Earthquake prediction model

Data preprocessing Techniques

1. *Import Libraries*:

- Import the necessary Python libraries:

- `pandas` for data manipulation.

- `numpy` for numerical operations.

- `train_test_split` from `sklearn.model_selection` to split the data into training and testing sets.

- `StandardScaler` from `sklearn.preprocessing` for feature standardization.

2. *Load the Dataset*:

- Load the earthquake dataset from the file located at '/content/database.csv' into a pandas DataFrame named `data`.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
data = pd.read_csv('/content/database.csv')
```

3. *Handling Missing Values*:

- Replace missing values in the dataset with the mean of their respective columns. This is done with the `fillna` method, which fills missing values with the mean of their columns. In this case, missing values are being replaced in all columns.

```
data.fillna(data.mean(), inplace=True)
```

4. *Handling Outliers*:

- Outliers in the 'Magnitude' column are being addressed. A common technique called the Z-score is used. It calculates the Z-score for each data point in the 'Magnitude' column, and data points with Z-scores greater than 3 (indicating they are 3 standard deviations away from the mean) are removed from the dataset.

```
z_score = np.abs((data['Magnitude'] - data['Magnitude'].mean()) / data['Magnitude'].std())
data = data[(z_score < 3)]
```

5. *Split the Data*:

- The data is split into features (X) and the target variable (y). The target variable is assumed to be in the 'Status' column. The features are all the columns except for 'Status.'

```
X = data.drop('Status', axis=1)
y = data['Status']
```

6. *Normalize or Standardize Features*:

- Feature standardization is performed to make the features have a mean of 0 and a standard deviation of 1. The 'StandardScaler' from scikit-learn is used for this purpose. It is applied to all features in the dataset (in the variable X).

```
# Normalize or Standardize a specific feature (e.g., 'Magnitude')
feature_to_normalize = 'Magnitude'
scaler = StandardScaler()
X[feature_to_normalize] = scaler.fit_transform(X[[feature_to_normalize]])
```

7. *Split Data into Training and Testing Sets*:

- The `train_test_split` function is used to divide the preprocessed data into training and testing sets. The data is divided in an 80-20 ratio, with 80% going to the training set (X_train and y_train) and 20% to the testing set (X_test and y_test). The `random_state` parameter is set to ensure reproducibility.

```
# Now, you can 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)
```