**SAMPLE CODE**

**USER VIEWS**

from ast import alias

from concurrent.futures import process

from django.shortcuts import render

# Create your views here.

from django.shortcuts import render, HttpResponse

from django.contrib import messages

import Employee\_promotion\_prediction

from .forms import UserRegistrationForm

from .models import UserRegistrationModel

from django.conf import settings

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.ticker as plticker

import datetime as dt

from sklearn import preprocessing, metrics

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

from sklearn.preprocessing import OneHotEncoder

from sklearn.linear\_model import LinearRegression

from sklearn import metrics

from sklearn.metrics import classification\_report

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score,  confusion\_matrix, roc\_curve, auc

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import OneHotEncoder

df = pd.read\_csv(r"C:\Users\lenovo\Desktop\nandini\Employee\_promotion\_prediction\media\train.csv")

df.fillna(0,inplace=True)

# Separate features and target

X=df[['department','region','education','gender','recruitment\_channel','no\_of\_trainings','age','previous\_year\_rating','length\_of\_service','awards\_won','avg\_training\_score']]

y = df['is\_promoted']

X = pd.get\_dummies(X, drop\_first=True)

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

rf\_classifier = RandomForestClassifier()

rf\_classifier.fit(X\_train, y\_train)

y\_pred = rf\_classifier.predict(X\_test)

# Create your views here.

def UserRegisterActions(request):

    if request.method == 'POST':

        form = UserRegistrationForm(request.POST)

        if form.is\_valid():

            print('Data is Valid')

            form.save()

            messages.success(request, 'You have been successfully registered')

            form = UserRegistrationForm()

            return render(request, 'UserRegistrations.html', {'form': form})

        else:

            messages.success(request, 'Email or Mobile Already Existed')

            print("Invalid form")

    else:

        form = UserRegistrationForm()

    return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):

    if request.method == "POST":

        loginid = request.POST.get('loginid')

        pswd = request.POST.get('pswd')

        print("Login ID = ", loginid, ' Password = ', pswd)

        try:

            check = UserRegistrationModel.objects.get(

                loginid=loginid, password=pswd)

            status = check.status

            print('Status is = ', status)

            if status == "activated":

                request.session['id'] = check.id

                request.session['loggeduser'] = check.name

                request.session['loginid'] = loginid

                request.session['email'] = check.email

                print("User id At", check.id, status)

                return render(request, 'users/UserHomePage.html', {})

            else:

                messages.success(request, 'Your Account Not at activated')

                return render(request, 'UserLogin.html')

        except Exception as e:

            print('Exception is ', str(e))

            pass

        messages.success(request, 'Invalid Login id and password')

    return render(request, 'UserLogin.html', {})

def UserHome(request):

    return render(request, 'users/UserHomePage.html', {})

def DatasetView(request):

    path = settings.MEDIA\_ROOT + "//" + 'train.csv'

    df = pd.read\_csv(path)

    df = df.to\_html

    return render(request, 'users/viewdataset.html', {'data': df})

def Training(request):

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

    print(f'Random Forest Accuracy: {accuracy:.2f}')

    report = classification\_report(y\_test, y\_pred)

    # Confusion matrix

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

    plt.figure(figsize=(8, 6))

    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)

    plt.title('Confusion Matrix')

    plt.xlabel('Predicted')

    plt.ylabel('Actual')

    plt.show()

    # Predict labels for both training and test sets

    y\_train\_pred = rf\_classifier.predict(X\_train)

    y\_test\_pred = rf\_classifier.predict(X\_test)

    # Calculate accuracies

    train\_acc = accuracy\_score(y\_train, y\_train\_pred)

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

    # Print the accuracies

    print(f"Train Accuracy: {train\_acc \* 100: .2f}%")

    print(f"Test Accuracy: {test\_acc \* 100: .2f}%")

    # Example values for accuracy and loss plots

    train\_accuracy = [0.8, 0.85, 0.88, 0.90]  # Example values

    val\_accuracy = [0.78, 0.82, 0.85, 0.87]   # Example values

    train\_loss = [0.5, 0.4, 0.35, 0.3]        # Example values

    val\_loss = [0.55, 0.45, 0.4, 0.38]        # Example values

    epochs = range(1, len(train\_accuracy) + 1)

    # Plot accuracy

    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)

    plt.plot(epochs, train\_accuracy, label='Train Accuracy')

    plt.plot(epochs, val\_accuracy, label='Validation Accuracy')

    plt.title('Train vs Validation Accuracy')

    plt.xlabel('Epochs')

    plt.ylabel('Accuracy')

    plt.legend()

    # Plot loss

    plt.subplot(1, 2, 2)

    plt.plot(epochs, train\_loss, label='Train Loss')

    plt.plot(epochs, val\_loss, label='Validation Loss')

    plt.title('Train vs Validation Loss')

    plt.xlabel('Epochs')

    plt.ylabel('Loss')

    plt.legend()

    plt.tight\_layout()

    plt.show()

    # ROC curve and AUC score

    y\_pred\_proba = rf\_classifier.predict\_proba(X\_test)[:, 1]  # Get probabilities for the positive class

    # Compute ROC curve

    fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_proba)

    roc\_auc = auc(fpr, tpr)

    # Plot ROC curve

    plt.figure(figsize=(8, 6))

    plt.plot(fpr, tpr, color='blue', label='ROC Curve (area = {:.2f})'.format(roc\_auc))

    plt.plot([0, 1], [0, 1], color='red', linestyle='--')  # Diagonal line

    plt.title('Receiver Operating Characteristic (ROC) Curve')

    plt.xlabel('False Positive Rate')

    plt.ylabel('True Positive Rate')

    plt.legend()

    plt.show()

    # Pass results to the template

    context = {

       'train\_acc': train\_acc,

       'test\_acc': test\_acc,

    }

    return render(request, 'users/ml.html', context)

def Prediction(request):

    if request.method == "POST":

        dept = request.POST.get('department')

        re = request.POST.get('region')

        ge = request.POST.get('gender')

        rc = request.POST.get('recruitment\_channel')

        nt = request.POST.get('no\_of\_trainings')

        ag = request.POST.get('age')

        pyr = request.POST.get('previous\_year\_rating')

        los = request.POST.get('length\_of\_service')

        ao = request.POST.get('awards\_won')

        ats = request.POST.get('avg\_training\_score')

        nt = int(nt) if nt else 0

        ag = int(ag) if ag else 0

        pyr = float(pyr) if pyr else 0.0

        los = int(los) if los else 0

        ats = float(ats) if ats else 0.0

        ao = 1 if ao == 'Yes' else 0

        gender\_mapping = {'Male': 1, 'Female': 0, 'Other': 2}

        ge = gender\_mapping.get(ge, 2)  # Default to 'Other' (2) if not Male or Female

        region\_mapping = {'region\_1': 0, 'region\_2': 1, 'region\_3': 2}

from django.shortcuts import render

from django.contrib import messages

from users.forms import UserRegistrationForm

from users.models import UserRegistrationModel

# Create your views here.

def AdminLoginCheck(request):

    if request.method == 'POST':

        usrid = request.POST.get('loginid')

        pswd = request.POST.get('pswd')

        print("User ID is = ", usrid)

        if usrid == 'admin' and pswd == 'admin':

            return render(request, 'admins/AdminHome.html')

        else:

            messages.success(request, 'Please Check Your Login Details')

    return render(request, 'AdminLogin.html', {})

def AdminHome(request):

    return render(request, 'admins/AdminHome.html',{})

def RegisterUsersView(request):

    data = UserRegistrationModel.objects.all()

    return render(request,'admins/viewregisterusers.html',{'data':data})

def ActivaUsers(request):

    if request.method == 'GET':

        id = request.GET.get('uid')

        status = 'activated'

        print("PID = ", id, status)

        UserRegistrationModel.objects.filter(id=id).update(status=status)

        data = UserRegistrationModel.objects.all()

        return render(request,'admins/viewregisterusers.html',{'data':data})

from django.shortcuts import render

from users.forms import UserRegistrationForm

def index(request):

    return render(request, 'index.html', {})

def AdminLogin(request):

    return render(request, 'AdminLogin.html', {})

def UserLogin(request):

    return render(request, 'UserLogin.html', {})

def UserRegister(request):

    form = UserRegistrationForm()

    return render(request, 'UserRegistrations.html', {'form': form})

**USERS(BASE.HTML)**

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="utf-8">

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

  <title>Arsha Bootstrap Template - Index</title>

  <meta content="" name="description">

  <meta content="" name="keywords">

  <!-- Favicons -->

  <link href="{% static 'assets/img/favicon.png' %}" rel="icon">

  <link href="{% static 'assets/img/apple-touch-icon.png' %}" rel="apple-touch-icon">

  <!-- Google Fonts -->

  <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Jost:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">

  <!-- Vendor CSS Files -->

  <link href="{% static 'assets/vendor/aos/aos.css' %}" rel="stylesheet">

  <link href="{% static 'assets/vendor/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">

  <link href="{% static 'assets/vendor/bootstrap-icons/bootstrap-icons.css' %}" rel="stylesheet">

  <link href="{% static 'assets/vendor/boxicons/css/boxicons.min.css' %}" rel="stylesheet">

  <link href="{% static 'assets/vendor/glightbox/css/glightbox.min.css' %}" rel="stylesheet">

  <link href="{% static 'assets/vendor/remixicon/remixicon.css' %}" rel="stylesheet">

  <link href="{% static 'assets/vendor/swiper/swiper-bundle.min.css' %}" rel="stylesheet">

  <!-- Template Main CSS File -->

  <link href="{% static 'assets/css/style.css' %}" rel="stylesheet">

  <!-- =======================================================

  \* Template Name: Arsha - v4.11.0

  \* Template URL: https://bootstrapmade.com/arsha-free-bootstrap-html-template-corporate/

  \* Author: BootstrapMade.com

  \* License: https://bootstrapmade.com/license/

  ======================================================== -->

</head>

<body>

    <style>

        span{

            color:pink;

        }

        header{

            background-color:black;

        }

        body{

             background-image: url('{% static 'assets\C:\Users\lenovo\Desktop\nandini\Employee\_promotion\_prediction\static\images\img2.jpg' %}'); \*/

        }

    </style>

  <!-- ======= Header ======= -->

  <header id="header" class="fixed-top ">

    <div class="container d-flex align-items-center">

      <h1 class="logo me-auto"><a href="index.html"><span>Analysis and Prediction of Employee Promotions

        Using Machine Learning</span></a></h1>

      <!-- Uncomment below if you prefer to use an image logo -->

      <!-- <a href="index.html" class="logo me-auto"><img src="assets/img/logo.png" alt="" class="img-fluid"></a>-->

      <nav id="navbar" class="navbar">

        <ul>

          <li><a class="nav-link scrollto active" href="{% url 'index' %}">Home</a></li>

          <li><a class="nav-link scrollto" href="{% url 'AdminLogin' %}">Admin</a></li>

          <li><a class="nav-link scrollto" href="{% url 'UserLogin' %}">User</a></li>

          <li><a class="nav-link   scrollto" href="{% url 'UserRegister' %}">Register</a></li>

        </ul>

        <i class="bi bi-list mobile-nav-toggle"></i>

      </nav><!-- .navbar -->

    </div>

  </header><!-- End Header -->

{% block contents %}

{% endblock %}

  <!-- ======= Footer ======= -->

  <footer id="footer">

  </footer><!-- End Footer -->

  <div id="preloader"></div>

  <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

  <!-- Vendor JS Files -->

  <script src="{% static 'assets/vendor/aos/aos.js' %}"></script>

  <script src="{% static 'assets/vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>

  <script src="{% static 'assets/vendor/glightbox/js/glightbox.min.js' %}"></script>

  <script src="{% static 'assets/vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>

  <script src="{% static 'assets/vendor/swiper/swiper-bundle.min.js' %}"></script>

  <script src="{% static 'assets/vendor/waypoints/noframework.waypoints.js' %}"></script>

  <script src="{% static 'assets/vendor/php-email-form/validate.js' %}"></script>

  <!-- Template Main JS File -->

  <script src="{% static 'assets/js/main.js' %}"></script>

</body>

</html>