SAMPLE CODE:

**User side views:**

from django.shortcuts import render, HttpResponse

from .forms import UserRegistrationForm

from django.contrib import messages

from .models import UserRegistrationModel

from django.conf import settings

import pandas as pd

from django.core.files.storage import FileSystemStorage

path=settings.MEDIA\_ROOT + "//" + 'Luke\_hair\_loss.csv'

df = pd.read\_csv(path)

# 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")

password=request.POST.get("pswd")

print(loginid)

print(password)

try:

check=UserRegistrationModel.objects.get(loginid=loginid,password=password)

status=check.status

if status=="activated":

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

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

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

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

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

else:

messages.success(request,"your account not activated")

return render(request,"UserLogin.html")

except Exception as e:

print('=======>',e)

messages.success(request,'invalid details')

return render(request,'UserLogin.html')

def UserHome(request):

return render(request,"users/UserHome.html",{})

import pandas as pd

from django.shortcuts import render

def hair\_data\_view(request):

# Assuming you have a dataset in CSV format named "hair\_data.csv"

# and it is located in the media folder of your Django project.

path = settings.MEDIA\_ROOT + "//" + 'Luke\_hair\_loss.csv'

# Read the dataset into a pandas DataFrame

df = pd.read\_csv(path)

# Drop the "hair\_loss" column

df = df.drop(columns=['hair\_loss'])

print("=====================done=========================")

# Convert the DataFrame to a list of dictionaries for template rendering

data = df.to\_dict(orient='records')

# Define column names for the template

columns = df.columns

return render(request, 'users/dataset.html', {'data': data, 'columns': columns})

import pandas as pd

from django.shortcuts import render

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

from django.conf import settings

# Define a function to preprocess the data

def preprocess\_data(df):

# Handle null values (you can choose an appropriate strategy)

df = df.dropna()

# Apply label encoding to categorical columns if needed

label\_encoder = LabelEncoder()

df['shampoo\_brand'] = label\_encoder.fit\_transform(df['shampoo\_brand'])

df['stress\_level'] = label\_encoder.fit\_transform(df['stress\_level'])

df['pressure\_level'] = label\_encoder.fit\_transform(df['pressure\_level'])

df['school\_assesssment'] = label\_encoder.fit\_transform(df['school\_assesssment'])

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

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

df['hair\_washing'] = label\_encoder.fit\_transform(df['hair\_washing'])

return df

# Define a function to train the K-NN classifier

def knn\_classification(request):

# Assuming you have a dataset in CSV format named "hair\_data.csv"

# and it is located in the media folder of your Django project.

path = settings.MEDIA\_ROOT + "//" + 'Luke\_hair\_loss.csv'

# Read the dataset into a pandas DataFrame

df = pd.read\_csv(path)

# Preprocess the data

df = preprocess\_data(df)

# Split the data into features (X) and the target variable (y)

X = df[['stay\_up\_late', 'pressure\_level', 'coffee\_consumed', 'brain\_working\_duration',

'school\_assesssment', 'swimming',

'hair\_washing', 'hair\_grease', 'dandruff', 'libido']]

y = df['hair\_loss']

# Split the data into a training set and a test set

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

# # Feature scaling (optional but can be beneficial for k-NN)

# scaler = StandardScaler()

# X\_train = scaler.fit\_transform(X\_train)

# X\_test = scaler.transform(X\_test)

# Create a k-NN classifier (you can adjust the number of neighbors, 'n\_neighbors')

knn\_classifier = KNeighborsClassifier(n\_neighbors=5)

# Train the k-NN classifier on the training data

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

# Return the trained classifier

return knn\_classifier

# Assuming you have a DataFrame 'df' with your dataset

def prediction(request):

if request.method == 'POST':

# Get user input values from the form

stay\_up\_late = float(request.POST['stay\_up\_late'])

pressure\_level = float(request.POST['pressure\_level'])

coffee\_consumed = float(request.POST['coffee\_consumed'])

brain\_working\_duration = float(request.POST['brain\_working\_duration'])

school\_assesssment = float(request.POST['school\_assesssment'])

swimming = float(request.POST['swimming'])

hair\_washing = float(request.POST['hair\_washing'])

hair\_grease = float(request.POST['hair\_grease'])

dandruff = float(request.POST['dandruff'])

libido = float(request.POST['libido'])

# Create a DataFrame with the user's input data

input\_data = pd.DataFrame({

'stay\_up\_late': [stay\_up\_late],

'pressure\_level': [pressure\_level],

'coffee\_consumed': [coffee\_consumed],

'brain\_working\_duration': [brain\_working\_duration],

'school\_assesssment': [school\_assesssment],

'swimming': [swimming],

'hair\_washing': [hair\_washing],

'hair\_grease': [hair\_grease],

'dandruff': [dandruff],

'libido': [libido]

})

# Call the knn\_classification function to get the trained K-NN classifier

knn\_classifier = knn\_classification(request)

# Predict hair loss for the user's input

predicted\_hair\_loss = knn\_classifier.predict(input\_data)

# Rest of your code remains the same

print('--------------------')

print(predicted\_hair\_loss)

print(type(predicted\_hair\_loss))

print(len(predicted\_hair\_loss))

# Determine the prevention tips based on the predicted hair loss level

prevention\_tips = ""

level = ""

if predicted\_hair\_loss[0] == "Few":

level = "Level 1"

prevention\_tips = [

"Maintain a healthy diet with essential nutrients for hair growth.",

"Avoid excessive stress and practice relaxation techniques.",

"Use a mild and suitable shampoo for your hair type.",

"Avoid excessive use of heat styling tools."

]

elif predicted\_hair\_loss[0] == "Medium":

level = "Level 2"

prevention\_tips = [

"Increase your intake of vitamins and minerals beneficial for hair health.",

"Consider using hair care products designed for your hair type and needs.",

]

elif predicted\_hair\_loss[0] == "Many":

level = "level 3"

prevention\_tips = [

"Consult a dermatologist or hair specialist for a thorough assessment.",

"Explore advanced hair care treatments and therapies.",

]

elif predicted\_hair\_loss[0] == "A lot":

level = "level 4"

prevention\_tips = [

"Seek immediate professional help from a hair loss expert.",

"Consider hair restoration treatments if appropriate.",

]

else:

return render(request,'users/result.html',{'msg':"No Hair Loss"})

context = {'level':level,"tips":prevention\_tips}

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

else:

return render(request, 'users/prediction\_form.html')

# Define a view for predicting hair loss

# def prediction(request):

# if request.method == 'POST':

# # Get user input values from the form

# stay\_up\_late = float(request.POST['stay\_up\_late'])

# pressure\_level = float(request.POST['pressure\_level'])

# coffee\_consumed = float(request.POST['coffee\_consumed'])

# brain\_working\_duration = float(request.POST['brain\_working\_duration'])

# school\_assesssment = float(request.POST['school\_assesssment'])

# swimming = float(request.POST['swimming'])

# hair\_washing = float(request.POST['hair\_washing'])

# hair\_grease = float(request.POST['hair\_grease'])

# dandruff = float(request.POST['dandruff'])

# libido = float(request.POST['libido'])

# # Create a DataFrame with the user's input data

# input\_data = pd.DataFrame({

# 'stay\_up\_late': [stay\_up\_late],

# 'pressure\_level': [pressure\_level],

# 'coffee\_consumed': [coffee\_consumed],

# 'brain\_working\_duration': [brain\_working\_duration],

# 'school\_assesssment': [school\_assesssment],

# 'swimming': [swimming],

# 'hair\_washing': [hair\_washing],

# 'hair\_grease': [hair\_grease],

# 'dandruff': [dandruff],

# 'libido': [libido]

# })

# # Feature scaling for the user's input data

# scaler = StandardScaler()

# input\_data\_scaled = scaler.fit\_transform(input\_data)

# # Call the knn\_classification function to get the trained K-NN classifier

# knn\_classifier = knn\_classification(request)

# # Predict hair loss for the user's input

# predicted\_hair\_loss = knn\_classifier.predict(input\_data\_scaled)

# return render(request, 'users/result.html', {

# 'predicted\_hair\_loss': predicted\_hair\_loss[0]

# })

# else:

# return render(request, 'users/prediction\_form.html')

**Base.html:**

<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

<meta charset="utf-8">

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

<title>Hair Fall Detection and Prevention</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|Roboto: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/animate.css/animate.min.css' %}" rel="stylesheet">

<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: Company

\* Updated: Sep 18 2023 with Bootstrap v5.3.2

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

\* Author: BootstrapMade.com

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

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

</head>

<body>

<!-- ======= Header ======= -->

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

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

<h1>An Enhanced Stress based Hairfall

Detection and Prevention using KNN and

Machine Learning Techniques

</h1>

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

<!-- <a href="index.html" class="logo me-auto me-lg-0"><img src="assets/img/logo.png" alt="" class="img-fluid"></a>-->

<nav id="navbar" class="navbar order-last order-lg-0">

<ul>

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

<li><a href="{% url 'UserLogin' %}">user</a></li>

<li><a href="{% url 'AdminLogin' %}">admin</a></li>

<li><a href="{% url 'UserRegister' %}">Register here</a></li>

</ul>

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

</nav><!-- .navbar -->

</div>

</header><!-- End Header -->

<br>

<br><br>

<br><br>

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

</div> </div>

<main id="main">

{% block content %}

{% endblock %}

<br><br>

<br><br><br><br>

<br><br>

<br><br>

</main><!-- End #main -->

<!-- ======= Footer ======= -->

<footer id="footer">

<div class="footer-top">

<div class="container">

<div class="row">

</div>

</div>

</div>

</div>

<div class="container">

<div class="copyright">

&copy; Copyright <strong><span>Medicio</span></strong>. All Rights Reserved

</div>

<div class="credits">

<!-- All the links in the footer should remain intact. -->

<!-- You can delete the links only if you purchased the pro version. -->

<!-- Licensing information: https://bootstrapmade.com/license/ -->

<!-- Purchase the pro version with working PHP/AJAX contact form: https://bootstrapmade.com/medicio-free-bootstrap-theme/ -->

Designed by <a href="https://bootstrapmade.com/">SHAISTA</a>

</div>

</div>

</footer><!-- End Footer -->

<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>

**Admin side views:**

from django.shortcuts import render

from django.contrib import messages

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})isterUsers.html',{'data':data})