EARTHQUAKE PREDICTION MODEL USING PYTHON

Phase-4 Documentation Submission

Team Member:

Balaji E

422521104006

Introduction:

- Earthquakes are natural disasters that can cause significant damage and loss of life, making their accurate prediction a matter of utmost importance for public safety and disaster preparedness.
- This project presents the development of an earthquake prediction model using Python.

Content for Project Phase 4:

- In this part we will visualize the data on a world map.
- Begin building the earthquake prediction model by Splitting it into training and testing sets.

Data Source:

- The data is provided by the United States Geological Survey (USGS), which monitors and reports on earthquake activity worldwide.
- The dataset contains information about worldwide earthquake events, including location, time, magnitude, depth, and more. It is a comprehensive collection of earthquake-related data**.**
- We also need to use another dataset to visualize tectonic plates.

Dataset Link:

https://www.kaggle.com/datasets/usgs/earthquake-database

https://www.kaggle.com/datasets/cwthompson/tectonic-plate-boundaries

```
[6]: df=pd.read_csv("database.csv")
     df.head()
```

[6]:

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	...	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Error	Root Mean Square
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	...	NaN	NaN	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	NaN	NaN
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	...	NaN	NaN	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	NaN	NaN
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	NaN	NaN

5 rows × 21 columns

Data Preprocessing:

- Data preprocessing is a crucial step in machine learning. It involves cleaning, transforming, and organizing raw data into a format that is suitable for the training model.

- The primary goals of data preprocessing are to improve data quality, reduce NaN, and make the data more amenable to training a model.

Steps to Preprocessing Earthquake Dataset:

1. Check the Longitude and Latitude degree.
2. Select the relevant column from the dataset.
3. Convert the date and time into the same format.
4. Convert date and time into TimeStamp.
5. Remove NaN row or data from a dataset.
6. Check for magnitude max and min (2.5 to 9.1).

Data Visualization:

- In this part, we are going to visualize the magnitude of earthquakes on a world map.
- First, we need to display the tectonic plates on the map.
- Then, we need to plot the magnitudes at various frequencies on the map.

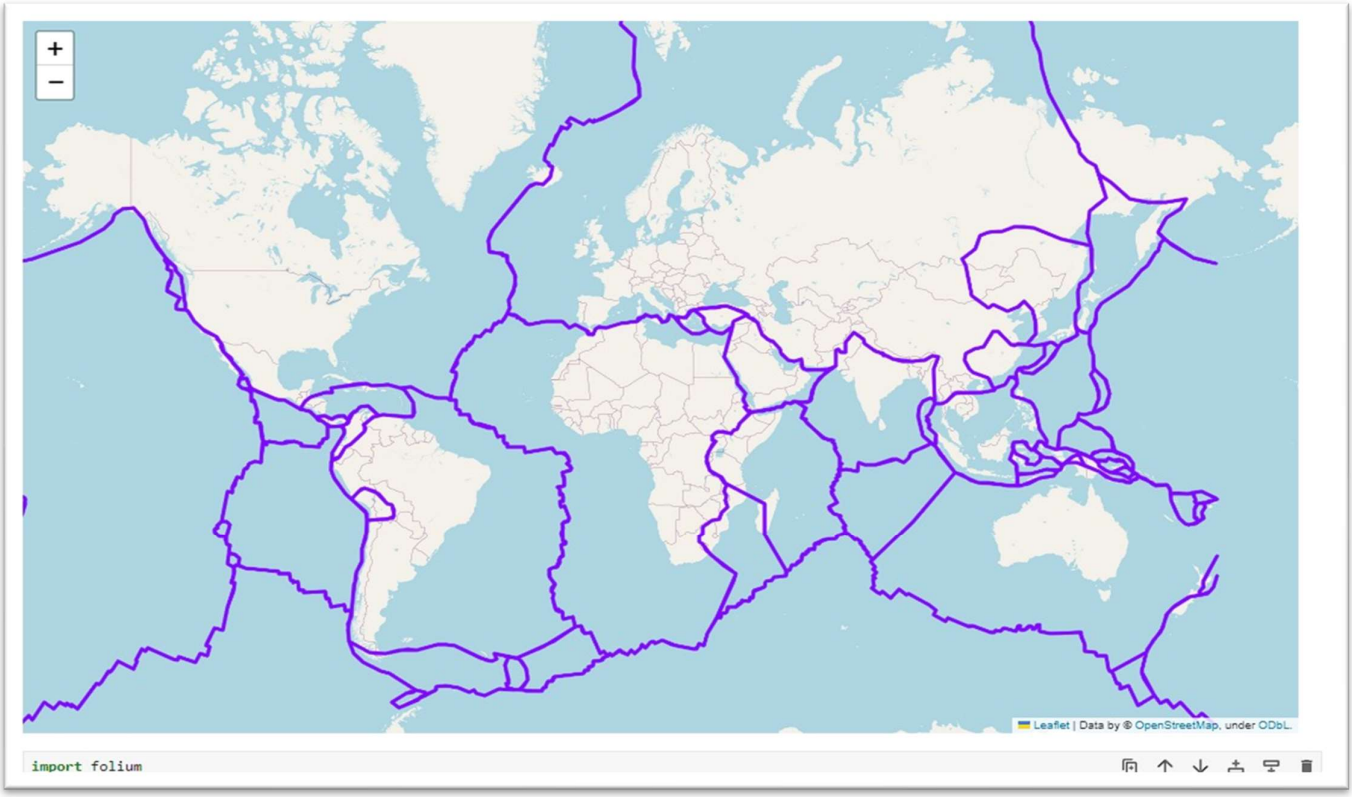
Visualizations on World Map:

- World map with Tectonic plates marking.
- World map with Magnitude marking.
- World map with Magnitude and Tectonic plates marking.

World map with Tectonic plates marking:

- Install the necessary libraries, including Folium, pandas, and geopandas.
- Create a Folium map centered at a location of your choice (e.g., the world) with an appropriate zoom level.
- Use the tectonic plate boundary data to draw lines or polygons representing plate boundaries on the m.

```
import folium
import pandas as pd
import geopandas as gpd


data=pd.read_csv('resultdata.csv')

tectonic_plates = pd.read_csv('all.csv')

m = folium.Map(tiles="cartodbpositron",location=[0,0], zoom_start=5)

plates = list(tectonic_plates["plate"].unique())
for plate in plates:
    plate_vals = tectonic_plates[tectonic_plates["plate"] == plate]
    lats = plate_vals["lat"].values
    lons = plate_vals["lon"].values
    points = list(zip(lats, lons))
    indexes = [None] + [i + 1 for i, x in enumerate(points) if i < len(points) - 1 and abs(x[1] - points[i + 1][1]) > 300] + [None]
    for i in range(len(indexes) - 1):
        folium.vector_layers.PolyLine(points[indexes[i]:indexes[i+1]], popup=plate, color="#7F00FF", fill=False, ).add_to(m)

m.save('earthquake_map_plates1.1.html')  # Save as HTML

m
```



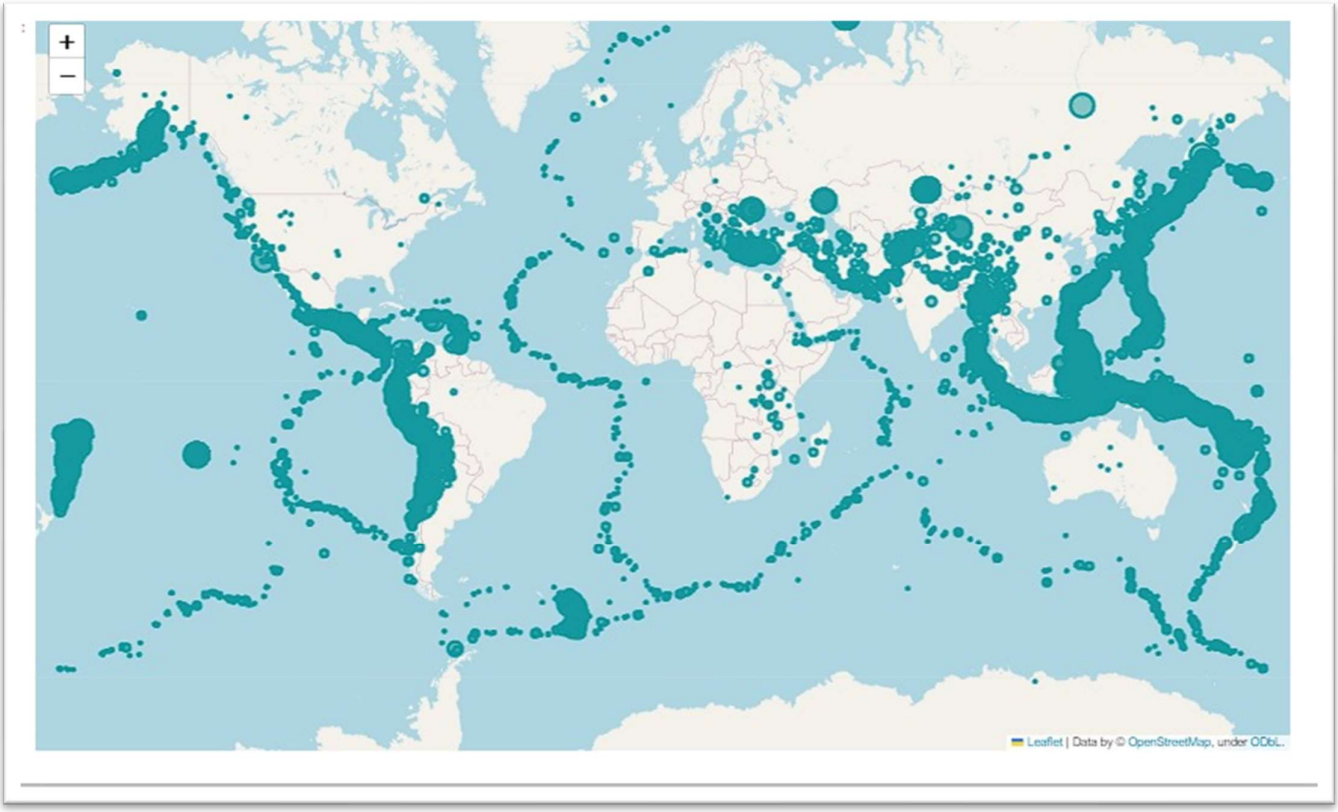
World map with Magnitude marking:

- **Magnitude as Color:**
  - In this visualization, earthquake magnitude is represented by the color of markers or symbols on the map. Typically, a color scale is used to indicate the range of magnitudes.
  - The specific color scheme and intervals may vary depending on the visualization, but the key is to provide a visual indication of the earthquake's strength.

- **Depth as Size or Circle:**
  - In this approach, the depth of an earthquake is represented by the size of markers or circles on the map. Deeper earthquakes may have larger markers, while shallower earthquakes have smaller ones.
  - The exact scaling and size intervals can vary, but the objective is to give viewers a sense of the depth of each seismic event.

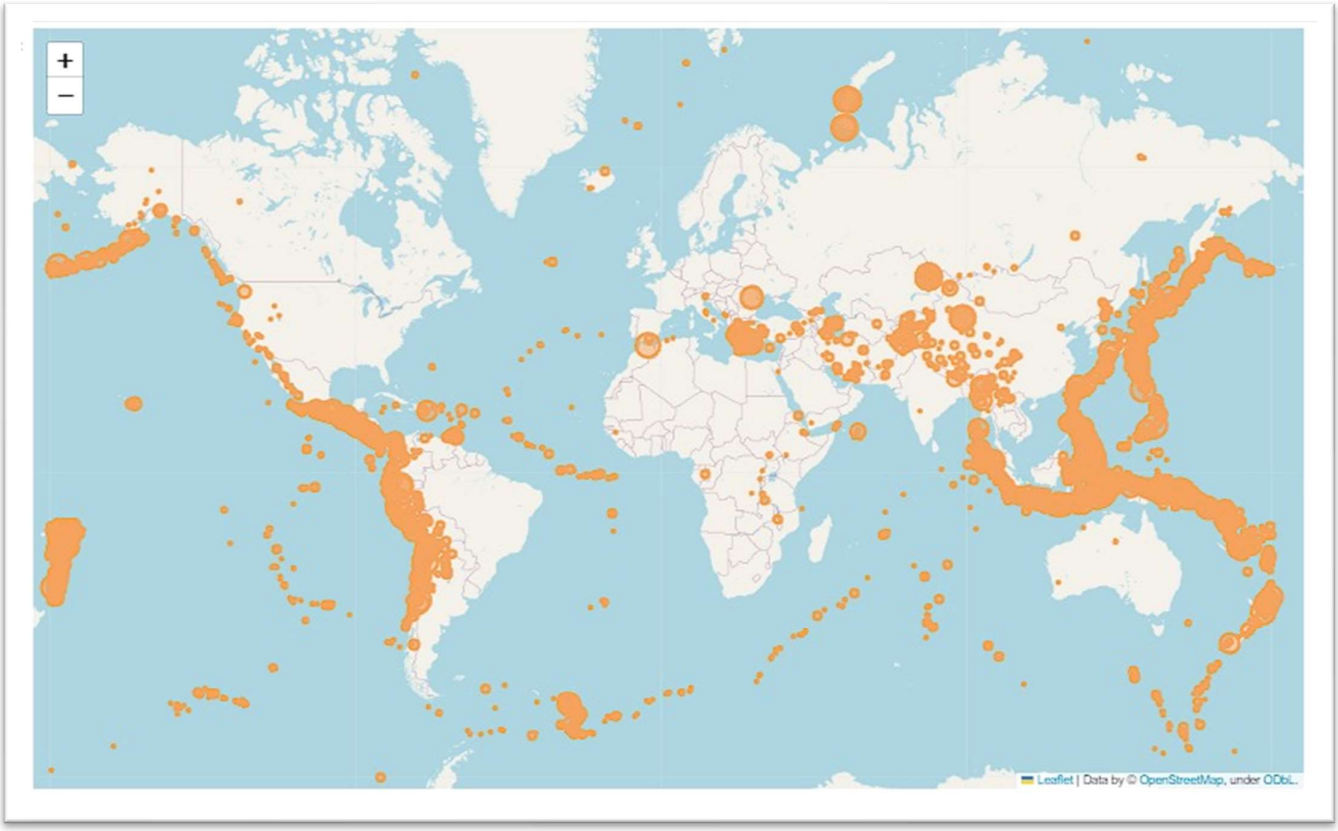
For Magnitude range from 5.5 to 6.0:

```
import folium
import pandas as pd

# Load earthquake data
earthquake_data = pd.read_csv('resultdata.csv')

# Filter earthquakes with magnitude less than 7
filtered_earthquakes = earthquake_data[(earthquake_data['Magnitude'] > 5.5) & (earthquake_data['Magnitude']<6)]

# Create a Folium map centered at a specific location
m = folium.Map(location=[0, 0], zoom_start=2)

# Add markers for the filtered earthquakes
for _, earthquake in filtered_earthquakes.iterrows():
    # Customize marker color and size based on magnitude and depth
    magnitude = earthquake['Magnitude']
    depth = earthquake['Depth']
    latitude = earthquake['Latitude']
    longitude = earthquake['Longitude']
    timestamp = earthquake['Timestamp']
    d = depth

    # Determine marker color based on magnitude


    # Determine marker size based on depth
    depth_size = d / 10 if d < 100 else d / 200

    # Add earthquake marker to the map
    folium.CircleMarker(
        location=[latitude, longitude],
        radius=depth_size,
        color='#0F969C',
        fill=True,
        fill_color='#0F969C',
        fill_opacity=0.5,
        popup=f'Depth: {depth} km<br>Magnitude: {magnitude}<br>Timestamp: {timestamp}',
    ).add_to(m)

# Save the map as an HTML file
m.save('earthquake_map_magnitude_lt_7.html')
m
```



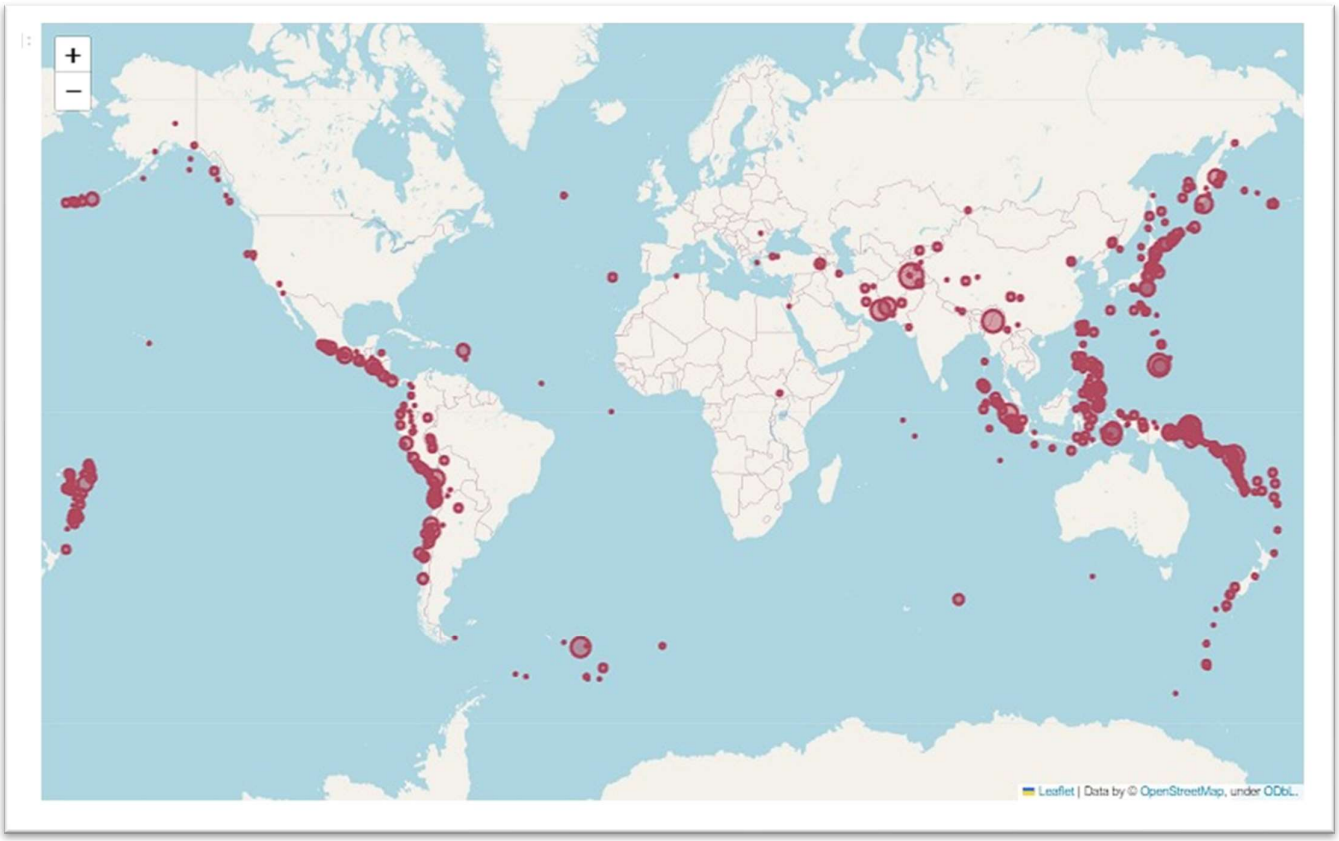
For Magnitude range from 6.0 to 7.0:

```
import folium
import pandas as pd

# Load earthquake data
earthquake_data = pd.read_csv('resultdata.csv')

# Filter earthquakes with magnitude less than 7
filtered_earthquakes = earthquake_data[(earthquake_data['Magnitude'] > 6) & (earthquake_data['Magnitude']<7)]

# Create a Folium map centered at a specific location
m = folium.Map(location=[0, 0], zoom_start=2)

# Add markers for the filtered earthquakes
for _, earthquake in filtered_earthquakes.iterrows():
    # Customize marker color and size based on magnitude and depth
    magnitude = earthquake['Magnitude']
    depth = earthquake['Depth']
    latitude = earthquake['Latitude']
    longitude = earthquake['Longitude']
    timestamp = earthquake['Timestamp']
    d = depth

    # Determine marker color based on magnitude


    # Determine marker size based on depth
    depth_size = d / 10 if d < 100 else d / 200

    # Add earthquake marker to the map
    folium.CircleMarker(
        location=[latitude, longitude],
        radius=depth_size,
        color='#F39F5A',
        fill=True,
        fill_color='#F39F5A',
        fill_opacity=0.5,
        popup=f'Depth: {depth} km<br>Magnitude: {magnitude}<br>Timestamp: {timestamp}',
    ).add_to(m)

# Save the map as an HTML file
m.save('earthquake_map_magnitude_lt_7.html')
m
```



For Magnitude range greater than 7.0:

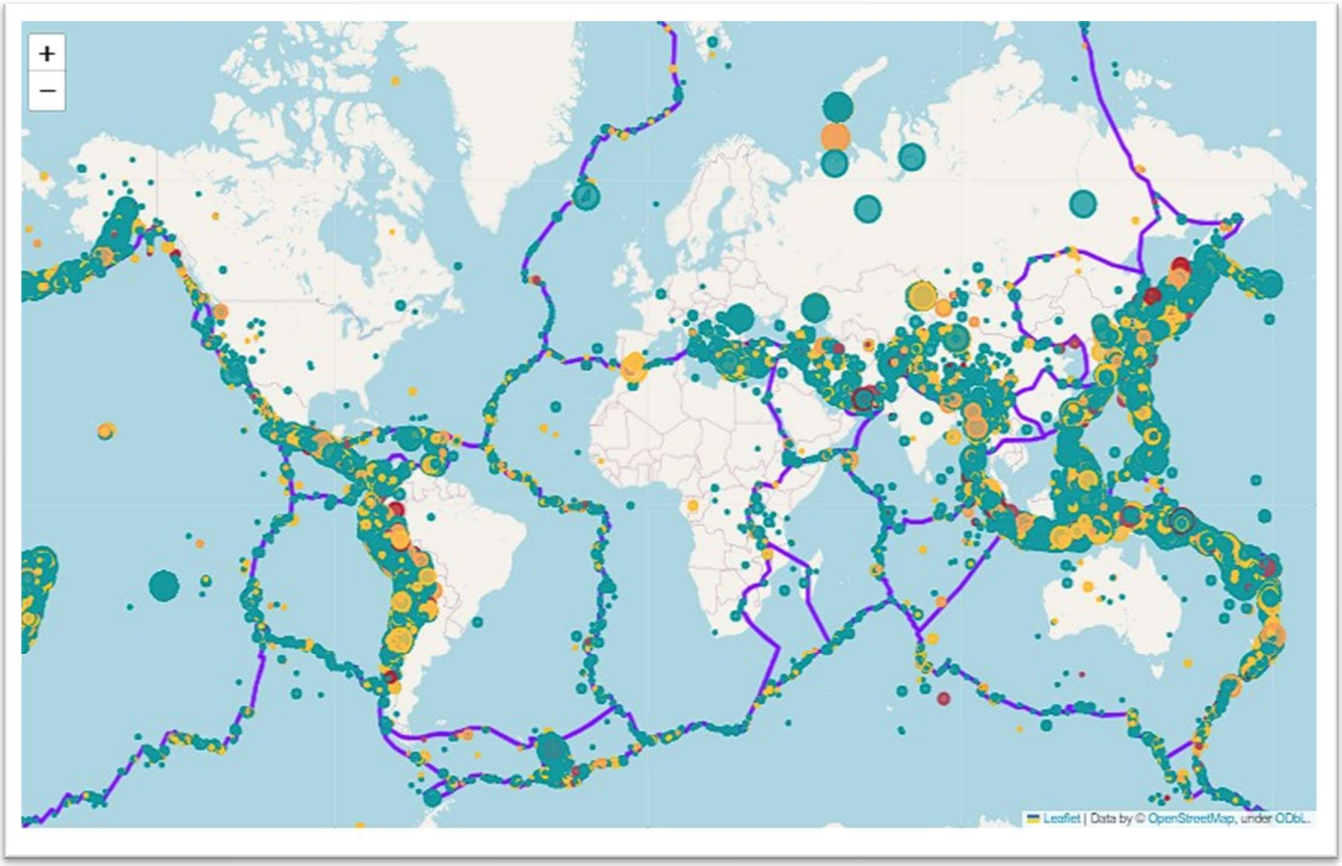
```
import folium
import pandas as pd

# Load earthquake data
earthquake_data = pd.read_csv('resultdata.csv')

# Filter earthquakes with magnitude less than 7
filtered_earthquakes = earthquake_data[(earthquake_data['Magnitude'] > 7) & (earthquake_data['Magnitude']<9)]

# Create a Folium map centered at a specific location
m = folium.Map(location=[0, 0], zoom_start=2)

# Add markers for the filtered earthquakes
for _, earthquake in filtered_earthquakes.iterrows():
    # Customize marker color and size based on magnitude and depth
    magnitude = earthquake['Magnitude']
    depth = earthquake['Depth']
    latitude = earthquake['Latitude']
    longitude = earthquake['Longitude']
    timestamp = earthquake['Timestamp']
    d = depth

    # Determine marker color based on magnitude


    # Determine marker size based on depth
    depth_size = d / 10 if d < 100 else d / 200

    # Add earthquake marker to the map
    folium.CircleMarker(
        location=[latitude, longitude],
        radius=depth_size,
        color='#AE445A',
        fill=True,
        fill_color='#AE445A',
        fill_opacity=0.5,
        popup=f'Depth: {depth} km<br>Magnitude: {magnitude}<br>Timestamp: {timestamp}',
    ).add_to(m)

# Save the map as an HTML file
m.save('earthquake_map_magnitude_lt_7.html')
m
```



World map with Magnitude and Tectonic plates marking:

- Earthquake visualization is a powerful and informative approach to understanding seismic activity and its geospatial context.
- It involves the use of maps, colors, markers, and interactive tools to represent key earthquake-related data, including earthquake locations, magnitudes, depths, and their relationship to tectonic plate boundaries.
- This visualization technique provides a clear and accessible way to comprehend and analyze earthquake patterns, assess seismic risks, and make informed decisions related to earthquake preparedness and response.

```
import folium
import pandas as pd
import geopandas as gpd

data=pd.read_csv('resultdata.csv')
tectonic_plates = pd.read_csv('all.csv')
m = folium.Map(location=[0,0], zoom_start=2)

plates = list(tectonic_plates["plate"].unique())
for plate in plates:
    plate_vals = tectonic_plates[tectonic_plates["plate"] == plate]
    lats = plate_vals["lat"].values
    lons = plate_vals["lon"].values
    points = list(zip(lats, lons))
    indexes = [None] + [i + 1 for i, x in enumerate(points) if i < len(points) - 1 and abs(x[1] - points[i + 1][1]) > 300] + [None]
    for i in range(len(indexes) - 1):
        folium.vector_layers.PolyLine(points[indexes[i]:indexes[i+1]], popup=plate, color="#7F00FF", fill=False, ).add_to(m)

for index, row in data.iterrows():
    magnitude = row['Magnitude']
    depth = row['Depth']
    latitude = row['Latitude']
    longitude = row['Longitude']
    timestamp = row['Timestamp']
    d=depth

    # Customize marker color and size based on depth and magnitude
    # Example color scheme: green for depth, red for magnitude
    if magnitude < 5.5:
        magnitude_color = '#355070'
    elif 5.5 <= magnitude < 6.0:
        magnitude_color = '#0F969C'
    elif 6.0 <= magnitude < 6.5:
        magnitude_color = '#FFBA42'
    elif 6.5 <= magnitude < 7.0:
        magnitude_color = '#F39F5A'
    elif 7.0 <= magnitude < 7.5:
        magnitude_color = '#AE445A'
    else:
        magnitude_color = '#851A2B'
    depth_size = d/10 if d<100 else d/100  # Adjust the multiplier to control marker size

    # Add the earthquake marker to the map
    folium.CircleMarker(
        location=[latitude, longitude],
        radius=depth_size,
        color=magnitude_color,
        fill=True,
        fill_color=magnitude_color,
        fill_opacity=0.8,
        popup=f'Depth: {depth} km<br>Magnitude: {magnitude}<br>Timestamp: {timestamp}',
    ).add_to(m)
m
```



Data Spilitting:

- X_train and X_test are the feature (input) datasets for the training and testing sets, respectively.
- y_train and y_test are the label (output) datasets for the training and testing sets, respectively.
- we use 20% of the data will be used for testing, and the remaining 80% will be used for training the model.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    input_data, target, test_size=0.2, random_state=42)
```

Conclusion:

- In conclusion, earthquake magnitude prediction and visualization are critical components of earthquake monitoring and preparedness.
- They empower individuals, organizations, and communities to better understand seismic risks and make informed decisions to mitigate the impact of earthquakes.
- By combining data analysis, machine learning, and geospatial visualization, we enhance our ability to comprehend, predict, and respond to seismic events.