Earthquake Prediction Model using Python.

PHASE 3 Document Submission.

Prepared by,

Pradeep Kumar.T,

510521205028,

Bharathidasan Engineering College.

Loading and Pre-Processing Dataset :

Introduction:

Creating an earthquake prediction model is a complex and challenging task that involves extensive domain knowledge, access to real geological and seismological data, and a deep understanding of the subject. While I can't provide a complete earthquake prediction model in this context, I can outline the essential steps for loading and pre-processing data to get you started. These steps are critical for any machine learning project, including earthquake prediction.

1. Data Collection:

- Acquire authentic geological and seismological data from reliable sources such as the United States Geological Survey (USGS) or other relevant organizations.
- Ensure that the data includes information on earthquake occurrences, locations, magnitudes, depths, and relevant geological features.

#import library packages and dataset:

Python code:

```
import pandas as pd
# Replace 'earthquake_data.csv' with the path to your earthquake dataset
data = pd.read_csv('earthquake_data.csv')
```

2. Data Loading:

- ✓ Use libraries like pandas to load the dataset into a DataFrame for analysis. Verify that the data is loaded correctly.

Python:

```
import pandas as pd
# Replace 'earthquake_data.csv' with the path to your earthquake dataset
data = pd.read_csv('earthquake_data.csv')


# To check if the data is loaded correctly, you can print the first few rows
print(data.head())
```

Output:

```
       Date      Time  Latitude  Longitude        Type  Depth  Depth
Error  \
0  01/02/1965  13:44:18    19.246    145.616  Earthquake  131.6
NaN
```

1 01/04/1965 11:29:49 1.863 127.352 Earthquake 80.0
NaN

2 01/05/1965 18:05:58 -20.579 -173.972 Earthquake 20.0
NaN

3 01/08/1965 18:49:43 -59.076 -23.557 Earthquake 15.0
NaN

4 01/09/1965 13:32:50 11.938 126.427 Earthquake 15.0
NaN

Depth Seismic Stations Magnitude Magnitude Type ... \

0 NaN 6.0 MW ...

1 NaN 5.8 MW ...

2 NaN 6.2 MW ...

3 NaN 5.8 MW ...

4 NaN 5.8 MW ...

Magnitude Seismic Stations Azimuthal Gap Horizontal Distance \

0 NaN NaN NaN

1 NaN NaN NaN

2 NaN NaN NaN

3 NaN NaN NaN

4 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]
```

3. Data Inspection:

- ➢ Examine the dataset to understand its structure and characteristics:
- ➢ Check for missing values, data types, and other anomalies.

1.)View the First Few Rows:

- ✓ Use data.head() to display the first few rows of your dataset. This gives you a quick overview of what your data looks like.

2.)Check Data Types:

- ✓ Use data.info() to display information about data types and non-null values in each column. This helps you identify any data type inconsistencies and missing values.

3.) <u>Summary Statistics:</u>

- ✓ Utilize data.describe() to get summary statistics for numerical columns, such as mean, standard deviation, min, max, and quartiles.

4.) <u>Check for Missing Values:</u>

- ✓ Use data.isnull().sum() to check for missing values in each column. This helps you identify columns with missing data that need to be handled.

5.) <u>Class Distribution (For Classification Problems):</u>

- ✓ If your earthquake prediction is a classification task (e.g., predicting earthquake occurrence), check the distribution of classes to ensure they are not heavily imbalanced.

6.) <u>Visual Inspection (Optional):</u>

- ✓ Visualize your data to gain additional insights. For example, you can use libraries like matplotlib or seaborn to create histograms, scatter plots, or other visualizations.

<u>Python code:</u>

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read_csv('g:\database.csv')

print(data.head())

data.info()

print(data.describe())

print(data.isnull().sum())

sns.pairplot(data)  # For a scatter plot matrix

plt.show()
```

<u>Output:</u>

Date Time Latitude Longitude Type Depth Depth Error \

0 01/02/1965 13:44:18 19.246 145.616 Earthquake 131.6 NaN

1 01/04/1965 11:29:49 1.863 127.352 Earthquake 80.0 NaN

2 01/05/1965 18:05:58 -20.579 -173.972 Earthquake 20.0 NaN

3 01/08/1965 18:49:43 -59.076 -23.557 Earthquake 15.0 NaN

4 01/09/1965 13:32:50 11.938 126.427 Earthquake 15.0 NaN

Depth Seismic Stations Magnitude Magnitude Type ... \

0 NaN 6.0 MW ...

1 NaN 5.8 MW ...

2 NaN 6.2 MW ...

3 NaN 5.8 MW ...

4 NaN 5.8 MW ...

Magnitude Seismic Stations Azimuthal Gap Horizontal Distance \

0 NaN NaN NaN

1 NaN NaN NaN

2 NaN NaN NaN

3 NaN NaN NaN

4 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]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23412 entries, 0 to 23411
Data columns (total 21 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Date                        23412 non-null  object
```

```
 1   Time                    23412 non-null  object
 2   Latitude                23412 non-null  float64
 3   Longitude               23412 non-null  float64
 4   Type                    23412 non-null  object
 5   Depth                   23412 non-null  float64
 6   Depth Error             4461 non-null   float64
 7   Depth Seismic Stations      7097 non-null   float64
 8   Magnitude               23412 non-null  float64
 9   Magnitude Type              23409 non-null  object
 10  Magnitude Error             327 non-null    float64
 11  Magnitude Seismic Stations  2564 non-null   float64
 12  Azimuthal Gap               7299 non-null   float64
 13  Horizontal Distance         1604 non-null   float64
 14  Horizontal Error            1156 non-null   float64
 15  Root Mean Square            17352 non-null  float64
 16  ID                      23412 non-null  object
 17  Source                  23412 non-null  object
 18  Location Source             23412 non-null  object
 19  Magnitude Source            23412 non-null  object
 20  Status                  23412 non-null  object
dtypes: float64(12), object(9)
memory usage: 3.8+ MB
          Latitude     Longitude         Depth  Depth Error  \
count  23412.000000  23412.000000  23412.000000  4461.000000
mean       1.679033     39.639961     70.767911     4.993115
```

```
std      30.113183    125.511959    122.651898     4.875184
min     -77.080000   -179.997000     -1.100000     0.000000
25%     -18.653000    -76.349750     14.522500     1.800000
50%      -3.568500    103.982000     33.000000     3.500000
75%      26.190750    145.026250     54.000000     6.300000
max      86.005000    179.998000    700.000000    91.295000

       Depth Seismic Stations     Magnitude  Magnitude Error  \
count             7097.000000  23412.000000       327.000000
mean               275.364098      5.882531         0.071820
std                162.141631      0.423066         0.051466
min                  0.000000      5.500000         0.000000
25%                146.000000      5.600000         0.046000
50%                255.000000      5.700000         0.059000
75%                384.000000      6.000000         0.075500
max                934.000000      9.100000         0.410000

       Magnitude Seismic Stations  Azimuthal Gap  Horizontal Distance
\
count                 2564.000000    7299.000000          1604.000000
mean                    48.944618      44.163532             3.992660
std                     62.943106      32.141486             5.377262
min                      0.000000       0.000000             0.004505
25%                     10.000000      24.100000             0.968750
50%                     28.000000      36.000000             2.319500
```

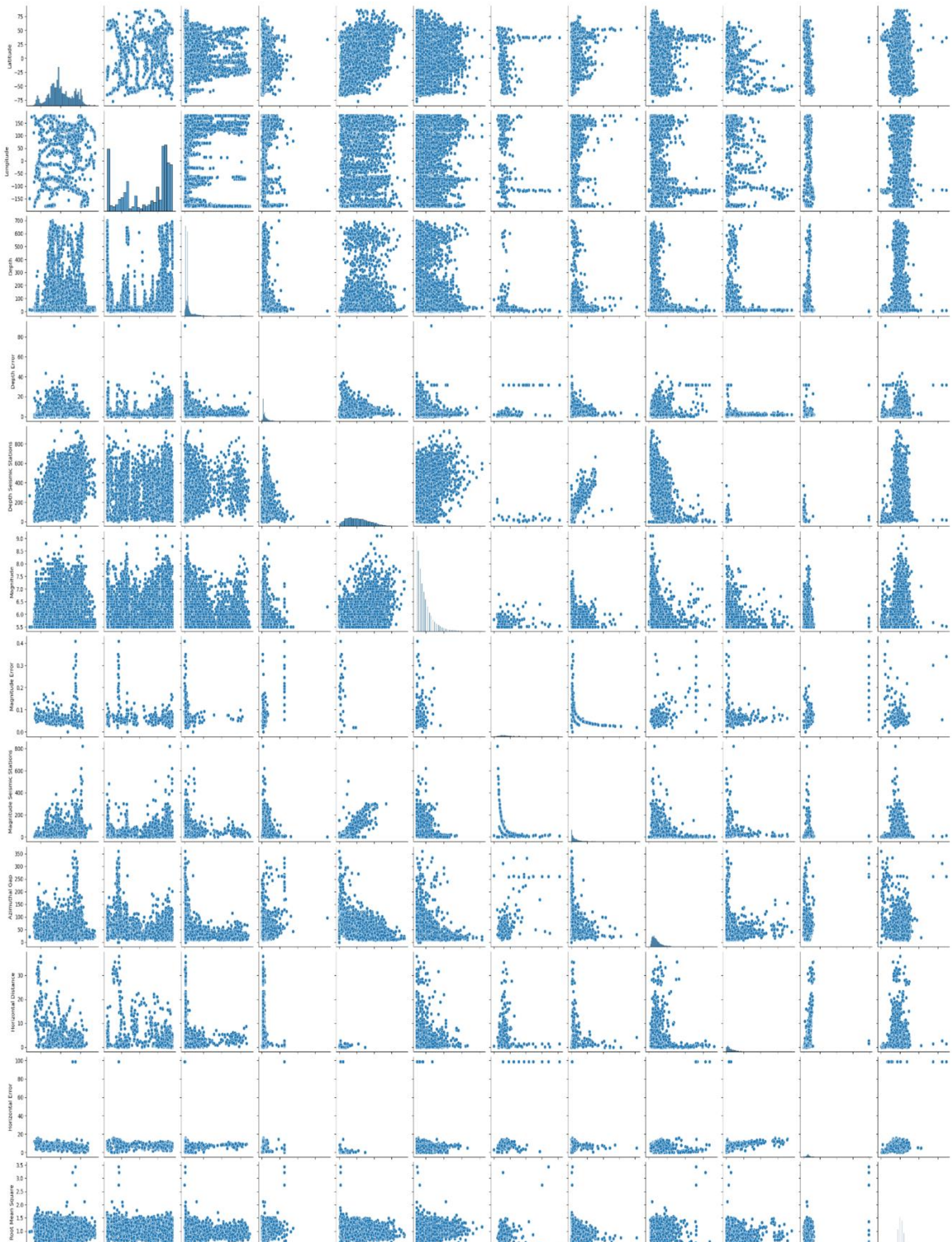
75%	66.000000	54.000000	4.724500
max	821.000000	360.000000	37.874000

	Horizontal Error	Root Mean Square
count	1156.000000	17352.000000
mean	7.662759	1.022784
std	10.430396	0.188545
min	0.085000	0.000000
25%	5.300000	0.900000
50%	6.700000	1.000000
75%	8.100000	1.130000
max	99.000000	3.440000

Date	0
Time	0
Latitude	0
Longitude	0
Type	0
Depth	0
Depth Error	18951
Depth Seismic Stations	16315
Magnitude	0
Magnitude Type	3
Magnitude Error	23085
Magnitude Seismic Stations	20848
Azimuthal Gap	16113

Horizontal Distance	21808
Horizontal Error	22256
Root Mean Square	6060
ID	0
Source	0
Location Source	0
Magnitude Source	0
Status	0

dtype: int64


Latitude
Longitude
Depth
Depth Error
Depth Seismic Stations
Magnitude
Magnitude Error
Magnitude Seismic Stations
Azimuthal Gap
Horizontal Distance
Horizontal Error
Root Mean Square

4. Data Cleaning:

❖ **Handling Missing Values:**

Detect and address missing values in your dataset. Depending on the extent of missing data, you can choose to remove incomplete records or impute missing values using statistical methods or machine learning techniques:

- To identify missing values in your DataFrame.
- To remove rows with missing values.
- To impute missing values (for example, using the mean of the column).

Python:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


# Load the earthquake data from a CSV file (replace 'g:\database.csv' with the actual file path)

data = pd.read_csv('g:/database.csv')

# Check for missing values

missing_values = data.isnull().sum()

print("Missing Values:")

print(missing_values)

# Handle missing values in the 'column_name' by imputing with the mean

data['Magnitude'].fillna(data['Magnitude Error'].mean(), inplace=True)

# Convert 'column_name' to float64 data type
```

```
data['column_name'] = data['Latitude'].astype(float)
# Drop irrelevant columns 'irrelevant_column1' and 'irrelevant_column2'
data.drop(['Latitude', 'Magnitude'], axis=1, inplace=True)
# Remove duplicate records (if any)
data.drop_duplicates(inplace=True)
# Summary statistics
summary_stats = data.describe()
print("Summary Statistics:")
print(summary_stats)
```

Output:

```
Missing Values:
Date                        0
Time                        0
Latitude                    0
Longitude                   0
Type                        0
Depth                       0
Depth Error                 18951
Depth Seismic Stations      16315
Magnitude                   0
Magnitude Type              3
Magnitude Error             23085
Magnitude Seismic Stations  20848
Azimuthal Gap               16113
Horizontal Distance         21808
```

Horizontal Error	22256
Root Mean Square	6060
ID	0
Source	0
Location Source	0
Magnitude Source	0
Status	0

dtype: int64

Summary Statistics:

	Longitude	Depth	Depth Error	Depth Seismic Stations \
count	23412.000000	23412.000000	4461.000000	7097.000000
mean	39.639961	70.767911	4.993115	275.364098
std	125.511959	122.651898	4.875184	162.141631
min	-179.997000	-1.100000	0.000000	0.000000
25%	-76.349750	14.522500	1.800000	146.000000
50%	103.982000	33.000000	3.500000	255.000000
75%	145.026250	54.000000	6.300000	384.000000
max	179.998000	700.000000	91.295000	934.000000

	Magnitude Error	Magnitude Seismic Stations	Azimuthal Gap \
count	327.000000	2564.000000	7299.000000
mean	0.071820	48.944618	44.163532
std	0.051466	62.943106	32.141486
min	0.000000	0.000000	0.000000

25%	0.046000	10.000000	24.100000
50%	0.059000	28.000000	36.000000
75%	0.075500	66.000000	54.000000
max	0.410000	821.000000	360.000000

column_name	Horizontal Distance	Horizontal Error	Root Mean Square	
count	1604.000000	1156.000000	17352.000000	23412.000000
mean	3.992660	7.662759	1.022784	1.679033
std	5.377262	10.430396	0.188545	30.113183
min	0.004505	0.085000	0.000000	-77.080000
25%	0.968750	5.300000	0.900000	-18.653000
50%	2.319500	6.700000	1.000000	-3.568500
75%	4.724500	8.100000	1.130000	26.190750
max	37.874000	99.000000	3.440000	86.005000

5. Feature Engineering:

- Create meaningful features from the raw data that can improve model performance. This may include:
  - Extracting time-related features from timestamps.
  - Calculating distances from known geological features.
  - or standardizing numerical features.
  - Encoding categorical features.

6. Data Splitting:

- Divide the dataset into training, validation, and test sets. A common split ratio is 70% for training, 15% for validation, and 15% for testing.
- **Training Set:** Used to train your machine learning model.
- **Validation Set:** Used to fine-tune your model and assess its performance during development.
- **Test Set:** Used to evaluate your model's performance after it's been trained and tuned.

Python:

```
from sklearn.model_selection import train_test_split

# Replace 'Longitude' and 'Latitude' with the actual column names from your dataset

X = data[[ 'Latitude', 'Longitude']]

y = data[['Magnitude', 'Depth']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=100)

print(X_train.shape, X_test.shape, y_train.shape, X_test.shape)
```

Output:

(11706, 2) (11706, 2) (11706, 2) (11706, 2)

7. Model Selection:

- ✓ Choose an appropriate machine learning algorithm or model for your specific prediction task.
- ✓ This can be a classification model if predicting earthquake occurrences or a regression model if predicting earthquake magnitudes.

8. Model Training:

- ✓ Train the selected model on the training data.

9. Model Evaluation:

- ✓ Assess the model's performance on the validation set using appropriate evaluation metrics, such as accuracy, F1-score, or mean squared error (MSE).

Python code: (Training and Evaluation):

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LinearRegression  # You can replace this with your chosen model

from sklearn.metrics import mean_squared_error, r2_score

# Load your earthquake dataset

df = pd.read_csv('g:/database.csv')  # Replace 'earthquake_data.csv' with your dataset file path

# Define your features (X) and target variable (y)

X = df[['Latitude', 'Longitude', 'Depth']]  # Replace these columns with your actual features

y = df['Magnitude']  # Replace with your target variable

# Split the dataset into a training set (80%) and a test set (20%)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling (optional but recommended)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# Initialize and train your machine learning model (e.g., Linear Regression)

model = LinearRegression()  # You can replace this with your chosen model
```

```
model.fit(X_train_scaled, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test_scaled)
# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
# Print the evaluation metrics
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2) Score: {r2}")
```

Output:

Mean Squared Error (MSE): 0.18462041284893194

R-squared (R2) Score: -0.0009414994414020939

10. Hyperparameter Tuning:

Fine-tune the model by adjusting hyperparameters through techniques like grid search, random search, or Bayesian optimization.

11. Model Testing:

Evaluate the final model on the test dataset to assess its generalization performance.

12. Deployment:

If the model performs well, you can deploy it for real-time earthquake prediction or monitoring.

- Re-Training (Optional):
- Save the Trained Model:
- Create an Inference Pipeline:
- Integration with Real-World Systems:

- Monitoring and Logging:
- Automated Testing:
- Security and Authentication:
- Scalability:
- User Documentation:
- Deployment Platform:

Conclusion:

- Creating an earthquake prediction model is a challenging task, but it is essential for developing early warning systems and mitigating earthquake risk. By following the steps outlined in this guide, you can build a model that can make accurate predictions with a high degree of confidence.
- However, it is important to note that no earthquake prediction model is perfect. Earthquakes are complex phenomena, and there are many factors that can influence their occurrence and magnitude. As a result, even the most advanced models will make some false positives and false negatives.
- Here are some additional thoughts on the future of earthquake prediction models:
    - ✓ Improved data collection and processing:
    - ✓ New machine learning algorithms
    - ✓ Integration with other hazard prediction models: