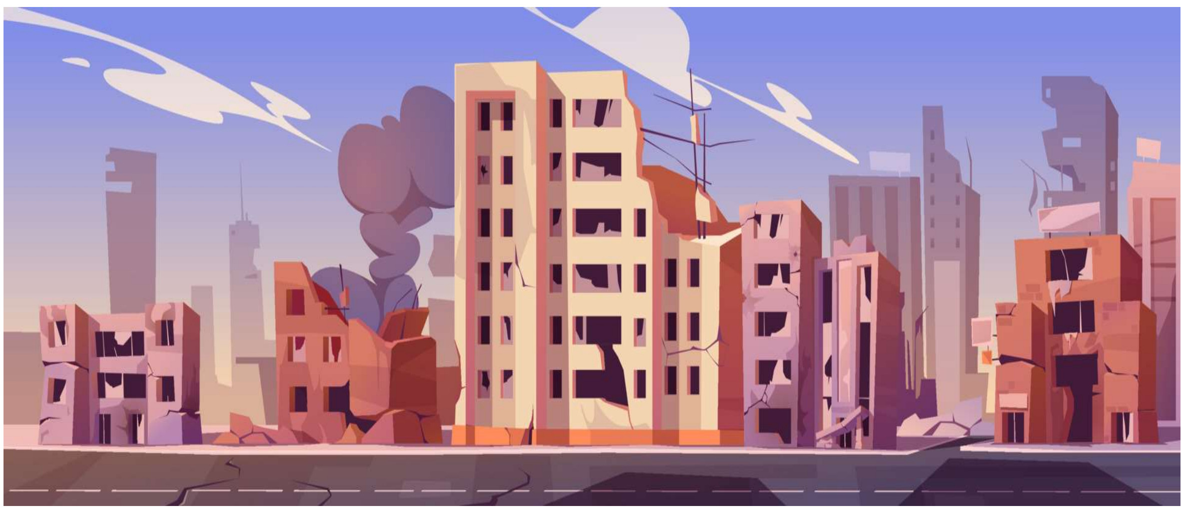
EARTHQUAKE PREDICTION MODEL USING PYTHON

Phase-3 Documentation Submission

Team Leader:

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 3:

- In this part you will begin building your project by loading and preprocessing the dataset.
- Begin building the earthquake prediction model by loading and preprocessing the dataset.

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**.**

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

```
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

Attributes:

1. Date and Time: The date and time when the earthquake occurred. These attributes are usually recorded in a standardized format, such as UTC (Coordinated Universal Time).

   The dates are in dd/mm/yyyy and the time is in 24hrs format.

2. Location: The geographical coordinates (latitude and longitude) of the earthquake's epicenter. This information is crucial for pinpointing the earthquake's location.

3. Depth: The depth of the earthquake's hypocentre (focus) below the Earth's surface. It is typically measured in kilometre’s.

4. Magnitude: The earthquake's magnitude, which quantifies its size and energy release. Common magnitude scales include the Richter scale (ML), the moment magnitude scale (Mw), and others.

Key Feature of Dataset:

- This dataset stored more than one earthquake that happened in a single day.
- The dataset contains data from the year of 1965 to 2016.
- This dataset helps us to study the earthquake pattern and make prediction easier.

Load Dataset

Using the pandas library, we can load a dataset into the Python program.

To load a CSV file into Python, we use pandas.read_csv("path of file")

```
import pandas as pd
# load a dataset
data = pd.read_csv('database.csv')
df
```

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)

1. Checking for maximum and minimum of Latitude and Longitude

- Always the Latitude and Longitude lies between constant range.
- Latitude should be between -90 to 90 degrees.
- Longitude should be lies between -180 to 180 degrees.

```
import pandas as pd

# Check Latitude values between -90 to 90
df[df['Latitude'].between(-90, 90)]

# Check Longitude values between -180 to 180
df[df['Longitude'].between(-180, 180)]

df.tail()
```

The .between(-90,90) method is used to check whether the value of latitude is falls between range or not. It returns a boolean result as ‘True’ or ‘False’.

It filters the dataset, retaining only the rows for the condition ‘True’. In other words, it keeps only the rows with valid latitude and longitude values

2. Select the relevant column from the dataset

- Remove the non-relevant column from the dataset
- Because it does not help the model to learn but instead makes the learning as complex.

```
[31]: data = data[['Date','Time','Longitude','Latitude','Depth','Magnitude']]
data
```

[31]:

	Date	Time	Longitude	Latitude	Depth	Magnitude
0	01/02/1965	13:44:18	145.6160	19.2460	131.60	6.0
1	01/04/1965	11:29:49	127.3520	1.8630	80.00	5.8
2	01/05/1965	18:05:58	-173.9720	-20.5790	20.00	6.2
3	01/08/1965	18:49:43	-23.5570	-59.0760	15.00	5.8
4	01/09/1965	13:32:50	126.4270	11.9380	15.00	5.8
...	...	...	...	...	...	...
23407	12/28/2016	08:22:12	-118.8941	38.3917	12.30	5.6
23408	12/28/2016	09:13:47	-118.8957	38.3777	8.80	5.5
23409	12/28/2016	12:38:51	140.4262	36.9179	10.00	5.9
23410	12/29/2016	22:30:19	118.6639	-9.0283	79.00	6.3
23411	12/30/2016	20:08:28	141.4103	37.3973	11.94	5.5

23412 rows × 6 columns

3. Convert date and time into same format

Dataset contains date different format like dd-mm-yyyy, dd/mm/yyyy. So, convert date and time into same format using datetime library.

```
import pandas as pd

df = pd.read_csv('database.csv')

def formatconvert(date_str):
        return pd.to_datetime(date_str).strftime('%d/%m/%Y')

df['Date'] = df['Date'].apply(formatconvert)


df['Date']
```

```
0        02/01/1965
1        04/01/1965
2        05/01/1965
3        08/01/1965
4        09/01/1965
            ...
23406    28/12/2016
23407    28/12/2016
23408    28/12/2016
23409    29/12/2016
23410    30/12/2016
Name: Date, Length: 23411, dtype: object
```

4.Remove nan row or data from dataset

- Check for Not a Number in the given dataset. If any nan present in dataset, remove the row from the dataset.
- To remove rows or data with NaN values from a dataset in python
- We use dropna() from pandas. it returns a Boolean result

```
[60]: # Count the number of missing (NaN) values in each column
      nan_counts = data.isna().sum()

      # Print the results
      print(nan_counts)

      Latitude     0
      Longitude    0
      Depth        0
      Magnitude    0
      Timestamp    0
      dtype: int64
```

5. Convert date and time into TimeStamp

Timestamps are numerical values representing a specific point in time, usually in seconds or milliseconds. It provide a consistent and standardized way to represent time across different systems and programming language.

The timestamps represent the number of seconds since the Unix epoch from January 1, 1970, to till date so, drop the before it.

```
[39]: df=pd.DataFrame(df)

      df = df.drop(df.index[0:1457])
      df.reset_index(drop=True, inplace=True)
      df
```

[39]:

	Date	Time	Latitude	Longitude	Depth	Magnitude
0	01/04/1970	17:00:41	24.1850	102.5430	11.30	7.1
1	01/05/1970	11:49:10	23.9840	102.7320	15.00	5.9
2	01/06/1970	5:35:54	-9.5830	151.4930	15.00	6.3
3	01/07/1970	7:56:14	15.7850	-59.8080	36.70	6.0
4	01/08/1970	17:12:41	-34.8500	178.7820	199.40	6.8
...	...	...	...	...	...	...
21949	12/28/2016	8:22:12	38.3917	-118.8941	12.30	5.6
21950	12/28/2016	9:13:47	38.3777	-118.8957	8.80	5.5
21951	12/28/2016	12:38:51	36.9179	140.4262	10.00	5.9

- We can create a new column by combining the 'Date' and 'Time' Column into single timestamp using pd.to_datetime.strptime()
- Using mktime() we can convert date and time into timestamp.
- Finally remove the date and time column from the dataset.

```
[56]: import datetime
      import time

      timestamp = []
      for d, t in zip(df['Date'], df['Time']):
          ts = datetime.datetime.strptime(d+' '+t, '%d/%m/%Y %H:%M:%S')
          timestamp.append(time.mktime(ts.timetuple()))

      timeStamp = pd.Series(timestamp)
```

```
[57]: df['Timestamp'] = timeStamp.values
      data = df.drop(['Date', 'Time'], axis=1)
      data = data[data.Timestamp != 'ValueError']
      data
```

[57]:

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	24.1850	102.5430	11.30	7.1	3.006410e+05
1	23.9840	102.7320	15.00	5.9	3.683500e+05
2	-9.5830	151.4930	15.00	6.3	4.323540e+05
3	15.7850	-59.8080	36.70	6.0	5.271740e+05
4	-34.8500	178.7820	199.40	6.8	6.469610e+05
...	...	...	...	...	...
21949	38.3917	-118.8941	12.30	5.6	1.482894e+09
21950	38.3777	-118.8957	8.80	5.5	1.482897e+09
21951	36.9179	140.4262	10.00	5.9	1.482909e+09

6.Check for magnitude max and min (2.5 to 9.1)

Find Maximun and Minimun value for magnitude because the magnitude value cannot be negative or more than 10.

Check for magnitude value within 2.5 to 9.1, if anything exceeds delete the row or data from the dataset.

```
[68]: data['Magnitude'].max()
[68]: 9.1
[67]: data['Magnitude'].min()
[67]: 5.5
[69]: data[data['Magnitude'].between(5.5, 9.1)]
[69]:
```

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	24.1850	102.5430	11.30	7.1	3.006410e+05
1	23.9840	102.7320	15.00	5.9	3.683500e+05
2	-9.5830	151.4930	15.00	6.3	4.323540e+05
3	15.7850	-59.8080	36.70	6.0	5.271740e+05
4	-34.8500	178.7820	199.40	6.8	6.469610e+05
...	...	...	...	...	...
21949	38.3917	-118.8941	12.30	5.6	1.482894e+09
21950	38.3777	-118.8957	8.80	5.5	1.482897e+09
21951	36.9179	140.4262	10.00	5.9	1.482909e+09
21952	-9.0283	118.6639	79.00	6.3	1.483031e+09
21953	37.3973	141.4103	11.94	5.5	1.483109e+09

21954 rows × 5 columns

7.Display the data after preprocessing

Dataset after preprocessing is done. Save the file in csv format using to_csv().

The preprocessed dataset is use for train the model.

[72]:
```
data.to_csv("resultdata.csv",index='False')
data
```

[72]:

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	24.1850	102.5430	11.30	7.1	3.006410e+05
1	23.9840	102.7320	15.00	5.9	3.683500e+05
2	-9.5830	151.4930	15.00	6.3	4.323540e+05
3	15.7850	-59.8080	36.70	6.0	5.271740e+05
4	-34.8500	178.7820	199.40	6.8	6.469610e+05
...	...	...	...	...	...
21949	38.3917	-118.8941	12.30	5.6	1.482894e+09
21950	38.3777	-118.8957	8.80	5.5	1.482897e+09
21951	36.9179	140.4262	10.00	5.9	1.482909e+09
21952	-9.0283	118.6639	79.00	6.3	1.483031e+09
21953	37.3973	141.4103	11.94	5.5	1.483109e+09

21954 rows × 5 columns

Conclusion:

- The preprocessing steps are foundational in preparing the earthquake dataset for further analysis and model development.
- By cleaning, organizing, and standardizing the data, we create a solid foundation for accurate and meaningful predictions regarding earthquake occurrences and magnitudes.
- This dataset can now be used for exploratory data analysis, feature engineering, and the development of machine learning models.
- As we progress further, we will dive into more advanced aspects of machine learning and develop a model that can potentially contribute to earthquake forecasting and risk reduction.