit\_data = pd.read\_csv('credit\_data.csv')

# Handle missing values

credit\_data = credit\_data.dropna()

# Encode categorical variables

credit\_data = pd.get\_dummies(credit\_data, columns=['education', 'employment\_status'])

# Normalize numerical features

credit\_data['income'] = (credit\_data['income'] - credit\_data['income'].mean()) / credit\_data['income'].std()

# Split the data into features and target variable

X = credit\_data.drop('target\_variable', axis=1)

y = credit\_data['target\_variable']

# Split the data 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)

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

model = RandomForestClassifier()

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

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

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

print(f'Model Accuracy: {accuracy}')

from flask import Flask, render\_template, request

import pickle

import numpy as np

app = Flask(\_\_name\_\_)

# Load the pre-trained ML model (LightGBM or XGBoost)

model\_path = "model/credit\_model.pkl"

model = pickle.load(open(model\_path, "rb"))

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

if request.method == 'POST':

# Collect form data

income = float(request.form['income'])

debt = float(request.form['debt'])

credit\_history = int(request.form['credit\_history'])

repayment\_behavior = float(request.form['repayment\_behavior'])

# Prepare data for prediction

input\_data = np.array([[income, debt, credit\_history, repayment\_behavior]])

prediction = model.predict(input\_data)[0]

# Map prediction to creditworthiness

if prediction == 1:

result = "Creditworthy"

else:

result = "Not Creditworthy"

return render\_template('result.html', prediction=result)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Credit Scoring System</title>

<link rel="stylesheet" href="/static/style.css">

</head>

<body>

<div class="container">

<h1>Credit Scoring System</h1>

<form action="/predict" method="post">

<label for="income">Monthly Income:</label>

<input type="number" step="0.01" name="income" id="income" required>

<label for="debt">Debt-to-Income Ratio:</label>

<input type="number" step="0.01" name="debt" id="debt" required>

<label for="credit\_history">Credit History (Years):</label>

<input type="number" name="credit\_history" id="credit\_history" required>

<label for="repayment\_behavior">Repayment Behavior (Score):</label>

<input type="number" step="0.01" name="repayment\_behavior" id="repayment\_behavior" required>

<button type="submit">Check Creditworthiness</button>

</form>

</div>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Result</title>

<link rel="stylesheet" href="/static/style.css">

</head>

<body>

<div class="container">

<h1>Credit Scoring System</h1>

<p>The individual is <strong>{{ prediction }}</strong>.</p>

<a href="/">Check Another</a>

</div>

</body>

</html>