Earthquake Prediction Model with Machine Learning

Phase 4

312621243025----Srikanth.A

Earthquake Prediction Model

It is well known that if a disaster occurs in one region, it is likely to happen again. Some regions have frequent earthquakes, but this is only a comparative amount compared to other regions.

So, predicting the earthquake with date and time, latitude and longitude from previous data is not a trend that follows like other things, it happens naturally.

I will start this task to create a model for earthquake prediction by importing the necessary python libraries:

Visualizing Earthquake Data on a World Map:

To visualize earthquake data on a world map, you can use various libraries and tools. Here, we'll use Python and the **folium** library as an example. Ensure you have your earthquake data in a suitable format (e.g., a CSV Here's a basic example of how to visualize earthquake data on a world map using Folium

Programming code:

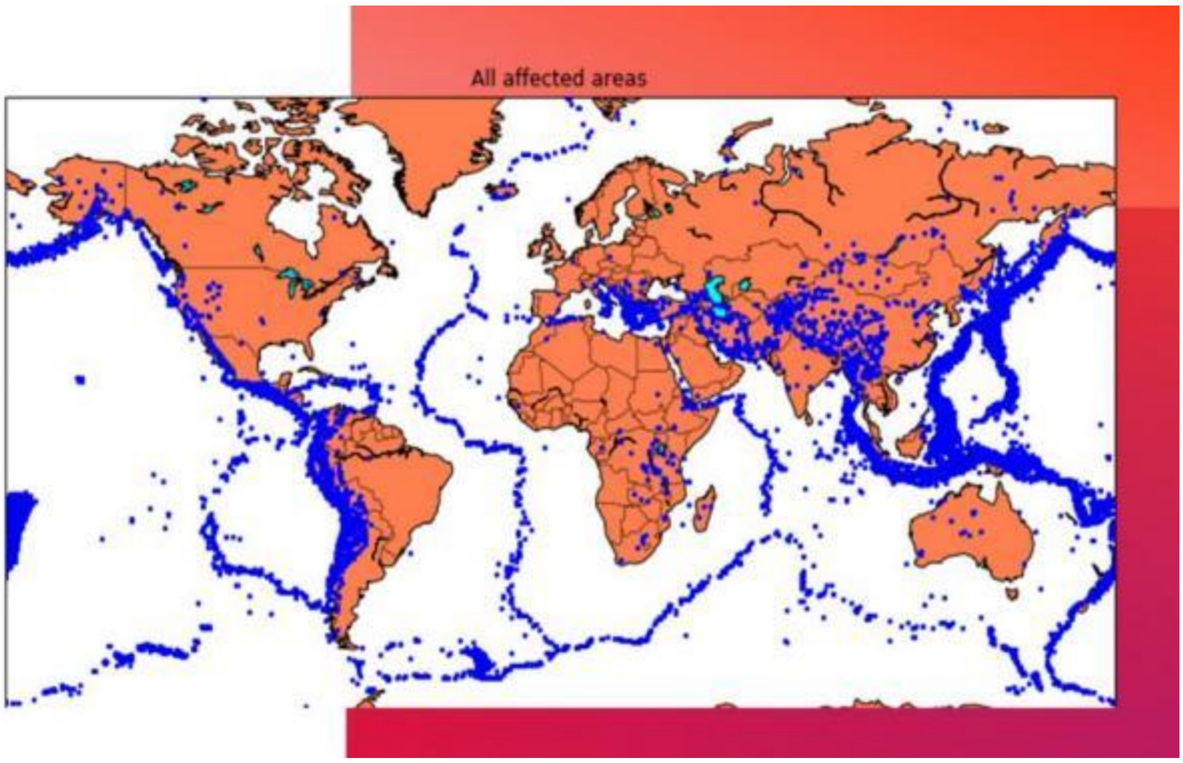
```
import folium

import pandas as pd

# Load your earthquake data into a DataFrame (assuming you
have latitude, longitude, and magnitude columns)

data = pd.read_csv("earthquake_data.csv")


# Create a base map

m = folium.Map(location=[0, 0], zoom_start=2)


# Add markers for each earthquake event

for index, row in data.iterrows():

   folium.CircleMarker(

       location=[row['latitude'], row['longitude']],

       radius=row['magnitude'],  # You can adjust the marker
size based on magnitude

       color='red',

       fill=True,

       fill_color='red',

       fill_opacity=0.4

   ).add_to(m)
```

Save the map as an HTML file or display it in aJupyter Notebook

m.save("earthquake_map.html")

Data Visualization

Now, before we create the earthquake prediction model, let’s visualize the data on a world map that shows a clear representation of where the earthquake frequency will be more:



Best: 0.957655 using {'activation': 'relu','batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'} 0.333316 (0.471398) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10,'loss': 'squared_hinge', 'neurons': 16,'optimizer': 'SGD'} 0.000000 (0.000000) with: {'activation': 'sigmoid','batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge','neurons': 16,'optimizer': 'Adadelta'} 0.957655 (0.029957) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16,

'optimizer': 'SGD'} 0.645111 (0.456960) with: {'activation': 'relu','batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16,'optimizer': 'Adadelta'}

Splitting Data into Training and Testing Sets:

To build a machine learning model, you need to split your data into training and testing sets. This allows you to train the model on one subset and evaluate its performance on another. Python's scikit-

library is commonly used for this task. Here's how you can split your data:

Program :

```
from sklearn.model_selection import train_test_split



# Assuming you have a DataFrame'data'with features (X) and a target (y)
X = data[['feature1','feature2', ...]]  # Replace with your features

y = data['target_column']  # Replace with your target variable



# Split the data into training and testing sets (e.g., 80% for training, 20% for testing)
X_train, X_test, y_train,y_test = train_test_split(X,y, test_size=0.2, random_state=42)


# Now you have X_train, X_test, y_train, andy_test for model training and evaluation
```

Output:

Table form:

After completing these steps, you'll have earthquake data visualized on a world map, and your data will be split into training and testing sets, which you can use training your earthquake prediction model. The actual modeling and will depend on the nature of your dataset and the specific goals of

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-1.57631e+08
1	1.863	127.352	80.0	5.8	-1.57466e+08
2	-20.579	-173.972	20.0	6.2	-1.57356e+08
3	-59.076	-23.557	15.0	5.8	-1.57094e+08
4	11.938	126.427	15.0	5.8	-1.57026e+08