**BACKEND**

import pandas as pd

import pickle

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

# Assuming you have a pandas DataFrame df loaded from your dataset

# Example:

df = pd.read\_csv('customer.csv')

# Preprocessing

df['payment\_mode'] = df['payment\_mode'].astype(str) # Ensuring it's a string for LabelEncoder

# Use LabelEncoder to encode 'payment\_mode'

label\_encoder = LabelEncoder()

df['payment\_mode'] = label\_encoder.fit\_transform(df['payment\_mode'])

# Define features (X) and target (y)

X = df[['quantity', 'payment\_mode', 'Rating']]

y = df['order\_status'] # Assuming order\_status is the target, 1 for fraudulent, 0 for not

# Split the data

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

# Train a RandomForestClassifier

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

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

# Save the model and label encoder

with open('model.pkl', 'wb') as model\_file:

pickle.dump(model, model\_file)

with open('label\_encoder.pkl', 'wb') as le\_file:

pickle.dump(label\_encoder, le\_file)

print("Model and label encoder saved!")

**FLASK**

from flask import Flask, render\_template, request

import pickle

import numpy as np

import traceback # For better error logging

app = Flask(\_name\_)

# Load the model and label encoder

try:

with open('model.pkl', 'rb') as model\_file:

model = pickle.load(model\_file)

with open('label\_encoder.pkl', 'rb') as le\_file:

label\_encoder = pickle.load(le\_file)

except Exception as e:

print(f"Error loading model or label encoder: {e}")

raise

@app.route('/')

def index():

return render\_template('index.html')

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

def predict():

try:

# Extract input features from the form

quantity = int(request.form.get('quantity'))

payment\_mode = request.form.get('payment\_mode')

rating = float(request.form.get('Rating')) # Using 'Rating' with uppercase R

# Debugging: Print the inputs for checking

print(f"Received inputs - Quantity: {quantity}, Payment Mode: {payment\_mode}, Rating: {Rating}")

# Encode the payment\_mode using the LabelEncoder

try:

payment\_mode\_encoded = label\_encoder.transform([payment\_mode])[0]

print(f"Encoded Payment Mode: {payment\_mode\_encoded}") # Debugging encoded value

except Exception as e:

print(f"Error encoding payment mode: {e}")

raise ValueError(f"Invalid payment mode: {payment\_mode}")

# Prepare the input for the model (reshape for single sample)

# Ensure the order of features matches your training data

final\_input = np.array([quantity, payment\_mode\_encoded, rating]).reshape(1, -1)

# Debugging: Print the input array for prediction

print(f"Final input for model prediction: {final\_input}")

# Predict the fraud probability

prediction = model.predict\_proba(final\_input)

fraud\_probability = prediction[0][1]

# Format the result for display

output = '{0:.{1}f}'.format(fraud\_probability, 2)

# Generate result message based on probability

if fraud\_probability > 0.5:

result\_message = f"Warning: High likelihood of fraud! Fraud probability is {output}."

advice = "Consider verifying this order."

else:

result\_message = f"Low likelihood of fraud. Fraud probability is {output}."

advice = "The order appears to be legitimate."

# Render the result template

return render\_template('index.html', pred=result\_message, advice=advice)

except Exception as e:

# Log the exception with traceback for debugging

print(f"Error occurred: {e}")

print("Full error traceback:", traceback.format\_exc()) # Display the full error traceback

return render\_template('index.html', pred="An error occurred while processing your input.", advice="Please check your inputs and try again.")

if \_name\_ == "\_main\_":

app.run(debug=True)

**FRONTEND**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Fraud Detection System</title>

<link href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css" rel="stylesheet">

<style>

/\* Set background image \*/

body {

display: flex;

min-height: 100vh;

flex-direction: column;

background-image: url('{{ url\_for("static", filename="ba.jpg") }}'); /\* Reference the image in the static folder \*/

background-size: cover; /\* Ensure the image covers the entire page \*/

background-position: center; /\* Center the background image \*/

background-attachment: fixed; /\* Keep the background fixed when scrolling \*/

}

main {

flex: 1 0 auto;

padding-top: 20px;

}

/\* Customize the colors for high-risk and low-risk sections \*/

.high-risk {

background-color: rgba(255, 205, 210, 0.8);

color: #b71c1c;

}

.low-risk {

background-color: rgba(200, 230, 201, 0.8);

color: #1b5e20;

}

/\* Result Section Styling \*/

.result-section {

padding: 20px;

margin-top: 30px;

border-radius: 5px;

border: 1px solid #ddd;

}

.btn-large {

width: 50%;

}

/\* Ensure all text is readable over the background \*/

.blue-text {

color: #007bb5 !important;

}

/\* Adjust text for headers \*/

h2 {

color: #ffffff;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5); /\* Optional: adds shadow for better readability \*/

}

/\* Customize the navigation bar to fit the theme \*/

nav {

background-color: rgba(0, 0, 0, 0.7); /\* Semi-transparent black \*/

}

.brand-logo {

color: #ffffff;

}

/\* Set text color to black for form fields labels and inputs \*/

input[type="number"], input[type="text"], input[type="password"], select {

color: black !important; /\* Ensure input text is black \*/

}

label {

color: black !important; /\* Change label text to black \*/

}

/\* Optional: Adjust label text for focus state \*/

input[type="number"]:focus + label, input[type="text"]:focus + label, input[type="password"]:focus + label, select:focus + label {

color: #007bb5 !important; /\* Optional: color change for focused labels \*/

}

input[type="number"]:focus, input[type="text"]:focus, input[type="password"]:focus, select:focus {

border-bottom: 2px solid #007bb5 !important; /\* Focused input border \*/

box-shadow: 0 1px 0 0 #007bb5 !important; /\* Optional: adds blue shadow to the focused input \*/

}

</style>

</head>

<body>

<!-- Navigation Bar -->

<nav class="blue lighten-1">

<div class="nav-wrapper container">

</div>

</nav>

<!-- Main Content -->

<main class="container">

<div class="section">

<h2 class="center blue-text">Order Fraud Detection</h2>

<!-- Form for entering order data -->

<div class="row">

<form action="/predict" method="post" class="col s12">

<div class="row">

<!-- Order Quantity -->

<div class="input-field col s12 m3">

<input id="quantity" name="quantity" type="number" min="1" required>

<label for="quantity">Order Quantity</label>

</div>

<!-- Payment Mode -->

<div class="input-field col s12 m3">

<select name="payment\_mode" required>

<option value="" disabled selected>Payment Mode</option>

<option value="Credit Card">Credit Card</option>

<option value="Debit Card">Debit Card</option>

<option value="Cash">Cash</option>

<option value="Bank Transfer">Bank Transfer</option>

<option value="PayPal">PayPal</option>

</select>

<label>Payment Mode</label>

</div>

<!-- Customer Rating -->

<div class="input-field col s12 m3">

<input id="rating" name="rating" type="number" min="1" max="5" step="1" required>

<label for="rating">Customer Rating (1-5)</label>

</div>

<!-- Transaction Amount -->

<div class="input-field col s12 m3">

<input id="transaction\_amount" name="transaction\_amount" type="number" step="0.01" min="0" required>

<label for="transaction\_amount">Transaction Amount</label>

</div>

</div>

<!-- Submit Button -->

<div class="row center">

<button type="submit" class="btn-large blue">Detect Fraud</button>

</div>

</form>

</div>

<!-- Prediction Results (Dynamic Content) -->

{% if pred %}

<div class="row result-section {{ 'high-risk' if 'High' in pred else 'low-risk' }}">

<div class="col s12">

<h5><b>Prediction Result:</b></h5>

<p>{{ pred }}</p>

<p>{{ advice }}</p>

</div>

</div>

{% endif %}

</div>

</main>

<!-- Footer -->

<footer class="page-footer blue">

<div class="footer-copyright">

<div class="container">© 2024 Fraud Detection System</div>

</div>

</footer>

<!-- Materialize JavaScript -->

<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>

<script>

// Initialize Materialize Select Elements

document.addEventListener('DOMContentLoaded', function() {

var elems = document.querySelectorAll('select');

M.FormSelect.init(elems);

});

</script>

</body>

</html>