from flask import Flask

from flask import request, jsonify

import os

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

import joblib # Used for avoid computing with the same thing twice

 # Importing the neccessary libraries

app = Flask(\_\_name\_\_) # Creating a server

@app.route('/',methods = ["POST"]) # Creating the flask or server to take the POST request as input for the model

def predict\_house\_price(): # Making a function

    if(os.path.exists("linear\_regression.joblib")): # Checking if the data is already exist then do not build the model else build the model

        data\_frame = pd.read\_csv('housing.csv') # Reading the data from CSV file

        data\_frame = data\_frame.drop(["longitude", "latitude","households"],axis=1) # Dropping the unwated columns

        data\_frame = data\_frame.dropna() # Dropping the values having input as nan

        data\_frame = pd.concat([data\_frame, pd.get\_dummies(data\_frame['ocean\_proximity'])],axis = 1) # Making dummy columns for ocean proximity

        data\_frame = data\_frame.drop('ocean\_proximity',axis = 1) # Dropping the column named as ocean proximity

        model = LinearRegression() # Linear regression class instence

        X = data\_frame[['housing\_median\_age', 'total\_rooms', 'total\_bedrooms', 'population',

            'median\_income', '<1H OCEAN', 'INLAND', 'ISLAND',

            'NEAR BAY', 'NEAR OCEAN', '<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY',

            'NEAR OCEAN']]  # Making the data frame with new dummy columns. Here X is the dependent variable

        Y = data\_frame[['median\_house\_value']] # Y is the independent variable or the variable that is going to be predicted

        model.fit(X,Y) # Training

        joblib.dump(model,"linear\_regression.joblib") # Using joblib.dump() to save the data (Serialization)

    input = request.form # Taking the request from the server

    model = joblib.load("linear\_regression.joblib") # Restoring the data (De-serialization) using joblib.load

    xtest = pd.DataFrame(input,index = [1])

    if input['ocean\_proximity'] != 'ISLAND':   # Converting the alphabets into the number

        xtest['ISLAND'] = [0]

    if input['ocean\_proximity'] != 'NEAR OCEAN':

        xtest['NEAR OCEAN'] = [0]

    if input['ocean\_proximity'] != 'INLAND':

        xtest['INLAND'] = [0]

    if input['ocean\_proximity'] != 'NEAR BAY':

        xtest['NEAR BAY'] = [0]

    if input['ocean\_proximity'] != '<1H OCEAN':

        xtest['<1H OCEAN'] = [0]

    xtest = pd.concat([xtest, pd.get\_dummies(xtest['ocean\_proximity'])],axis = 1) # Concatinating the dummy columns (one hot encoding)

    xtest = xtest.drop('ocean\_proximity',axis = 1)

    prediction = model.predict(xtest[['housing\_median\_age', 'total\_rooms', 'total\_bedrooms', 'population',

        'median\_income', '<1H OCEAN', 'INLAND', 'ISLAND',

        'NEAR BAY', 'NEAR OCEAN', '<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY',

        'NEAR OCEAN']]) # Predicting

    prediction = pd.DataFrame(prediction,columns = ['House\_price'] )

    #return jsonify({'House\_Price': prediction.iloc[0,0]})

    predicted = str(prediction.iloc[0,0]) # Storing the predicted value in variable called as predicted

    return  predicted # returning the predicted value

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

    app.run(host = "0.0.0.0") # Running the server on local host network