#Importing all the required libraries

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

import pickle

#read data from the csv file

data = pd.read\_csv("home/nnnnn.csv")

#target variable

predict = "Technostress"

#Model

x = data.drop(columns="Technostress")

y = data["Technostress"]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2)

best\_model = 0

#training model

for i in range(10000):

    x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=i)

    linear = LinearRegression()

    linear.fit(x\_train, y\_train)

    accuracy = linear.score(x\_test, y\_test)

    print(accuracy)

    if accuracy > best\_model:

        best = accuracy

        #Save the model

        with open("home/LnearRegressionModel.pickle", "wb") as f:

            pickle.dump(linear, f)

#Open and load the trained model

pickle\_in = open("home/LnearRegressionModel.pickle", "rb")

linear = pickle.load(pickle\_in)

#make predictions

predictions = linear.predict(pd.DataFrame([[0,1,1,1]], columns=["Gender","Age","hours\_spent","TECHNOLOGY"]))

print(predictions)

from django.shortcuts import render

import pickle

from django.views import View

# Create your views here.

class HomeView(View):

    def get(self, request, \*args, \*\*kwargs):

        return render(request, "home/index.html")

    def post(self, request, \*args, \*\*kwargs):

        #Get user data from form

        gender = int(request.POST.get("gender"))

        age = int(request.POST.get("age"))

        hours\_spent = int(request.POST.get("hours\_spent"))

        tech\_used = int(request.POST.get("tech\_used"))

        #Open and load the trained model

        pickle\_in = open("home/LnearRegressionModel.pickle", "rb")

        linear = pickle.load(pickle\_in)

        #make predictions

        prediction = linear.predict(pd.DataFrame([[gender,age,hours\_spent,tech\_used]], columns=["Gender","Age","hours\_spent","TECHNOLOGY"]))

        prediction\_point = prediction[0]

        #Making classification

        if prediction\_point >= 1.38:

            classification = "You are not Stressed"

        else:

            classification = "You are Stressed"

        #Passing the necessary information to the html file

        context = {

            "prediction\_point":prediction\_point,

            "classification":classification

        }

        return render(request, "home/result.html", context)