**Source code**

**Base.html:**

<!DOCTYPE html>

<html>

<head>

<title>New Template</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Raleway">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>

body,h1 {font-family: "Raleway", Arial, sans-serif}

h1 {letter-spacing: 6px}

.w3-row-padding img {margin-bottom: 12px}

.w3-button {

background-color: black !important; /\* Change to a thicker green \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

transition-duration: 0.4s;

cursor: pointer;

}

.w3-button:hover {

background-color: yellowgreen !important;

color: white !important;

border: 2px solid black !important;

}

.w4-button:hover {

background-color: yellowgreen !important;

color: black !important;

border: 2px solid white !important;

}

.w3-xlarge{

color:black

}

.ntg {

background-image: url(https://i.pinimg.com/736x/5d/8c/39/5d8c39a71a34b459f39cf73446de5fec.jpg);

background-repeat: no-repeat;

background-size: cover; /\* Ensures the image covers the entire screen \*/

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

height: 100vh; /\* Sets the height to full screen \*/

width: 100vw; /\* Sets the width to full screen \*/

}

</style>

</head>

<body class="ntg">

<!-- !PAGE CONTENT! -->

<div class="w3-content" style="max-width:1500px">

<!-- Header -->

<header class="w3-panel w3-center w3-opacity" style="padding:128px 16px">

<div class="w3-xlarge">

<h1><b>flight delay predection</b></h1>

<div class="w3-padding-32">

<div class="w3-bar w3-border">

<a href="{% url 'index' %}" class="w3-bar-item w3-button">Home</a>

<a href="{% url 'userlogin' %}" class="w3-bar-item w4-button w4-light-grey">User Login</a>

<a href="{% url 'adminlogin' %}" class="w3-bar-item w3-button">Admin Login</a>

<a href="{% url 'register' %}" class="w3-bar-item w3-button w3-hide-small">Register</a>

</div>

</div>

</div>

</header>

<hr>

{% block content %}{% endblock %}

<hr><br>

<footer class="w3-container w3-padding-64 w3-light-grey w3-center w3-large">

<i class="fa fa-facebook-official w3-hover-opacity"></i>

<i class="fa fa-instagram w3-hover-opacity"></i>

<i class="fa fa-snapchat w3-hover-opacity"></i>

<i class="fa fa-pinterest-p w3-hover-opacity"></i>

<i class="fa fa-twitter w3-hover-opacity"></i>

<i class="fa fa-linkedin w3-hover-opacity"></i>

</footer>

</body>

</html>

**Views.py:**

from django.shortcuts import render,redirect

from admins.views import viewuser

from django.contrib import messages

from .models import Reg\_User

# Create your views here.

def index(request):

return render(request, 'base.html')

def register(request):

return render(request, 'user\_register.html')

def ActivateUser(request, id):

if request.method == 'GET':

if id is not None:

status = 'Activated'

print("PID = ", id, status)

Reg\_User.objects.filter(id=id).update(status=status)

return redirect(viewuser)

def BlockUser(request, id):

if request.method == 'GET':

if id is not None:

status = 'Waiting'

print("PID = ", id, status)

Reg\_User.objects.filter(id=id).update(status=status)

return redirect(viewuser)

def userbase(request):

return render(request,'users/userbase.html')

def userlogin(request):

if request.method == 'POST':

username = request.POST.get('uname')

password = request.POST.get('psw')

print(f"Username: {username}, Password: {password}")

try:

data=Reg\_User.objects.get(username=username, password=password)

if data.status == 'Activated':

return redirect('userbase')

else:

messages.error(request, 'Invalid Credentials')

except Exception as e:

print(f'the exception is {e}')

messages.error(request, f'Invalid Credentials {str(e)}')

return render(request, 'userlogin.html')

#========================================================

import pandas as pd

from sklearn.preprocessing import LabelEncoder

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

from matplotlib import pyplot as plt

import urllib

import io

import base64

import numpy as np

import seaborn as sns

data=pd.read\_csv(r'C:\Users\chund\Desktop\New folder (3)\template2\template2\media\FlightDelay.csv')

data.fillna('1')

le=LabelEncoder()

data['FL\_DATE']=le.fit\_transform(data['FL\_DATE'])

data['OP\_UNIQUE\_CARRIER']=le.fit\_transform(data['OP\_UNIQUE\_CARRIER'])

data['OP\_CARRIER']=le.fit\_transform(data['OP\_CARRIER'])

data['TAIL\_NUM']=le.fit\_transform(data['TAIL\_NUM'])

data['ORIGIN']=le.fit\_transform(data['ORIGIN'])

data['DEST']=le.fit\_transform(data['DEST'])

data['DEP\_TIME']=le.fit\_transform(data['DEP\_TIME'])

data['TAXI\_OUT']=le.fit\_transform(data['TAXI\_OUT'])

data['WHEELS\_OFF']=le.fit\_transform(data['WHEELS\_OFF'])

data['WHEELS\_ON']=le.fit\_transform(data['WHEELS\_ON'])

data['TAXI\_IN']=le.fit\_transform(data['TAXI\_IN'])

data['ARR\_TIME']=le.fit\_transform(data['ARR\_TIME'])

data['CANCELLED']=le.fit\_transform(data['CANCELLED'])

data['DISTANCE']=le.fit\_transform(data['DISTANCE'])

data['ARR\_DELAY\_GROUP']=le.fit\_transform(data['ARR\_DELAY\_GROUP'])

x = data.drop(['ARR\_DELAY\_GROUP'], axis=1)

y = data['ARR\_DELAY\_GROUP']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=42)

lr=LogisticRegression()

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

p=lr.predict(X\_test)

def training(request):

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

sns.heatmap(cm,annot=True)

plt.savefig('cm.png')

plt.title('Confusion Matrix')

plt.xlabel('Predicted Values')

plt.ylabel('Actual Values')

plt.show()

acc=accuracy\_score(p,y\_test)

return render(request, "users/training.html", {

'accuracy': acc,

})

#==================================================================

import pandas as pd

from django.shortcuts import render

import pandas as pd

def prediction(request):

# Assuming 'lr' is your pre-trained Logistic Regression model

if request.method == 'POST':

# Collecting data from the HTML form

input\_data = [

request.POST.get('FL\_DATE'),

request.POST.get('OP\_UNIQUE\_CARRIER'),

request.POST.get('OP\_CARRIER'),

request.POST.get('TAIL\_NUM'),

request.POST.get('OP\_CARRIER\_FL\_NUM'),

request.POST.get('ORIGIN\_AIRPORT\_ID'),

request.POST.get('ORIGIN'),

request.POST.get('DEST\_AIRPORT\_ID'),

request.POST.get('DEST'),

request.POST.get('CRS\_DEP\_TIME'),

request.POST.get('DEP\_TIME'),

request.POST.get('TAXI\_OUT'),

request.POST.get('WHEELS\_OFF'),

request.POST.get('WHEELS\_ON'),

request.POST.get('TAXI\_IN'),

request.POST.get('CRS\_ARR\_TIME'),

request.POST.get('ARR\_TIME'),

request.POST.get('CANCELLED'),

request.POST.get('DISTANCE')

]

# Creating a DataFrame from the input

input\_df = pd.DataFrame([input\_data])

# Convert all values to numeric where possible, fill non-convertible values with NaN, then replace NaN with 0

input\_df = input\_df.apply(pd.to\_numeric, errors='coerce').fillna(0)

print(input\_df)

# Predicting using the model

try:

op = lr.predict(input\_df)

except Exception as e:

return render(request, 'users/predictForm.html', {'output': f"Error in prediction: {str(e)}"})

# Interpreting the prediction result

if op[0] == 0:

op = "early\_arrival"

elif op[0] == 1:

op = "ontime"

else:

op = "delayed"

# Render the result to the HTML template

return render(request, 'users/predictForm.html', {'output': op})

# If the request is not POST, render the form page

return render(request, 'users/predictForm.html')