Name:Padma priya.E

AI-Phase-4

October 31,2023

[ ]:
```
Date:31 october 2023
Team ID:NM2023TMID345
Team Name:Proj_227274_team_1
Project Name:Earthquake prediction model
```

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
from sklearn.model_selection import train_test_split
```

[ ]:
```
# Load your earthquake data into a Pandas DataFrame
```

[6]:
```
data = pd.read_csv(r'C:\Users\91912\Desktop\AI_Phase3/database.csv')
```

[7]:
```
data.head()
```

[7]:

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	…	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance
0	01-02-1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	…	NaN	NaN	NaN
1	01-04-1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	…	NaN	NaN	NaN
2	01-05-1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	…	NaN	NaN	NaN
3	01-08-1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	…	NaN	NaN	NaN
4	01-09-1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	…	NaN	NaN	NaN

```
  Horizontal Error  Root Mean Square           ID Source Location Source  \
0              NaN               NaN  ISCGEM860706  ISCGEM          ISCGEM
1              NaN               NaN  ISCGEM860737  ISCGEM          ISCGEM
2              NaN               NaN  ISCGEM860762  ISCGEM          ISCGEM
3              NaN               NaN  ISCGEM860856  ISCGEM          ISCGEM
4              NaN               NaN  ISCGEM860890  ISCGEM          ISCGEM

  Magnitude Source    Status
0          ISCGEM  Automatic
1          ISCGEM  Automatic
2          ISCGEM  Automatic
3          ISCGEM  Automatic
4          ISCGEM  Automatic

[5 rows x 21 columns]
```

[8]:
```
 import datetime
import time

timestamp = []

for d, t in zip(data['Date'], data['Time']):
    try:
        ts = datetime.datetime.strptime(d + ' ' + t, '%m/%d/%Y %H:%M:%S')
        min_timestamp = datetime.datetime(1970, 1, 1)
        max_timestamp = datetime.datetime(2038, 1, 19)

        if min_timestamp <= ts <= max_timestamp:
            timestamp.append(time.mktime(ts.timetuple()))
        else:
            timestamp.append('OutofRange')
    except ValueError:
        # print('ValueError')
        timestamp.append('ValueError')

data['Timestamp'] = timestamp

timeStamp = pd.Series(timestamp)
data['Timestamp'] = timeStamp.values

final_data = data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

[8]:
```
   Latitude Longitude      Type  Depth  Depth Error \
7   -13.309   166.212 Earthquake 35.0         NaN
```

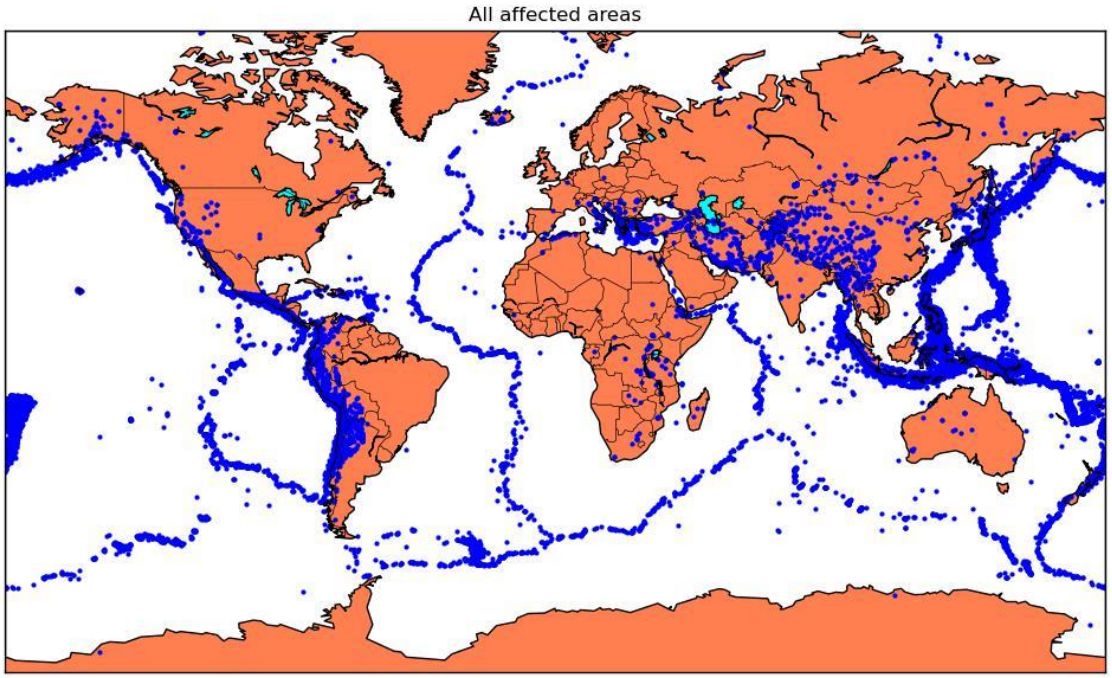
```
8    -56.452   -27.043 Earthquake   95.0         NaN
9    -24.563   178.487 Earthquake  565.0         NaN
10    -6.807   108.988 Earthquake  227.9         NaN
11    -2.608   125.952 Earthquake   20.0         NaN

    Depth Seismic Stations  Magnitude Magnitude Type Magnitude Error  \
7                     NaN       6.0            MW             NaN
8                     NaN       6.0            MW             NaN
9                     NaN       5.8            MW             NaN
10                    NaN       5.9            MW             NaN
11                    NaN       8.2            MW             NaN

    Magnitude Seismic Stations Azimuthal Gap Horizontal Distance  \
7                         NaN           NaN                 NaN
8                         NaN           NaN                 NaN
9                         NaN           NaN                 NaN
10                        NaN           NaN                 NaN
11                        NaN           NaN                 NaN

    Horizontal Error  Root Mean Square              ID   Source  \
7                NaN              NaN   ISCGEM861111    ISCGEM
8                NaN              NaN  ISCGEMSUP861125  ISCGEMSUP
9                NaN              NaN   ISCGEM861148    ISCGEM
10               NaN              NaN   ISCGEM861155    ISCGEM
11               NaN              NaN   ISCGEM861299    ISCGEM

    Location Source Magnitude Source    Status Timestamp
7          ISCGEM          ISCGEM  Automatic OutofRange
8          ISCGEM          ISCGEM  Automatic OutofRange
9          ISCGEM          ISCGEM  Automatic OutofRange
10         ISCGEM          ISCGEM  Automatic OutofRange
11         ISCGEM          ISCGEM  Automatic OutofRange
```

[ ]:
```
# Visualizing the data on a world map
```

[9]:
```
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()
x, y = m(longitudes, latitudes)
```

[10]:
```
 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 it into training and testing sets
```

[5]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

[ ]:
```
# Load your earthquake data into a Pandas DataFrame
```

[20]:
```
data = pd.read_csv(r'C:\Users\91912\Desktop\AI_Phase3/database.csv')
```

[ ]:
```
# Define your feature columns and target columns
```

[17]:
```
X = data[['Latitude', 'Longitude']]
y = data[['Magnitude', 'Depth']]
```

[ ]:
```
# Split the data into training and testing sets
```

[18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, ␣ ↪random_state=42)
```

[ ]:
```
# Print the shapes of the training and testing sets to verify the split
```

[19]:
```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (18729, 2)
X_test shape: (4683, 2)
y_train shape: (18729, 2)
y_test shape: (4683, 2)
```

[ ]:
```
# using a Decision Tree regressor for your earthquake prediction model.
```

[21]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

[ ]:
```
# Create a Decision Tree regressor
```

[23]:
```
tree_model = DecisionTreeRegressor(random_state=42)
```

[ ]:
```
# Fit the Decision Tree model to the training data
```

[24]:
```
tree_model.fit(X_train, y_train)
```

[24]:
```
DecisionTreeRegressor(random_state=42)
```

[ ]:
```
# Make predictions on the testing data
```

[25]:
```
y_pred_tree = tree_model.predict(X_test)
```

[ ]:
```
# Evaluate the Decision Tree model
```

[26]:
```
mse_tree = mean_squared_error(y_test, y_pred_tree)
mae_tree = mean_absolute_error(y_test, y_pred_tree)
r2_tree = r2_score(y_test, y_pred_tree)
```

[27]:
```
print("Decision Tree Model:")
print("Mean Squared Error:", mse_tree)
print("Mean Absolute Error:", mae_tree)
print("R-squared Score:", r2_tree)
```

```
Decision Tree Model:
Mean Squared Error: 1678.1163417214373
Mean Absolute Error: 12.115441063420867
R-squared Score: -0.03476557402029423
```