**import os, sys, shutil, time**

**from flask import Flask, request, jsonify, render\_template,send\_from\_directory**

**import pandas as pd**

**import joblib**

**from sklearn.ensemble import RandomForestClassifier**

**import numpy as np**

**import urllib.request**

**import json**

**from geopy.geocoders import Nominatim**

**import geopy**

**#geopy.geocoders.options.default\_user\_agent = "my-application"**

**app = Flask(\_\_name\_\_)**

**@app.route('/')**

**def root():**

**return render\_template('index.html')**

**@app.route('/index.html')**

**def index():**

**return render\_template('index.html')**

**@app.route('/work')**

**def work():**

**return render\_template('work.html')**

**@app.route('/about')**

**def about():**

**return render\_template('about.html')**

**@app.route('/result.html', methods = ['POST'])**

**def predict():**

**rfc = joblib.load('model.sav')**

**print('model loaded')**

**if request.method == 'POST':**

**address = request.form['Location']**

**geolocator = Nominatim(user\_agent="app")**

**location = geolocator.geocode(address,timeout=None)**

**print(location.address)**

**lat=[location.latitude]**

**log=[location.longitude]**

**latlong=pd.DataFrame({'latitude':lat,'longitude':log})**

**print(latlong)**

**DT= request.form['timestamp']**

**latlong['timestamp']=DT**

**data=latlong**

**cols = data.columns.tolist()**

**cols = cols[-1:] + cols[:-1]**

**data = data[cols]**

**data['timestamp'] = pd.to\_datetime(data['timestamp'].astype(str), errors='coerce')**

**data['timestamp'] = pd.to\_datetime(data['timestamp'], format = '%d/%m/%Y %H:%M:%S')**

**column\_1 = data.iloc[:,0]**

**DT=pd.DataFrame({"year": column\_1.dt.year,**

**"month": column\_1.dt.month,**

**"day": column\_1.dt.day,**

**"hour": column\_1.dt.hour,**

**"dayofyear": column\_1.dt.dayofyear,**

**"week": column\_1.dt.week,**

**"weekofyear": column\_1.dt.weekofyear,**

**"dayofweek": column\_1.dt.dayofweek,**

**"weekday": column\_1.dt.weekday,**

**"quarter": column\_1.dt.quarter,**

**})**

**data=data.drop('timestamp',axis=1)**

**final=pd.concat([DT,data],axis=1)**

**X=final.iloc[:,[1,2,3,4,6,10,11]].values**

**my\_prediction = rfc.predict(X)**

**if my\_prediction[0][0] == 1:**

**my\_prediction='Predicted crime : Act 379-Robbery'**

**elif my\_prediction[0][1] == 1:**

**my\_prediction='Predicted crime : Act 13-Gambling'**

**elif my\_prediction[0][2] == 1:**

**my\_prediction='Predicted crime : Act 279-Accident'**

**elif my\_prediction[0][3] == 1:**

**my\_prediction='Predicted crime : Act 323-Violence'**

**elif my\_prediction[0][4] == 1:**

**my\_prediction='Predicted crime : Act 302-Murder'**

**elif my\_prediction[0][5] == 1:**

**my\_prediction='Predicted crime : Act 363-kidnapping'**

**else:**

**my\_prediction='Place is safe no crime expected at that timestamp.'**

**return render\_template('result.html', prediction = my\_prediction)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**app.run(debug = True)**