Earthquake prediction model-Visualize the data in world map and Splitting the data

Introduction:

As we discussed about the earthquake prediction, a model which predicts the earthquake before it happened. Most earthquakes, or 90%, are natural and result from tectonic activity. 10% of the remaining characteristics are associated with volcanism, man-made consequences, or other variables. So let us develop a model to predict the earthquake as much as accuracy we can. In this phase we Visualizing the data on a world map and split the data into training and testing sets.

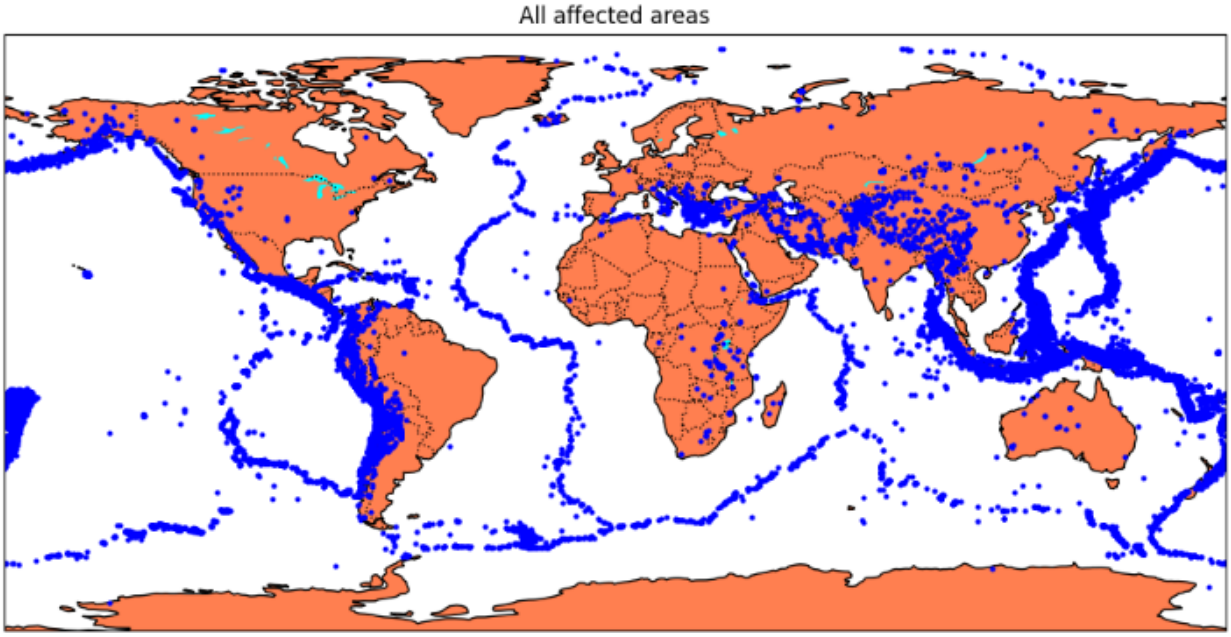
Visualizing the data on a world map:

We visualize the data in the world map using cartopy. The code snippet leverages Python libraries such as Matplotlib, Cartopy, and Cartopy's feature module to create a geospatial visualization. It begins by establishing a Cartopy Plate Carrie projection, a straightforward projection ideal for displaying data on a map. The figure and axis are defined within the context of this projection, with the figure set to a size of 12x10 inches. The plot title, "All affected areas," is specified. The code assumes the presence of a DataFrame ('df') with 'Longitude' and 'Latitude' columns, which are then converted into lists ('longitudes' and 'latitudes'). These data points are plotted as blue circles ('bo') on the map, with a marker size of 2, and they are transformed according to the established Cartopy projection.

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature

# Create a Cartopy projection
projection = ccrs.PlateCarree()

# Define the figure and axis
fig, ax = plt.subplots(subplot_kw={'projection': projection}, figsize=(12, 10))
ax.set_title("All affected areas")

# Replace 'df' with your actual DataFrame
longitudes = df["Longitude"].tolist()
latitudes = df["Latitude"].tolist()

# Plot the data on the map
ax.plot(longitudes, latitudes, 'bo', markersize=2, transform=projection)

# Add map features
ax.coastlines()
ax.add_feature(cfeature.LAND, facecolor='coral')
ax.add_feature(cfeature.LAKES, facecolor='aqua')
ax.add_feature(cfeature.BORDERS, linestyle=':')

# Display country borders

# Show the map
plt.show()
```

In addition to plotting data points, the code adds context to the map by incorporating various map features. Coastlines are included to delineate land and water areas. Land areas are filled with a coral color, while lakes are shaded in aqua. Borders are added with a dotted lifestyle to

provide visual separation between regions. Furthermore, the code adds country borders using the Natural Earth dataset, specifying a solid line style and a black edge color. The visualization, with the data points and these added features, is displayed using 'plt.show(),' resulting in a comprehensive map representation. To successfully execute this code, users should replace 'df' with their actual DataFrame containing the relevant longitude and latitude data.



We can see that the China, India borders of Africa USA and Mexico are affected.Also many earthquakes are happened in the ocean. Antarctica doesn't seem to be affected in earthquake. This clearly creates the doubt of maybe there no earthquake happened in Antarctica or nobody recorded in Antarctica about the earthquake.

Splitting the data:

Splitting the dataset is fundamental process. We split data into training and testing data. The training data will be fed into the model to find the hidden patterns in the data. Then we check the accuracy of the model by testing data. We have x and y in training and test data. X_train and Y_train data are fed into the model.Then the model find the hiden pattern and train itself. Then we test the data by giving the x_test data and predict the Y-test data. Then we check the accuracy by many methods like confusion matrix, F1 score, Precision and recall method etc,.

```
from sklearn.model_selection import train_test_split

X = data[['Timestamp', 'Latitude', 'Longitude']]
y = data[['Magnitude', 'Depth']]

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)
```

shape property

```
(18729, 3) (4683, 3) (18729, 2) (4683, 3)
```

Here we splitted the data into X_train, X_test, y_train, y_test.

```
from sklearn.model_selection import train_test_split

X = data[['Timestamp', 'Latitude', 'Longitude']]
y = data[['Magnitude', 'Depth']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("X_train")
print(X_train)
print("X_test")
print(X_test)
print("Y_train")
print(y_train)
print("X_test")
print(X_test)
```

```
X_train
          Timestamp  Latitude  Longitude
18765  1196362820.0    14.944    -61.274
21035  1319828074.0   -14.438    -75.966
18334  1174831078.0    38.340     20.420
16776  1083850993.0    42.525    145.021
9152    588702617.0   -15.864   -172.067
...             ...       ...        ...
11964   771284390.0    27.995    140.700
21575  1359604676.0   -10.682    166.381
5390    314944810.0    -6.847    129.634
860     -58874691.0    -5.469    153.269
15795  1019139444.0   -60.657    -25.843

[18729 rows x 3 columns]
X_test
          Timestamp  Latitude  Longitude
3848    193823279.0     3.166     99.015
14008   896888828.0    43.679    -29.020
16258  1052708585.0     1.142     98.911
18090  1161872917.0    38.649     15.390
15192   976898688.0    38.457     31.351
...             ...       ...        ...
15058   972105915.0   -17.286   -175.176
18377  1175638662.0    -6.742    154.889
87     -151488413.0    36.405     70.724
10309   662681490.0   -21.953    174.818
14530   938628092.0   -30.738    -71.993
```

Now we have select the perfect the model to predict the earthquake .We selected ANN model Artificial Neural Networks to train the data. Then we will check the accuracy score and if it is bad then we need to change the model. Let's check that in next phase.

Conclusion:

Thus we visualized the data in cartopy and splitted the data into X_train, X_Test, Y_train, Y_test. In next step we will try different model and we will check the accuracy with different models and check the accuracy score which will say which is better model.