***EARTHQUAKE PREDICTION MODEL USING PYTHON***

***PHASE - 4 DEVELOPMENT PART - 2***

***Visualizing Earthquake Data on a World Map***

1. *To visualize earthquake data on a world map, you can use libraries like Basemap or Folium.*

*ii. In this example,use Folium, which is easy to use and provides interactive maps.*

*iii. Make sure you have the necessary libraries installed before proceeding.*

1. ***Import the required libraries:***

*import folium*

*import pandas as pd*

1. ***Load your earthquake data:***

*Load your earthquake data into a Pandas DataFrame.*

1. ***Create a base map:***

*m = folium.Map(location=[0, 0], zoom\_start=2)*

*# Set the initial map location and zoom level*

1. ***Iterate through your earthquake data and add markers to the map:***

*for index, row in earthquake\_data.iterrows():*

*folium.Marker([row['Latitude'], row['Longitude']],*

*popup=f"Magnitude: {row['Magnitude']}").add\_to(m)*

1. ***Display the map:***

*m.save("earthquake\_map.html")*

*# Save the map to an HTML file*

*This code will create an interactive map with markers representing earthquake data.*

***SPLITTING EARTHQUAKE DATA INTO TRAINING AND TESTING SETS***

1. *To build earthquake prediction model, need to split our data into training and testing sets for model evaluation.*
2. *we can use libraries like scikit-learn for this purpose.*
3. ***Import the required libraries:***

*from sklearn.model\_selection*

*import train\_test\_split*

1. ***Split your data:***

*X = earthquake\_data.drop(columns=['Magnitude'])*

*# Features*

*y = earthquake\_data['Magnitude']*

*# Target variable*

*# Split the data into training and testing sets (adjust the test\_size and random\_state as needed)*

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

***Import the necessary libraries required for building the model and data analysis of the earthquakes.***

***In:***

*import numpy as np*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import os*

*print(os.listdir("../input"))*

*data = pd.read\_csv("../input/database.csv")*

*data.head()*

***OUT:***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***DATE*** | ***TIME*** | ***LATTITUDE*** | ***LONGITUDE*** | ***TYPE*** | ***DEPTH*** | ***MAGNITUDE*** |
| ***0*** | *01/02/1965* | *13:44:18* | *19.246* | *145.616* | *EARTHQUAKE* | *131.6* | *6.0* |
| ***1*** | *01/04/1965* | *11:29:49* | *1.863* | *127.352* | *EARTHQUAKE* | *80.0* | *5.8* |
| ***2*** | *01/05/1965* | *18:05:58* | *-20.579* | *-173.972* | *EARTHQUAKE* | *20.0* | *6.2* |
| ***3*** | *01/08/1965* | *18:49:43* | *-59.076* | *-23.557* | *EARTHQUAKE* | *15.0* | *5.8* |
| ***4*** | *01/09/1965* | *13:32:50* | *11.938* | *126.427* | *EARTHQUAKE* | *15.0* | *5.8* |

***VISUALIZATION:***

1. *Here, the data is random we need to scale according to inputs to the model.*
2. *In this, we convert given Date and Time to Unix time which is in seconds and a numeral.*
3. *This can be easily used as input for the network we built.*

***IN:***

*from mpl\_toolkits.basemap import Basemap*

*m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat\_ts=20,resolution='c')*

*longitudes = data["Longitude"].tolist()*

*latitudes = data["Latitude"].tolist()*

*#m=Basemap(width=12000000,height=9000000,projection='lcc',*

*#resolution=None,lat\_1=80.,lat\_2=55,lat\_0=80,lon\_0=-107.)*

*x,y = m(longitudes,latitudes)*

*fig = plt.figure(figsize=(12,10))*

*plt.title("All affected areas")*

*m.plot(x, y, "o", markersize = 2, color = 'blue')*

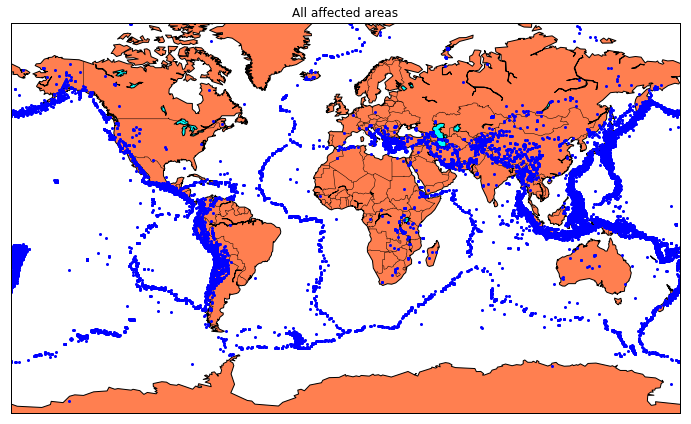
*m.drawcoastlines()*

*m.fillcontinents(color='coral',lake\_color='aqua')*

*m.drawmapboundary()*

*m.drawcountries()*

*plt.show()*



***SPLITTING THE DATA:***

1. *Firstly, split the data into Xs and ys which are input to the model and output of the model respectively.*
2. *Here, inputs are TImestamp, Latitude and Longitude and outputs are Magnitude and Depth.*
3. *Split the Xs and ys into train and test with validation.*
4. *Training dataset contains 80% and Test dataset contains 20%.*

***IN:***

*X = final\_data[['Timestamp', 'Latitude', 'Longitude']]*

*y = final\_data[['Magnitude', 'Depth']]*

*from sklearn.cross\_validation import train\_test\_split*

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

*print(X\_train.shape, X\_test.shape, y\_train.shape, X\_test.shape)*

*(18727, 3) (4682, 3) (18727, 2) (4682, 3)*

*from sklearn.ensemble import RandomForestRegressor*

*reg = RandomForestRegressor(random\_state=42)*

*reg.fit(X\_train, y\_train)*

*reg.predict(X\_test)*

***OUT:***

*array([[ 5.96, 50.97],*

*[ 5.88, 37.8 ],*

*[ 5.97, 37.6 ],*

*...,*

*[ 6.42, 19.9 ],*

*[ 5.73, 591.55],*

*[ 5.68, 33.61]])*

*EXAMPLE:*

*import pandas as pd*

*import random*

*# Generate synthetic earthquake data*

*n\_samples = 100 # Number of earthquake data points*

*data = {*

*'Latitude': [random.uniform(-90, 90) for \_ in range(n\_samples)],*

*'Longitude': [random.uniform(-180, 180) for \_ in range(n\_samples)],*

*'Magnitude': [round(random.uniform(4.0, 9.0), 1) for \_ in range(n\_samples)]*

*}*

*# Create a Pandas DataFrame*

*earthquake\_data = pd.DataFrame(data)*

*# Display the first few rows of the dataset*

*print(earthquake\_data.head())*

*# You can save this dataset to a CSV file for future use*

*earthquake\_data.to\_csv('earthquake\_sample\_data.csv', index=False)*

1. ***Import Libraries:***

*First, we import the necessary Python libraries:*

*import pandas as pd*

*import random*

1. ***Define the Number of Data Points:***

*We specify the number of data points (earthquake events) that we want to generate. In this example, n\_samples is set to 100, meaning we'll generate data for 100 earthquake events.*

1. ***Generate Synthetic Earthquake Data:***

*We create synthetic earthquake data using random values. Here's how each column is generated:*

* *'Latitude': We use random.uniform(-90, 90) to generate random latitude values between -90 and 90 degrees. This range covers the entire globe.*
* *'Longitude': We use random.uniform(-180, 180) to generate random longitude values between -180 and 180 degrees, covering the entire globe.*
* *'Magnitude': We use random.uniform(4.0, 9.0) to generate random magnitude values between 4.0 and 9.0. This range represents a typical range of earthquake magnitudes.The generated data is stored in the data dictionary.*

1. ***Create a DataFrame:***

*We use the Pandas library to create a DataFrame, earthquake\_data, from the data dictionary. This DataFrame will hold our earthquake data.*

*earthquake\_data = pd.DataFrame(data)*

1. ***Display the Dataset:***

*We print the first few rows of the dataset to see what it looks like.*

*print(earthquake\_data.head())*

1. ***Save to a CSV File :***

*If you want to save the sample dataset to a CSV file for later use, you can use the following line of code:*

*earthquake\_data.to\_csv('earthquake\_sample\_data.csv', index=False)*

*This line saves the dataset to a file named 'earthquake\_sample\_data.csv' in the current directory. The index=False argument tells Pandas not to save the DataFrame index as a separate column in the CSV file.*

***Splitting it into training and testing sets :***

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

*# Define your feature variables (X) and target variable (y)*

*X = earthquake\_data[['Latitude', 'Longitude']] # Features*

*y = earthquake\_data['Magnitude'] # Target variable*

*# Split the data into training and testing sets*

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

1. *Import train\_test\_split Function:*

*You first import the train\_test\_split function from the sklearn.model\_selection module. This function is part of the scikit-learn library and is used to split your dataset.*

1. *Define Feature and Target Variables:*

*You define your feature variables (X) and target variable (y):*

*X contains the feature data, which in this case includes 'Latitude' and 'Longitude' columns from your earthquake dataset. Depending on your model, you might include more features.*

*y contains the target variable, which is 'Magnitude' in this example.*

1. *Splitting the Data:*

*The train\_test\_split function is used to split your feature and target variables into training and testing sets.*

*X\_train and y\_train will contain the feature and target data for training your machine learning model.*

*X\_test and y\_test will contain the feature and target data for testing your model's performance.*

*test\_size=0.2 specifies that 20% of the data will be used for testing, and the remaining 80% will be used for training. You can adjust this percentage to suit your needs.*

*random\_state is set to 42, which is a random seed. Setting a random seed ensures that the split is reproducible; you can use any integer value for random\_state.*