EARTHQUAKE PREDICTION MODEL USING PYTHON

Phase 3 Submission Document:

Project Title: Earthquake Prediction

Phase 3: Development Part 1

INTRODUCTION:

- ✓ Earthquake Prediction is a way of predicting the magnitude of an earthquake based on parameters such as longitude, latitude, depth, and duration magnitude, country, and depth using machine learning to give warnings of potentially damaging earthquakes early enough to allow appropriate response to the disaster, enabling people to minimize loss of life and property.

1.Feature Engineering:

1. Extracting meaningful features from the data, such as temporal trends, spatial relationships, and interactions between different parameters.
2. Incorporating domain-specific knowledge and external factors that may influence seismic activities.

2.Machine Learning Models:

1. Developing predictive models, often based on machine learning algorithms, to analyze and learn patterns from historical earthquake data.
2. Common models include neural networks, support vector machines, and decision trees.

3.Hyperparameter Tuning:

1. Fine-tuning model parameters to optimize predictive performance through techniques like grid search or Bayesian optimization.

4.Validation and Evaluation:

1. Splitting the dataset into training and testing sets to validate the model's performance.
2. Evaluating the model's accuracy, precision, recall, and other relevant metrics.

5.Challenges in Earthquake Prediction:

1. Earthquakes are inherently unpredictable due to the dynamic and complex nature of tectonic processes.
2. Limited historical data for rare, large-magnitude earthquakes makes it challenging to train accurate models.

Developmet Part 1:

- Loading
- Preprocessing

Loading DataSet:

Data loading in earthquake prediction is a crucial step in the process. It involves gathering and organizing relevant information to train and test predictive models

Downloading DataSet:

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

Importing Libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns from sklearn.preprocessing
import StandardScaler from sklearn.model_selection
import train_test_split
import tensorflow as tf
```

Load the Dataset:

Load your dataset into a Pandas DataFrame. You can typically predict the earthquake datasets in CSV format, but you can adapt this code to other formats as needed.

```
data = pd.read_csv('../input/earthquake-database/database.csv')
```

```
pd.data
```

	Date	Time	Latitude	Longitude	Type
0	01/02/1965	13:44:18	19.2460	145.6160	Earthquake
1	01/04/1965	11:29:49	1.8630	127.3520	Earthquake
2	01/05/1965	18:05:58	-20.5790	-173.9720	Earthquake
3	01/08/1965	18:49:43	-59.0760	-23.5570	Earthquake
4	01/09/1965	13:32:50	11.9380	126.4270	Earthquake
...	...	...	...	...	...
23407	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake
23408	12/28/2016	09:13:47	38.3777	-118.8957	Earthquake
23409	12/28/2016	12:38:51	36.9179	140.4262	Earthquake
23410	12/29/2016	22:30:19	-9.0283	118.6639	Earthquake
23411	12/30/2016	20:08:28	37.3973	141.4103	Earthquake

```
pd.data.info()
```

```
<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
```

Selecting Columns:

To select specific columns from the dataset, you can use the column names. For example, if you want to select the "latitude" and "longitude" columns:

```
selected_columns = earthquake_data[['latitude', 'longitude']]
```

Filtering Rows Based on Conditions:

If you want to select rows based on certain conditions, you can use boolean indexing.

```
high_magnitude_earthquakes = earthquake_data[earthquake_data['magnitude'] > 5.0]
```

Selecting Specific Rows and Columns:

If you want to select specific rows and columns simultaneously, you can combine the two operations.

```
specific_data = earthquake_data.loc[earthquake_data['magnitude'] > 5.0, ['latitude', 'longitude']]
```

Preprocessing Data:

Preprocessing is a crucial step in earthquake prediction as it helps clean and transform raw data into a format suitable for training machine learning models.

Handling Missing Data:

Check for missing values in your dataset and decide on a strategy to handle them. You can either remove rows with missing values, fill them using imputation techniques, or use more advanced methods depending on the nature of the missing data.

```
data= data.drop('ID', axis=1)
```

```
data.isna().sum()
```

```
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
Source                            0
Location Source                   0
Magnitude Source                  0
Status                            0
dtype: int64
```

```
Data= data.drop(null_columns, axis=1)
```

```
data.isna().sum()
```

```
Date                        0
Time                        0
Latitude                    0
Longitude                   0
Type                        0
Depth                       0
Magnitude                   0
Magnitude Type              3
Root Mean Square         6060
Source                      0
Location Source             0
Magnitude Source            0
Status                      0
dtype: int64
```

Feature Engineering:

Create new features that might capture important information. For example, you could calculate the distance from each earthquake to a known fault line or landmark.

```
eq_data['distance_to_fault'] = calculate_distance(eq_data['latitude'],
eq_data['longitude'])
```

```
data
```

	Date	Time	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type
0	01/02/1965	13:44:18	19.2460	145.6160	Earthquake	131.60	6.0	MW
1	01/04/1965	11:29:49	1.8630	127.3520	Earthquake	80.00	5.8	MW
2	01/05/1965	18:05:58	-20.5790	-173.9720	Earthquake	20.00	6.2	MW
3	01/08/1965	18:49:43	-59.0760	-23.5570	Earthquake	15.00	5.8	MW
4	01/09/1965	13:32:50	11.9380	126.4270	Earthquake	15.00	5.8	MW
...	...	...	...	...	...	...	...	...
23404	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake	12.30	5.6	ML
23405	12/28/2016	09:13:47	38.3777	-118.8957	Earthquake	8.80	5.5	ML
23406	12/28/2016	12:38:51	36.9179	140.4262	Earthquake	10.00	5.9	MWW
23407	12/29/2016	22:30:19	-9.0283	118.6639	Earthquake	79.00	6.3	MWW
23408	12/30/2016	20:08:28	37.3973	141.4103	Earthquake	11.94	5.5	MB

```
data['Month']= data['Date'].apply(lambda x: x[0:2]) data['Year'] =

data['Date'].apply(lambda x: x[-4:])

data = data.drop('Date', axis=1)
```

```
data['Month'] = data['Month'].astype(np.int)
```

```
data[data['Year'].str.contains('Z')]
```

	Time	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type	Root Mean Square
3378	1975-02-23T02:58:41.000Z	8.017	124.075	Earthquake	623.0	5.6	MB	1.022784
7510	1985-04-28T02:53:41.530Z	-32.998	-71.766	Earthquake	33.0	5.6	MW	1.300000
20647	2011-03-13T02:23:34.520Z	36.344	142.344	Earthquake	10.1	5.8	MWC	1.060000

```
invalid_year_indices = data[data['Year'].str.contains('Z')].index

data = data.drop(invalid_year_indices, axis = 0).reset_index(drop = True)
```

```
data['Year'] = data['Year'].astype(np.int)
```

```
data['Hour'] = data['Time'].apply(lambda x: np.int(x[0:2]))

data = data.drop('Time', axis = 1)
```

	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type	Root Mean Square	Source	Location Source
0	19.2460	145.6160	Earthquake	131.60	6.0	MW	1.022784	ISCGEM	ISCGEM
1	1.8630	127.3520	Earthquake	80.00	5.8	MW	1.022784	ISCGEM	ISCGEM
2	-20.5790	-173.9720	Earthquake	20.00	6.2	MW	1.022784	ISCGEM	ISCGEM
3	-59.0760	-23.5570	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM
4	11.9380	126.4270	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM
...	...	...	...	...	...	...	...	...	...
23401	38.3917	-118.8941	Earthquake	12.30	5.6	ML	0.189800	NN	NN
23402	38.3777	-118.8957	Earthquake	8.80	5.5	ML	0.218700	NN	NN
23403	36.9179	140.4262	Earthquake	10.00	5.9	MWW	1.520000	US	US
23404	-9.0283	118.6639	Earthquake	79.00	6.3	MWW	1.430000	US	US
23405	37.3973	141.4103	Earthquake	11.94	5.5	MB	0.910000	US	US

PYTHON PROGRAM :

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Step 1: Load the dataset
def load_earthquake_data(file_path):
    data = pd.read_csv(file_path)  # Assuming the dataset is in CSV format
    return data

# Step 2: Preprocess the data
def preprocess_data(data):
    # Perform any necessary preprocessing steps, such as handling missing values, feature engineering, etc.
    # For simplicity, let's assume the dataset has features (X) and labels (y).
```

```
    # Example: Drop rows with missing values
    data = data.dropna()

    # Example: Split the data into features (X) and labels (y)
    X = data.drop('label_column', axis=1)
    y = data['label_column']

    # Example: Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    # Example: Standardize features using StandardScaler
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    return X_train_scaled, X_test_scaled, y_train, y_test

# Step 3: Train a machine learning model
def train_model(X_train, y_train):
    # Example: Use a Random Forest classifier
    model = RandomForestClassifier(n_estimators=100,
random_state=42)
    model.fit(X_train, y_train)
    return model
```

```
# Step 4: Evaluate the model
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy}")

# Example usage
if __name__ == "__main__":
    file_path = "path/to/earthquake_data.csv"

    # Load data
    earthquake_data = load_earthquake_data(file_path)

    # Preprocess data
    X_train, X_test, y_train, y_test = preprocess_data(earthquake_data)

    # Train model
    trained_model = train_model(X_train, y_train)

    # Evaluate model
    evaluate_model(trained_model, X_test, y_test)
```

Creating a complete earthquake prediction program involves several steps, from loading the dataset to preprocessing and training a model. Below is a simplified Python script using popular libraries such as pandas and scikit-learn for loading and preprocessing earthquake data. Keep in mind that this is a basic example, and real-world applications might require more sophisticated techniques.

CONCLUSION:

In conclusion, the success of earthquake prediction relies heavily on effective data loading and preprocessing. These initial steps lay the foundation for accurate and meaningful insights from machine learning models

In the dynamic field of earthquake prediction, the journey doesn't end with data loading and preprocessing. Continuous learning, adaptation, and innovation are key to refining models and advancing our understanding of seismic behavior.