**EARTHQUAKE PREDICTION USING WITH PYTHON**

Documentation

This code is written in Python and is meant to perform a classification task to predict whether an earthquake occurred or not based on seismic data. Here's a natural language description of the code:

1. Import necessary libraries:

- The code starts by importing the required libraries, including Pandas for data manipulation, scikit-learn's `train\_test\_split` for data splitting, `RandomForestClassifier` for the machine learning model, and metrics for model evaluation.

2. Load and preprocess your seismic data:

- The code loads seismic data from a CSV file named 'data.csv'. This is where you would replace 'data.csv' with the actual filename of your dataset. Data preprocessing and feature engineering steps are commented out, indicating that you should perform these steps according to your specific dataset.

3. Define the target variable and features:

- The target variable, 'earthquake\_occurred,' represents whether an earthquake occurred (1) or not (0).

- The features (X) are all the columns in the dataset except for the 'earthquake\_occurred' column.

4. Split the data into training and testing sets:

- The dataset is divided into two subsets: the training set and the testing set. In this case, 80% of the data is used for training, and 20% is reserved for testing. A random seed of 42 is used to ensure reproducibility.

5. Create and train a Random Forest classifier:

- A Random Forest classifier is instantiated with 100 decision trees (n\_estimators=100) and a random seed of 42. It is then trained on the training data using the `.fit` method.

6. Make predictions on the test data:

- The trained model is used to make predictions on the test data (X\_test) using the `.predict` method.

7. Evaluate the model's performance:

- The code calculates the accuracy of the model by comparing the predicted labels (y\_pred) with the actual labels (y\_test) using the `accuracy\_score` function.

- It also generates a classification report using the `classification\_report` function, which provides more detailed information about the model's performance, such as precision, recall, and F1-score.

8. Print the results:

- The code prints the model's accuracy and the classification report, which includes a breakdown of the model's performance for each class.

SOURCE CODE:

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

# Load and preprocess your seismic data (replace 'data.csv' with your data file)

data = pd.read\_csv('data.csv')

# Perform data preprocessing and feature engineering

# Define the target variable and features

X = data.drop('earthquake\_occurred', axis=1) # Features

y = data['earthquake\_occurred'] # Target variable (1 if an earthquake occurred, 0 otherwise)

# 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)

# Create and train a Random Forest classifier

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

In summary, this code loads seismic data, splits it into training and testing sets, trains a Random Forest classifier, makes predictions, and evaluates the model's performance. You can adapt this code to your specific seismic dataset by performing the necessary data preprocessing and feature engineering.