

# **EARTHQUAKE PREDICTION MODEL USING PYTHON**

BATCH MEMBER

TEAM-ID : Proj\_272172\_Team\_2

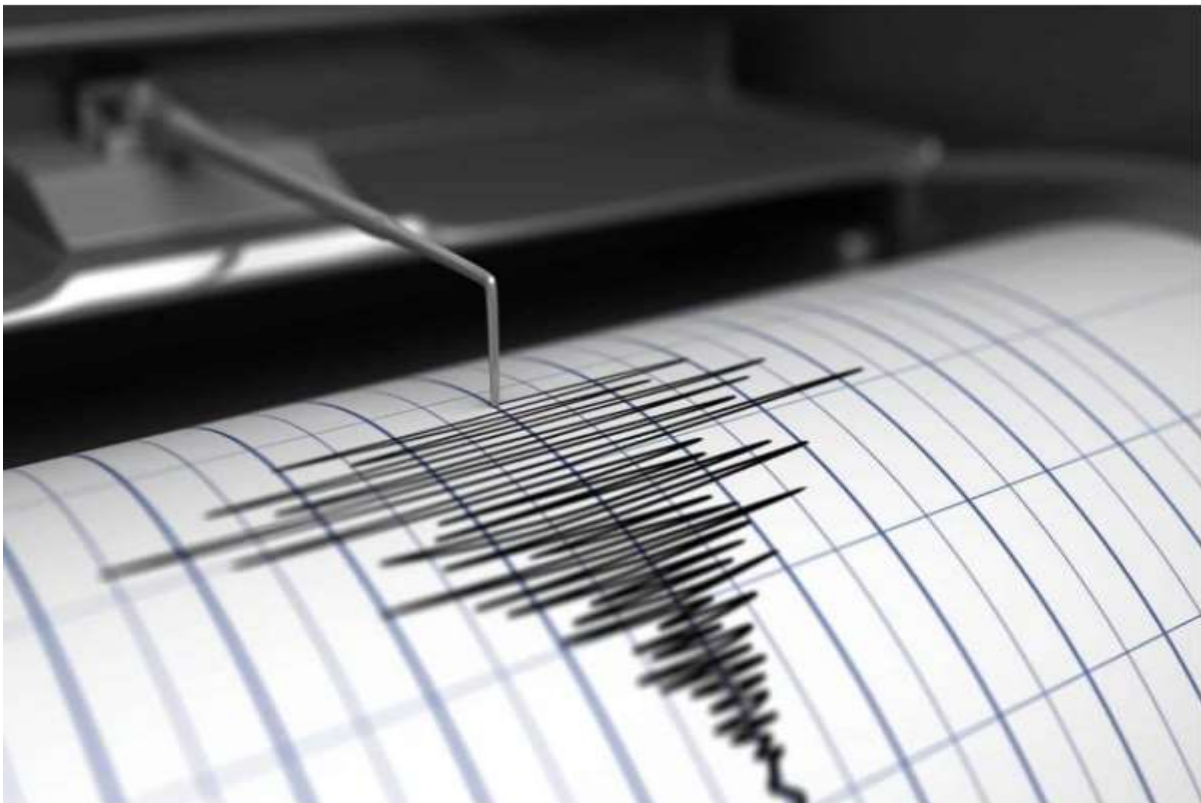
950621104097 : S.SOBIKA

Phase 5 Submission Document

**Project Title:** Earthquake Prediction Model Using Python.

**Phase 5:** Project Documentation & Submission.

**Topic:** In this section you will document the complete project and prepare it for submission.



# **EARTHQUAKE PREDICTION**

## **Introduction:**

- Earthquake prediction is a branch of the science of seismology that aims to specify the time, location, and magnitude of future earthquakes within certain limits.
- It is different from earthquake forecasting, which is the probabilistic assessment of general earthquake hazard in a given area over a long period of time.
- It is also different from earthquake warning systems, which provide a real-time alert of seconds to neighboring regions that might be affected by an earthquake.
- Earthquake prediction is a challenging and controversial topic, as there is no reliable method to predict earthquakes with high accuracy and precision.
- Many possible earthquake precursors have been proposed, such as changes in seismicity patterns, ground deformation, electromagnetic signals, geochemical anomalies, and animal behavior, but none of them have been proven to be consistent and reliable across different regions and scales.
- Some scientists are optimistic that with more research and data, prediction might be possible, while others are pessimistic and argue that earthquake prediction is inherently impossible.
- Prediction can be further distinguished from earthquake warning systems, which, upon detection of an earthquake, provide a real-time warning of seconds to neighboring regions that might be affected.
- Earthquake prediction is sometimes distinguished from earthquake forecasting, which can be defined as the probabilistic assessment of *general* earthquake hazard, including the frequency and magnitude of damaging earthquakes in a given area over years or decades.
- Not all scientists distinguish "prediction" and "forecast", but the distinction is useful.
- Extensive searches have reported many possible earthquake precursors, but, so far, such precursors have not been reliably identified across significant spatial and temporal scales.

## **Data Source :**


A good data source for earthquake prediction using machine learning should be accurate time, location, depth, magnitude.

Dataset Link: ( <https://www.kaggle.com/datasets/usgs/earthquake-database>)

## Given data set:

|       | Date       | Time     | Latitude | Longitude | Type       | Depth  | Depth<br>Error | Depth<br>Seismic<br>Stations | Magnitude | Magnitude<br>Type | ... | Magnitude<br>Seismic<br>Stations | Azimuthal<br>Gap | Horizontal<br>Distance | Horizontal<br>Error |
|-------|------------|----------|----------|-----------|------------|--------|----------------|------------------------------|-----------|-------------------|-----|----------------------------------|------------------|------------------------|---------------------|
| 0     | 01/02/1965 | 13:44:18 | 19.2460  | 145.6160  | Earthquake | 131.60 | NaN            | NaN                          | 6.0       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| 1     | 01/04/1965 | 11:29:49 | 1.8630   | 127.3520  | Earthquake | 80.00  | NaN            | NaN                          | 5.8       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| 2     | 01/05/1965 | 18:05:58 | -20.5790 | -173.9720 | Earthquake | 20.00  | NaN            | NaN                          | 6.2       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| 3     | 01/08/1965 | 18:49:43 | -59.0760 | -23.5570  | Earthquake | 15.00  | NaN            | NaN                          | 5.8       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| 4     | 01/09/1965 | 13:32:50 | 11.9380  | 126.4270  | Earthquake | 15.00  | NaN            | NaN                          | 5.8       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| ...   | ...        | ...      | ...      | ...       | ...        | ...    | ...            | ...                          | ...       | ...               | ... | ...                              | ...              | ...                    | ...                 |
| 23407 | 12/28/2016 | 08:22:12 | 38.3917  | -118.8941 | Earthquake | 12.30  | 1.2            | 40.0                         | 5.6       | ML                | ... | 18.0                             | 42.47            | 0.120                  | NaN                 |
| 23408 | 12/28/2016 | 09:13:47 | 38.3777  | -118.8957 | Earthquake | 8.80   | 2.0            | 33.0                         | 5.5       | ML                | ... | 18.0                             | 48.58            | 0.129                  | NaN                 |
| 23409 | 12/28/2016 | 12:38:51 | 36.9179  | 140.4262  | Earthquake | 10.00  | 1.8            | NaN                          | 5.9       | MWW               | ... | NaN                              | 91.00            | 0.992                  | 4.8                 |
| 23410 | 12/29/2016 | 22:30:19 | -9.0283  | 118.6639  | Earthquake | 79.00  | 1.8            | NaN                          | 6.3       | MWW               | ... | NaN                              | 26.00            | 3.553                  | 6.0                 |
| 23411 | 12/30/2016 | 20:08:28 | 37.3973  | 141.4103  | Earthquake | 11.94  | 2.2            | NaN                          | 5.5       | MB                | ... | 428.0                            | 97.00            | 0.681                  | 4.5                 |

23412 rows x 21 columns



*Here's a list of tools and software commonly used in the process:*

### **1. Programming Language:**

-Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like numpy, pandas, scikit-learn, and more.

### **2. Integrated Development Environment(IDE):**

-Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google Colab, or traditional IDEs like PyCharm.

### **3.Machine Learning Libraries:**

- You'll need various machine learning libraries, including:
- scikit-learn for building and evaluating machine learning models.

- TensorFlow or PyTorch for deep learning, if needed.
- XGBoost, LightGBM, or CatBoost for gradient boosting models.

#### **4.Data Visualization Tools:**

- Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.

#### **5.Data Preprocessing Tools:**

- Libraries like pandas help with data cleaning, manipulation, and preprocessing.

#### **6.Data Collection and Storage:**

- Depending on your data source, you might need web scraping tools or databases for data storage.

#### **7.Version Control:**

- Version control systems like Git are valuable for tracking changes in your code and collaborating with others.

#### **8.Notebooks and Documentation:**

- Tools for documenting your work, such as Jupyter Notebooks or Markdown for creating *README* files and documentation.

#### **9.Hyperparameter Tuning:**

- Tools like GridSearchCV or RandomizedSearchCV from scikit-learn can help with hyperparameter tuning.

#### **10.Web Development Tools(for Deployment):**

- If you plan to create a web application for model deployment, knowledge of web development tools like *Flask* or *Django* for backend development, and *HTML*, *CSS*, and *JavaScript* for the front-end can be useful.

#### **11.Cloud Services (for Scalability):**

- For large-scale applications, cloud platforms like AWS, Google Cloud, or

Azure can provide scalable computing and storage resources.

## **12.External Data Sources (if applicable):**

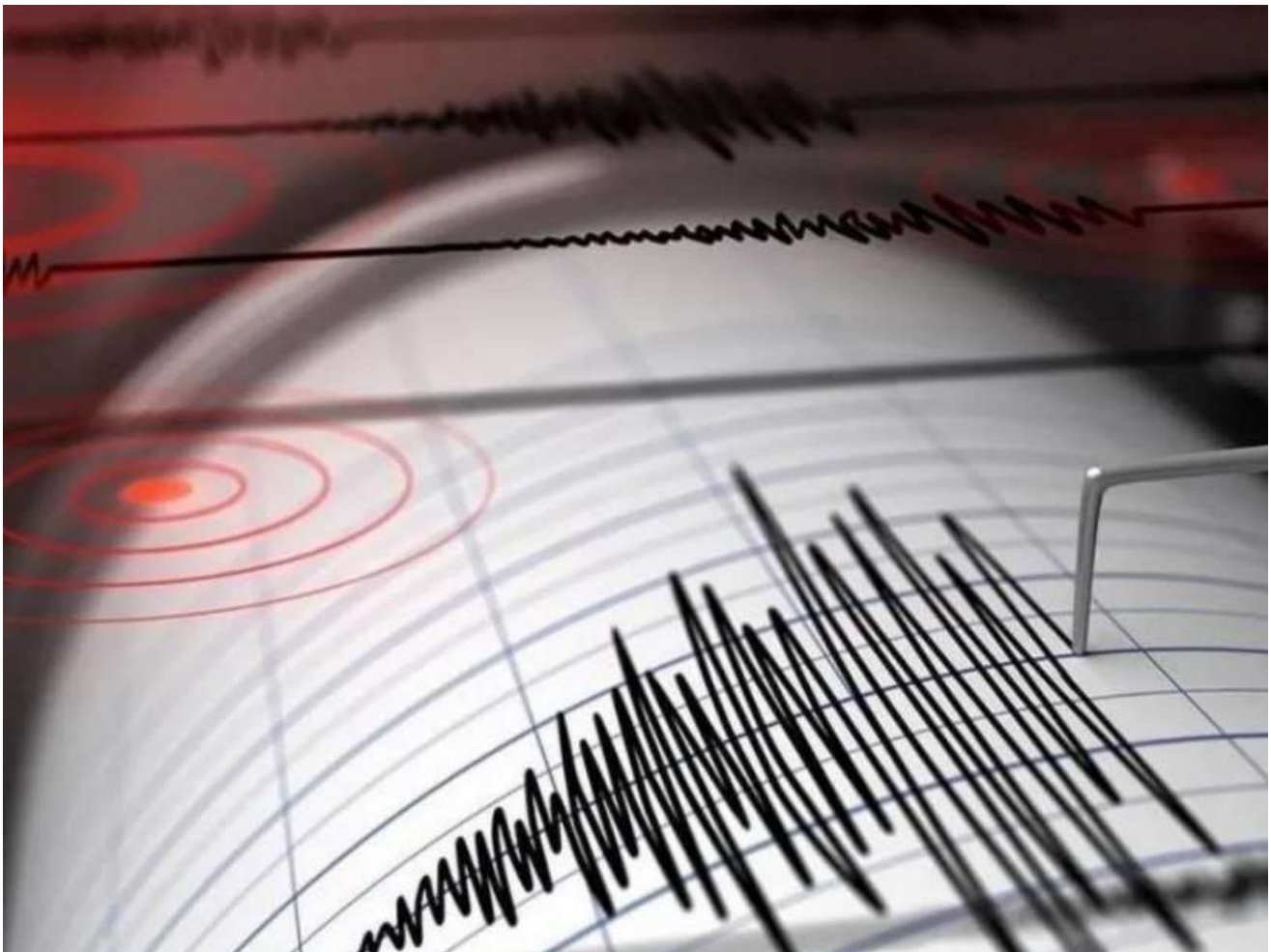
- Depending on your project's scope, you might require tools to access external data sources, such as APIs or data scraping tools.

## **13.Data Annotation and Labeling Tools (if applicable):**

- For specialized projects, tools for data annotation and labeling may be necessary, such as Labelbox or Supervisely.

## **14.Geospatial Tools (for location-based features):**

- If your dataset includes geospatial data, geospatial libraries like GeoPandas can be helpful.



# **1.DESIGN THINKING AND PRESENT IN FORM**

## **OF DOCUMENT**

### **1.Empathize:**

- Understand the needs and challenges of all stakeholders involved in the earthquake prediction process
- Conduct interviews and surveys to gather insights on what users value in property valuation and what information is most critical for their decision-making.

### **2.Define:**

- Clearly articulate the problem statement, such as "How might we predict earthquake more accurately and transparently using machine learning?"
- Identify the key goals and success criteria for the project, such as increasing prediction accuracy, reducing bias, or improving user trust in the valuation process.

### **3.Ideate:**

- Brainstorm creative solutions and data sources that can enhance the accuracy and transparency of earthquake predictions.
- Encourage interdisciplinary collaboration to generate a wide range of ideas, including the use of alternative data, new algorithms, or improved visualization techniques.

### **4.Prototype:**

- Create prototype machine learning models based on the ideas generated during the ideation phase.
- Test and iterate on these prototypes to determine which approaches are most promising in terms of accuracy and usability.

### **5.Test:**

- Gather feedback from users and stakeholders by testing the machine learning models with real-world data and scenarios.
- Assess how well the models meet the defined goals and success criteria, and make adjustments based on user feedback.

## **6.Implement:**

- Develop a production-ready machine learning solution for predicting earthquake, integrating the best-performing algorithms and data sources.
- Implement transparency measures, such as model interpretability tools, to ensure users understand how predictions are generated.

## **7.Evaluate:**

- Continuously monitor the performance of the machine learning model after implementation to ensure it remains accurate.
- Gather feedback and insights from users.

## **8.Iterate:**

- Apply an iterative approach to refine the machine learning model based on ongoing feedback and changing user needs.
- Continuously seek ways to enhance prediction accuracy, transparency, and user satisfaction.

## **9.Scale and Deploy:**

- Once the machine learning model has been optimized and validated, deploy it .
- Ensure the model is accessible through user-friendly interfaces and integrates seamlessly into real estate workflows.

## **10.Educate and Train:**

- Provide training and educational resources to help users understand how the machine learning model works, what factors it considers, and its limitations.
- Foster a culture of data literacy among stakeholders to enhance trust in the technology.

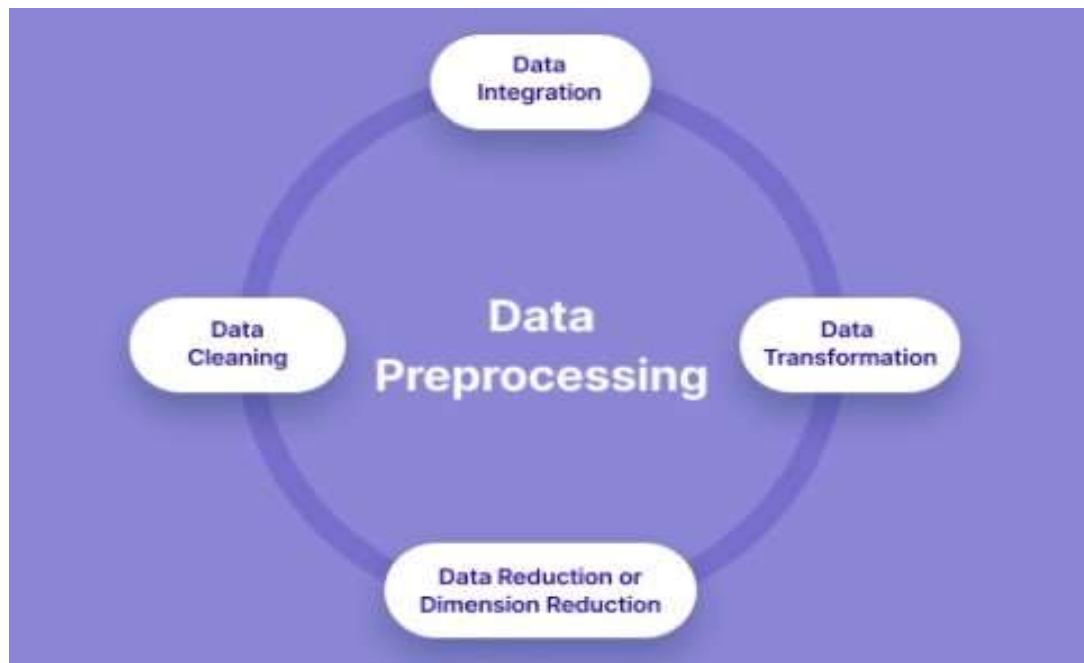
## **2.DESIGN INTO INNOVATION**

### **1. Data Collection:**

Gather a comprehensive dataset that includes features such as latitude, longitude, magnitude, depth, others relevant variables.

### **2.Data Preprocessing:**

Clean the data by handling missing values, outliers, and encoding categorical variables. Standardize or normalize numerical features as necessary.





## **PYTHON PROGRAM:**

*# Import necessary libraries*

```
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.impute import SimpleImputer
```

```
from sklearn.preprocessing import StandardScaler
```

*# Load the dataset (replace 'earthquake.csv' with your dataset file)*

```
df = pd.read_csv('C:/Users/barat/Downloads/archive/database.csv')
```

*# Display the first few rows of the dataset to get an overview*

```
print("Dataset Preview:")
```

```
print(data.head())
```

**# Data Pre-processing**

*# Handle Missing Values*

*# Let's fill missing values in numeric columns with the mean and in categorical columns with the most frequent value.*

```
numeric_cols = data.select_dtypes(include='number').columns categorical_cols =
```

```
data.select_dtypes(exclude='number').columns
```

```
imputer_numeric = SimpleImputer(strategy='mean') imputer_categorical =
```

```
SimpleImputer(strategy='most_frequent')
```

```
data[numeric_cols] = imputer_numeric.fit_transform(data[numeric_cols])
```

```
data[categorical_cols] =
```

```
imputer_categorical.fit_transform(data[categorical_cols])
```

*# Convert Categorical Features to Numerical*

*# We'll use Label Encoding for simplicity here. You can also use one-hot encoding for nominal categorical features.*

```
label_encoder = LabelEncoder() for
```

```
col in categorical_cols:
```

```
    data[col] = label_encoder.fit_transform(data[col])
```

*# Split Data into Features(X)&Target(y)*

```
X = data.drop(columns=[Magnitude])
```

```
y = data['Magnitude']
```

*# Normalize the Data*

```
scaler = StandardScaler() X_scaled =
```

```
scaler.fit_transform(X)
```

*# Split data into training and testing sets*

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

*# Display the preprocessed data*

```
print("\nPreprocessed Data:")
```

```
print(X_train[:5])
```

```
print(y_train[:5])
```

### **3.Feature Engineering:**

Create new features or transform existing ones to extract more valuable information.

#### **4.Model Selection:**

Choose the appropriate machine learning model for the task. Common models for regression problems like house price prediction include *Linear Regression, Decision Trees, Random Forest, Gradient Boosting, and Neural Networks*.

#### **5.Training:**

Split the dataset into training and testing sets to evaluate the model's performance. Consider techniques like cross-validation to prevent overfitting.

#### **6.Hyperparameter Tuning:**

Optimize the model's hyperparameters to improve its predictive accuracy. Techniques like grid search or random search can help with this.

#### **7.Evaluation Metrics:**

Select appropriate evaluation metrics for regression tasks, such as *Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE)*. Choose the metric that aligns with the specific objectives of your project.

#### **8.Regularization:**

Apply regularization techniques like L1 (Lasso) or L2 (Ridge) regularization to prevent overfitting.

#### **9.Feature Selection:**

Use techniques like feature importance scores or recursive feature elimination to identify the most relevant features for the prediction.

#### **10.Interpretability:**

Ensure that the model's predictions are interpretable and explainable. This is especially important for earthquake applications where stakeholders want to understand the factors affecting predictions.

#### **11.Deployment:**

Develop a user-friendly interface or API for end-users to input property details and receive price predictions.

## **12.Continuous Improvement:**

Implement a feedback loop for continuous model improvement based on user feedback and new data.

## **13.Ethical Considerations:**

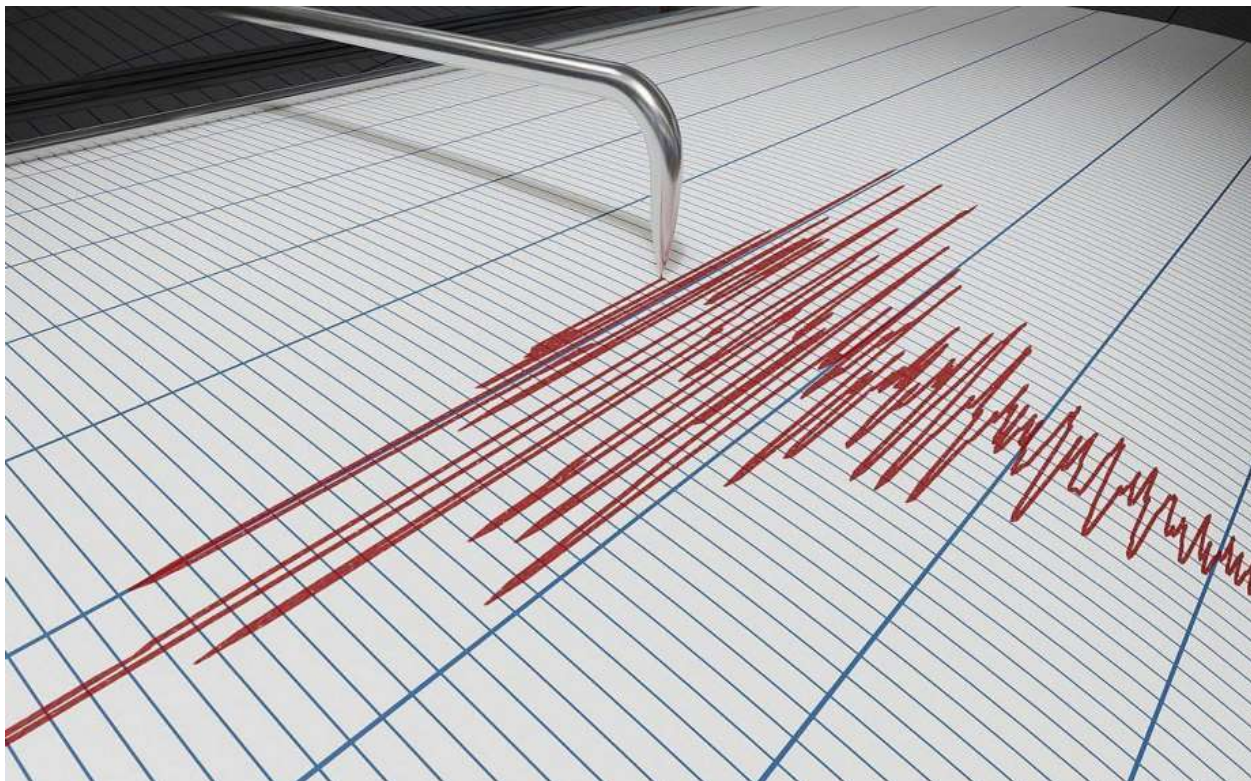
Be mindful of potential biases in the data and model. Ensure fairness and transparency in your predictions.

## **14.Monitoring and Maintenance:**

Regularly monitor the model's performance in the real world and update it as needed.

## **15.Innovation:**

Consider innovative approaches such as using satellite imagery or IoT data for real-time property condition monitoring.



### **3.BUILD LOADING AND PREPROCESSING THE DATASET**

#### **1. Data Collection:**

Obtain a dataset that contains information about earthquake and their corresponding prices. This dataset can be obtained from sources like real estate websites, government records, or other reliable data providers.

#### **2. Load the Dataset:**

- Import relevant libraries, such as pandas for data manipulation and numpy for numerical operations.
- Load the dataset into a pandas DataFrame for easy data handling.
- You can use *pd.read\_csv()* for CSV files or other appropriate functions for different file formats.

#### **Program:**

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2_score,
mean_absolute_error, mean_squared_error

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
```

### Loading Dataset:

```
df=pd.read_csv("C:/Users/barat/Downloads/archive/databa  
se.csv")
```

### Output:

|       | Date       | Time     | Latitude | Longitude | Type       | Depth  | Depth<br>Error | Depth<br>Seismic<br>Stations | Magnitude | Magnitude<br>Type | ... | Magnitude<br>Seismic<br>Stations | Azimuthal<br>Gap | Horizontal<br>Distance | Horizontal<br>Error |
|-------|------------|----------|----------|-----------|------------|--------|----------------|------------------------------|-----------|-------------------|-----|----------------------------------|------------------|------------------------|---------------------|
| 0     | 01/02/1965 | 13:44:18 | 19.2460  | 145.6160  | Earthquake | 131.60 | NaN            | NaN                          | 6.0       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| 1     | 01/04/1965 | 11:29:49 | 1.8630   | 127.3520  | Earthquake | 80.00  | NaN            | NaN                          | 5.8       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| 2     | 01/05/1965 | 18:05:58 | -20.5790 | -173.9720 | Earthquake | 20.00  | NaN            | NaN                          | 6.2       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| 3     | 01/08/1965 | 18:49:43 | -59.0760 | -23.5570  | Earthquake | 15.00  | NaN            | NaN                          | 5.8       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| 4     | 01/09/1965 | 13:32:50 | 11.9380  | 126.4270  | Earthquake | 15.00  | NaN            | NaN                          | 5.8       | MW                | ... | NaN                              | NaN              | NaN                    | NaN                 |
| ...   | ...        | ...      | ...      | ...       | ...        | ...    | ...            | ...                          | ...       | ...               | ... | ...                              | ...              | ...                    | ...                 |
| 23407 | 12/28/2016 | 08:22:12 | 38.3917  | -118.8941 | Earthquake | 12.30  | 1.2            | 40.0                         | 5.6       | ML                | ... | 18.0                             | 42.47            | 0.120                  | NaN                 |
| 23408 | 12/28/2016 | 09:13:47 | 38.3777  | -118.8957 | Earthquake | 8.80   | 2.0            | 33.0                         | 5.5       | ML                | ... | 18.0                             | 48.58            | 0.129                  | NaN                 |
| 23409 | 12/28/2016 | 12:38:51 | 36.9179  | 140.4262  | Earthquake | 10.00  | 1.8            | NaN                          | 5.9       | MWW               | ... | NaN                              | 91.00            | 0.992                  | 4.8                 |
| 23410 | 12/29/2016 | 22:30:19 | -9.0283  | 118.6639  | Earthquake | 79.00  | 1.8            | NaN                          | 6.3       | MWW               | ... | NaN                              | 26.00            | 3.553                  | 6.0                 |
| 23411 | 12/30/2016 | 20:08:28 | 37.3973  | 141.4103  | Earthquake | 11.94  | 2.2            | NaN                          | 5.5       | MB                | ... | 428.0                            | 97.00            | 0.681                  | 4.5                 |

23412 rows x 21 columns

### **3.Data Exploration:**

Explore the dataset to understand its structure and contents. Check for the presence of missing values, outliers, and data types of each feature.

### **4.Data Cleaning:**

Handle missing values by either removing rows with missing data or imputing values based on the nature of the data.

### **5.Feature Selection:**

Identify relevant features for earthquake prediction. Features like the latitude, longitude, magnitude, and depth are often important.

## *#Checking for missing values*

**In[1]:**

```
print("Missing values")
print("-" *30)
print(df.isna().sum())
print("-"*30)
print("Total missing values", df.isna().sum().sum())
```

**Out[1]:**

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
```

-----  
Total missing values 145439

## **6.Feature Engineering:**

Create new features or transform existing ones to capture additional information that may impact earthquake.

## **7.Data Encoding:**

Convert categorical variables into numerical format using techniques like one-hot encoding.

## **8.Train-Test Split:**

Split the dataset into training and testing sets to evaluate the

machine learning model's performance.

**Program:**

```
X = df[['Latitude', 'Longitude']]
```

```
y = df[['Depth', 'Magnitude']]
```

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

## **4. PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION etc.,**

### **1. Feature Engineering:**

- As mentioned earlier, feature engineering is crucial. It involves creating new features or transforming existing ones to provide meaningful information for your model.
- Extracting information from textual descriptions.

### **2.Data Preprocessing & Visualisation:**

Continue data preprocessing by handling any remaining missing values or outliers based on insights from your data exploration.

### **Visualisation and Pre-Processing of Data:**

**In[1]:**

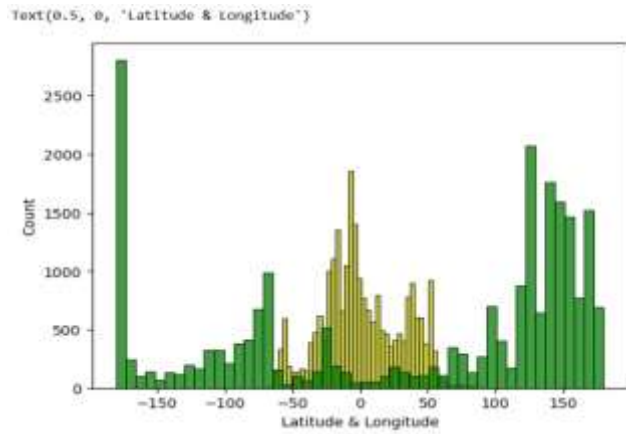
```
sns.histplot(df,x = 'Latitude', bins=50, color='y')
```

```
sns.histplot(df,x = 'Longitude', bins=50, color='g')
```

```
plt.xlabel('Latitude & Longitude')
```

**Out[1]:**



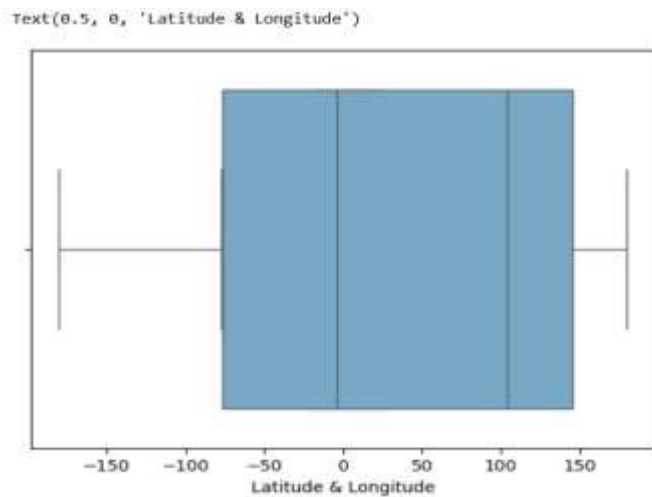


**In[2]:**

```
sns.boxplot(df,x='Latitude',palette='Blues')
```

```
sns.boxplot(df,x='Longitude',palette='Blues') plt.xlabel('Latitude & Longitude')
```

**Out[2]:**

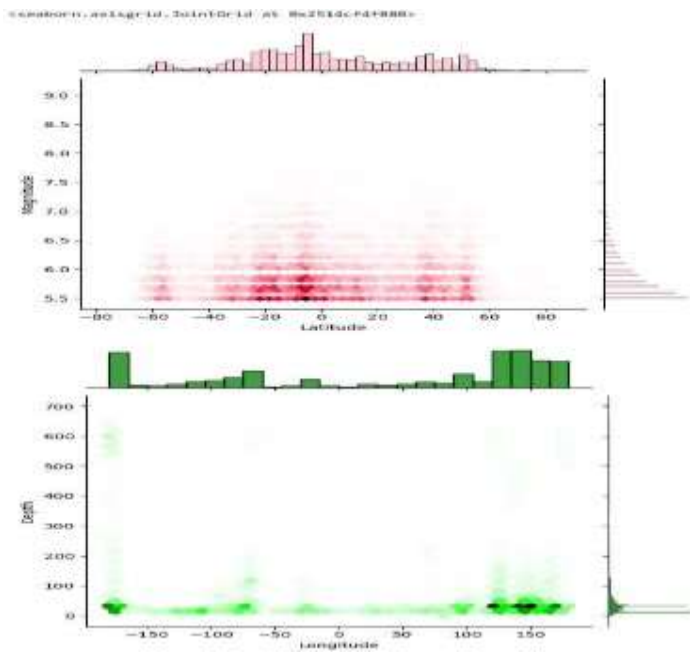


**In[3]:**

```
sns.jointplot(df,x='Latitude', y='Magnitude', kind='hex', color='pink')
```

```
sns.jointplot(df,x='Longitude', y='Depth', kind='hex', color='green')
```

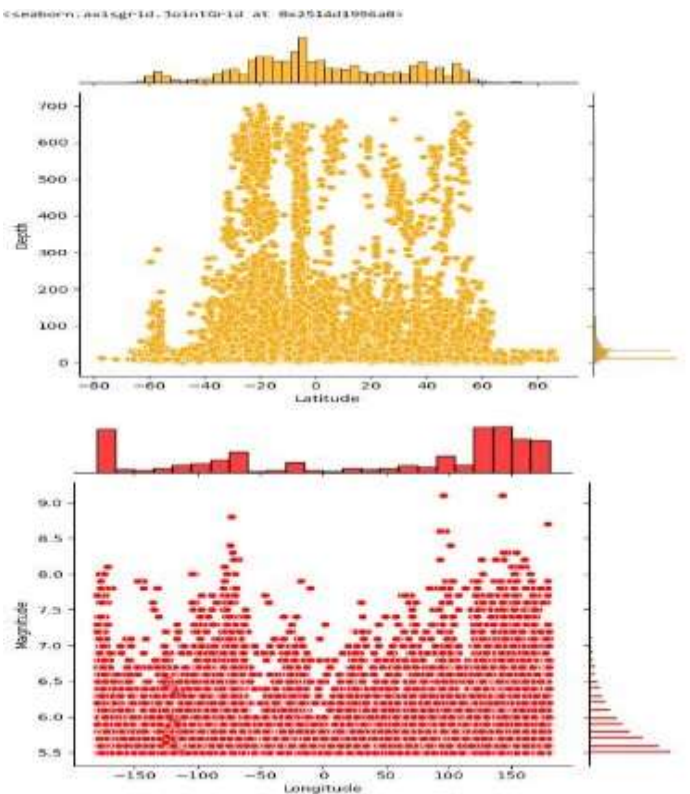
**Out[3]:**



**In[4]:**

```
sns.jointplot(df,x='Latitude', y='Depth', color='orange')
sns.jointplot(df,x='Longitude', y='Magnitude', color='red')
```

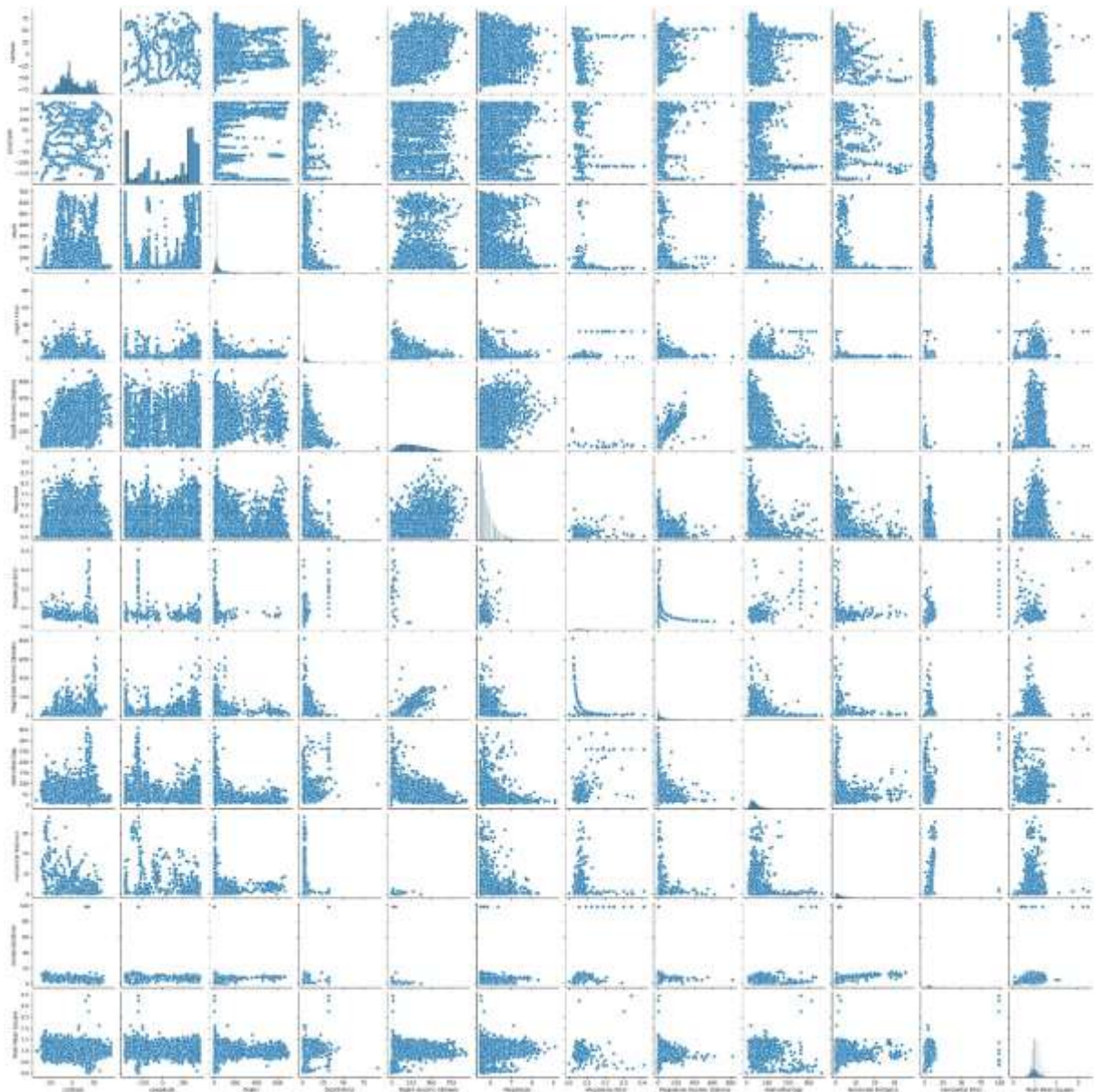
**Out[4]:**



**In[5]:**

```
plt.figure(figsize=(6,8))
sns.pairplot(df) plt.show()
```

**Out[5]:**

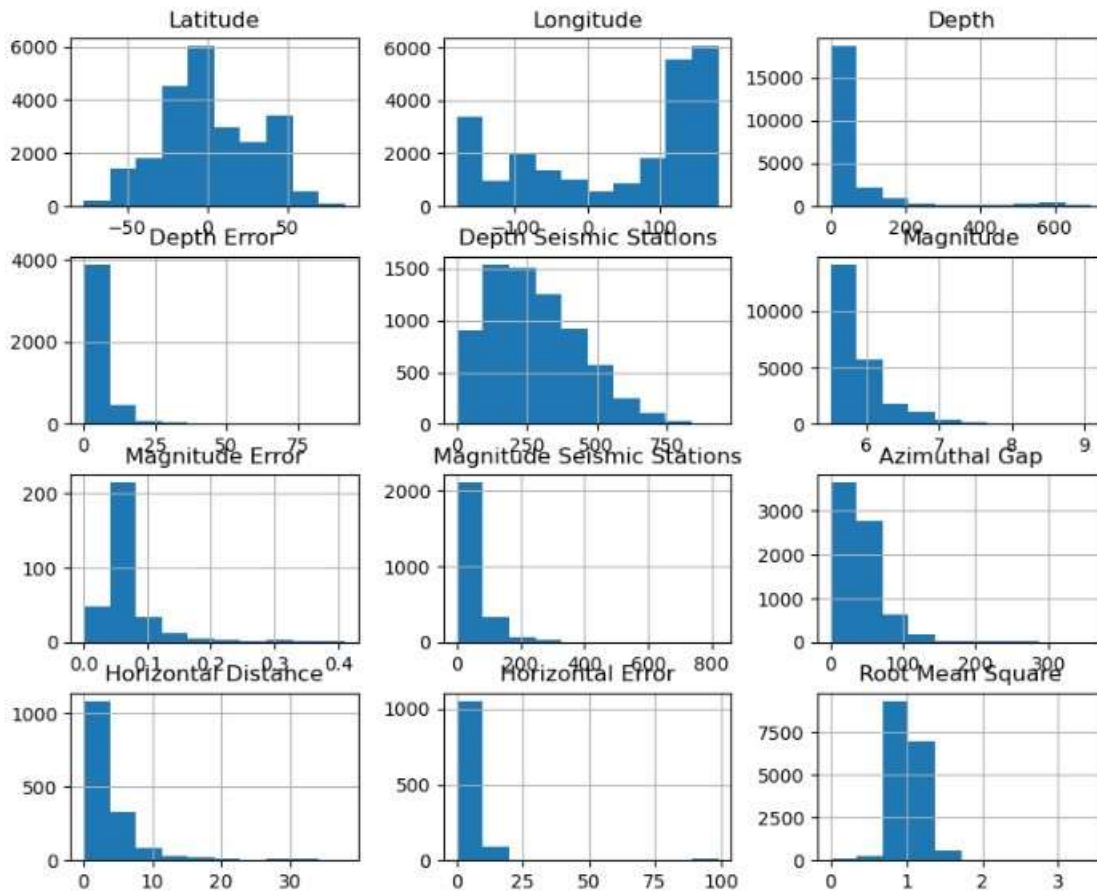


In[6]:

```
df.hist(figsize=(10,8))
```

Out[6]:

```
array([[<AxesSubplot: title={'center': 'Latitude'}>,  
       <AxesSubplot: title={'center': 'Longitude'}>,  
       <AxesSubplot: title={'center': 'Depth'}>],  
      [[<AxesSubplot: title={'center': 'Depth Error'}>,  
       <AxesSubplot: title={'center': 'Depth Seismic Stations'}>,  
       <AxesSubplot: title={'center': 'Magnitude'}>],  
      [[<AxesSubplot: title={'center': 'Magnitude Error'}>,  
       <AxesSubplot: title={'center': 'Magnitude Seismic Stations'}>,  
       <AxesSubplot: title={'center': 'Azimuthal Gap'}>],  
      [[<AxesSubplot: title={'center': 'Horizontal Distance'}>,  
       <AxesSubplot: title={'center': 'Horizontal Error'}>,  
       <AxesSubplot: title={'center': 'Root Mean Square'}>]],  
      dtype=object)
```



In[7]:

```
df.corr(numeric_only= True)
```

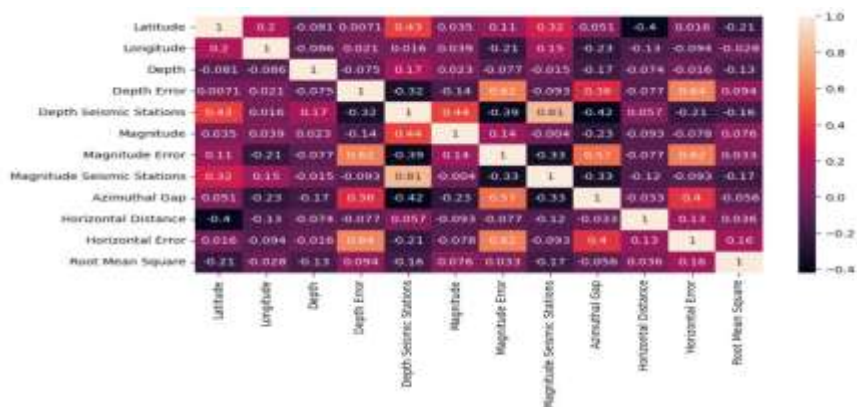
Out[7]:

|                            | Latitude  | Longitude | Depth     | Depth Error | Depth Seismic Stations | Magnitude | Magnitude Error | Magnitude Seismic Stations | Azimuthal Gap | Horizontal Distance | Horizontal Error | Root Mean Square |
|----------------------------|-----------|-----------|-----------|-------------|------------------------|-----------|-----------------|----------------------------|---------------|---------------------|------------------|------------------|
| Latitude                   | 1.000000  | 0.203546  | -0.081020 | 0.007080    | 0.433815               | 0.034987  | 0.113208        | 0.315075                   | 0.050794      | -0.396768           | 0.015625         | -0.214782        |
| Longitude                  | 0.203546  | 1.000000  | -0.085861 | 0.020552    | 0.015824               | 0.038579  | -0.214609       | 0.148510                   | -0.233097     | -0.131313           | -0.093827        | -0.028061        |
| Depth                      | -0.081020 | -0.085861 | 1.000000  | -0.074809   | 0.174883               | 0.023457  | -0.078918       | -0.015254                  | -0.171162     | -0.073832           | -0.016467        | -0.134002        |
| Depth Error                | 0.007080  | 0.020552  | -0.074809 | 1.000000    | -0.320579              | -0.135880 | 0.618254        | -0.093292                  | 0.357704      | -0.077423           | 0.644593         | 0.094398         |
| Depth Seismic Stations     | 0.433815  | 0.015824  | 0.174883  | -0.320579   | 1.000000               | 0.440582  | -0.385993       | 0.813374                   | -0.420556     | 0.056619            | -0.214959        | -0.158620        |
| Magnitude                  | 0.034987  | 0.038579  | 0.023457  | -0.135880   | 0.440582               | 1.000000  | 0.135573        | -0.003972                  | -0.233579     | -0.092609           | -0.078406        | 0.075885         |
| Magnitude Error            | 0.113208  | -0.214609 | -0.078918 | 0.618254    | -0.385993              | 0.135573  | 1.000000        | -0.334062                  | 0.567411      | -0.076744           | 0.617721         | 0.032616         |
| Magnitude Seismic Stations | 0.315075  | 0.148510  | -0.015254 | -0.093292   | 0.813374               | -0.003972 | -0.334062       | 1.000000                   | -0.334864     | -0.117606           | -0.093143        | -0.167473        |
| Azimuthal Gap              | 0.050794  | -0.233097 | -0.171162 | 0.357704    | -0.420556              | -0.233579 | 0.567411        | -0.334864                  | 1.000000      | -0.033482           | 0.398450         | -0.058217        |
| Horizontal Distance        | -0.396768 | -0.131313 | -0.073832 | -0.077423   | 0.056619               | -0.092609 | -0.076744       | -0.117606                  | -0.033482     | 1.000000            | 0.126877         | 0.035778         |
| Horizontal Error           | 0.015625  | -0.093827 | -0.016467 | 0.644593    | -0.214959              | -0.078406 | 0.617721        | -0.093143                  | 0.398450      | 0.126877            | 1.000000         | 0.157842         |
| Root Mean Square           | -0.214782 | -0.028061 | -0.134002 | 0.094398    | -0.158620              | 0.075885  | 0.032616        | -0.167473                  | -0.058217     | 0.035778            | 0.157842         | 1.000000         |

In[8]:

```
plt.figure(figsize=(10,5)) sns.heatmap(df.corr(numeric_only=
True),annot = True) plt.show()
```

Out[8]:





### 3.Model Selection:

Choose an appropriate machine learning model for your regression task. *Common choices include:*

- Linear Regression
- Decision Trees
- Random Forest

#### **Program:**

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2_score,
mean_absolute_error, mean_squared_error

from sklearn.linear_model import ElasticNet
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso

from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
```

#### **Loading Dataset:**

```
df=pd.read_csv("C:/Users/barat/Downloads/archive/databa
se.csv")
```

## **Model 1 – Linear Regression**

**In[1]:**

```
new_row = {"Model": "Ridge", "MAE":mae, "MSE": mse,"RMSE":rmse,  
           "R2 Score": r_squared, "RMSE(Cross-Validation)":rmse_cross_val}  
models = models.append(new_row, ignore_index=True)
```

**In[2]:**

```
def evaluation(y_true, y_pred):  
  
    mae = mean_absolute_error(y_true, y_pred)  
  
    mse = mean_squared_error(y_true, y_pred)  
  
    rmse = np.sqrt(mse) rmse_cross_val = np.mean(rmse)  
  
    r_squared = r2_score(y_true, y_pred)  
  
    return mae, mse, rmse, r_squared, rmse_cross_val
```

**In[3]:**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  
lin_reg = LinearRegression()  
  
lin_reg.fit(X_train, y_train)  
  
predictions = lin_reg.predict(X_test)  
  
mae, mse, rmse, r_squared,rmse_cross_val = evaluation(y_test, predictions)  
print("MAE:",mae)  
  
print("MSE:",mse)  
  
print("RMSE:",rmse)  
  
print("R2 Score:",r_squared)
```

```
print("-" *30)
```

```
print("RMSE Cross-Validation:",rmse_cross_val)
```

**Out[3]:**

```
MAE: 16.214208564591
MSE: 413.6507308565237
RMSE: 20.33840531744128
R2 Score: -0.15997292842810484
-----
RMSE Cross-Validation: 20.33840531744128
```

### **Evaluation of Predicted Data :**

**In[4]:**

```
plt.figure(figsize=(12,6))

plt.plot(np.arange(len(y_test)), y_test)

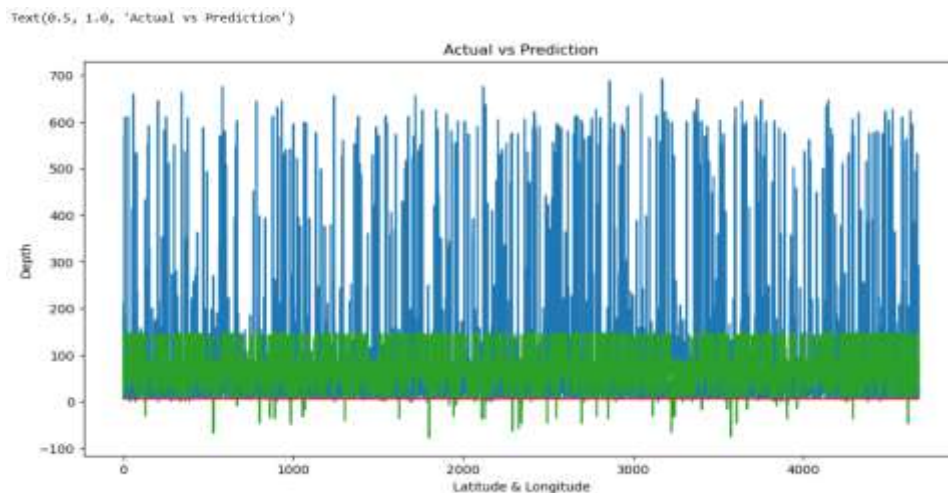
plt.plot(np.arange(len(y_test)), predictions)

plt.xlabel("Latitude & Longitude")

plt.ylabel("Depth")

plt.title("Actual vs Prediction")
```

**Out[4]:**





## **Model 2 – Lasso Regression**

**In[1]:**

```
lasso = Lasso()

lasso.fit(X_train, y_train)

predictions = lasso.predict(X_test)

mae,mse, rmse, r_squared,rmse_cross_val = evaluation(y_test, predictions)

print("MAE:",mae)

print("MSE:",mse)

print("RMSE:",rmse)

print("R2 Score:",r_squared)

print("-" *30)

print("RMSE Cross-Validation:",rmse_cross_val)
```

**Out[1]:**

```
MAE: 10.864336073458082
MSE: 195.0097579097878
RMSE: 13.96458942861507
R2 Score: 0.4531472494046366
-----
RMSE Cross-Validation: 13.96458942861507
```

### **Model 3 – Elastic Net**

**In[1]:**

```
elasticnet = ElasticNet()

elasticnet.fit(X_train, y_train)

predictions = elasticnet.predict(X_test)

mae,mse, rmse, r_squared,rmse_cross_val = evaluation(y_test, predictions)

print("MAE:",mae)

print("MSE:",mse)

print("RMSE:",rmse)

print("R2 Score:",r_squared)

print("-" *30)

print("RMSE Cross-Validation:",rmse_cross_val)
```

**Out[1]:**

```
MAE: 10.872423700794576
MSE: 195.23917220459506
RMSE: 13.972801158128425
R2 Score: 0.4525039183247668
-----
RMSE Cross-Validation: 13.972801158128425
```

## **Model 4 – SVR**

**In[1]:**

```
svr = SVR(C=100000)

svr.fit(X_train,y_train)

predictions = svr.predict(X_test)

mae,mse, rmse, r_squared,rmse_cross_val = evaluation(y_test, predictions)

print("MAE:",mae)

print("MSE:",mse)

print("RMSE:",rmse)

print("R2 Score:",r_squared)

print("-" *30)

print("RMSE Cross-Validation:",rmse_cross_val)
```

**Out[1]:**

```
MAE: 60.364276908953464
MSE: 3877.4242177347583
RMSE: 62.26896673090664
R2 Score: -9.873199994813712
-----
RMSE Cross-Validation: 62.26896673090664
```

## **Model 5 – Random Forest Regressor**

**In[1]:**

```
random_forest = RandomForestRegressor(n_estimators= 100)

random_forest.fit(X_train, y_train)

predictions = random_forest.predict(X_test)

mae,mse, rmse, r_squared,rmse_cross_val = evaluation(y_test, predictions)

print("MAE:",mae)

print("MSE:",mse)

print("RMSE:",rmse)

print("R2 Score:",r_squared)

print("-" *30)

print("RMSE Cross-Validation:",rmse_cross_val)
```

**Out[1]:**

```
MAE: 10.295796132468222
MSE: 198.72930732017593
RMSE: 14.097138267044697
R2 Score: 0.44271676711570895
-----
RMSE Cross-Validation: 14.097138267044697
```

## **Model 6 – Polynomial Regression(Degree = 2)**

**In[1]:**

```
poly_reg = PolynomialFeatures(degree =2)

X_train_2d = poly_reg.fit_transform(X_train)

X_test_2d = poly_reg.transform(X_test)

lin_reg = LinearRegression()

lin_reg.fit(X_train_2d, y_train)

predictions = lin_reg.predict(X_test_2d)

mae,mse, rmse, r_squared,rmse_cross_val = evaluation(y_test, predictions)

print("MAE:",mae)

print("MSE:",mse)

print("RMSE:",rmse)

print("R2 Score:",r_squared)

print("-" *30)

print("RMSE Cross-Validation:",rmse_cross_val)
```

**Out[1]:**

```
MAE: 39.11674027433722
MSE: 1563.7117106065875
RMSE: 39.5437948432695
R2 Score: -3.3850115976195143
-----
RMSE Cross-Validation: 39.5437948432695
```

### **Advantages:**

- It can help people prepare for and mitigate the potential damages and losses caused by earthquakes. For example, people can evacuate areas, disconnect gas, electricity and water, organize and prepare evacuation centers, alert relatives and authorities, and secure their belongings.
- It can help scientists and engineers better understand the causes, mechanisms, patterns, and effects of earthquakes. For example, by monitoring seismic activity and precursors, they can develop models and algorithms for earthquake prediction and forecasting, identify locations within urban regions that are especially vulnerable to damaging earthquake ground motions (seismic zonation), and improve the design and construction of earthquake-resistant structures.
- It can help government agencies and authorities develop and implement policies and regulations related to earthquake risk management, disaster preparedness, response, and recovery. For example, by providing reliable information and warnings about earthquake hazards and advisories, they can allocate resources and support for earthquake research and monitoring programs, coordinate emergency services and relief efforts, and facilitate economic development and recovery of the affected areas.

### **Disadvantages:**

- It is not a reliable or accurate method, as there is no proven way to predict earthquakes with high precision and certainty. Many possible earthquake precursors have been proposed, but none of them have been consistently and reliably identified across different regions and scales. There is also a risk of

false alarms or missed predictions that can cause panic or complacency among the public.

- It is not a cheap or easy method, as it requires a lot of resources and expertise to conduct extensive research and monitoring of seismic activity and precursors. It also involves a lot of ethical and legal issues related to the responsibility and accountability of the predictors and the users of the predictions.
- It is not a sufficient or effective method, as it does not guarantee that people will act appropriately or rationally in response to the predictions. There may be barriers or constraints that prevent people from taking preventive or protective measures, such as lack of awareness, education, trust, access, or resources. There may also be unintended or adverse consequences that result from the predictions, such as social disruption, economic loss, or environmental damage.

## **Conclusion:**

- Earthquake prediction is a challenging and important task that aims to forecast the occurrence, location, magnitude, and impact of future earthquakes based on various types of data and models.
- Earthquake prediction can help reduce the loss of life and property, improve the preparedness and resilience of communities, and advance the scientific understanding of the earth's processes.
- However, earthquake prediction is also subject to many uncertainties, limitations, and ethical issues that need to be addressed.

- Earthquake data is often noisy, incomplete, inconsistent, or unreliable. For example, seismic waveforms may be affected by environmental factors, instrument errors, or human interference.
- Earthquake catalog may be biased, incomplete, or inaccurate due to different reporting standards, detection thresholds, or measurement methods.
- Geological features may be difficult to measure or estimate due to the complexity and variability of the earth's structure and dynamics.  
  
Environmental factors may be irrelevant, redundant, or misleading as potential precursors or indicators of seismic activity.
- Earthquake models are often based on simplifying assumptions, approximations, or empirical rules that may not capture the true physics or statistics of the earthquake phenomenon.
- Earthquake prediction is inherently probabilistic and uncertain due to the randomness and complexity of the earthquake process.
- Earthquake prediction is not a perfect science but a continuous learning process that requires collaboration, innovation, and evaluation.
- By improving the data quality and availability, developing more realistic and robust models, enhancing the prediction accuracy and uncertainty quantification, and considering the ethical and social implications, earthquake prediction can become more feasible and beneficial for society.



