PHASE-3

EARTHQUAKE PREDICTION MODEL USING PYTHON

TEAM MEMBERS:

1.NISHAANTHI.R
2.ABINAYA.P
3.DHARSHINI.Y
4.AISHWARYA LAKSHMI.M
5.SANGEETHA.M.L

Data Collection:

Gather historical earthquake data from sources like the USGS (United States Geological Survey) or other seismic data repositories.

Data Preprocessing:

Clean and preprocess the data. This involves handling missing values, normalizing or scaling features, and converting categorical data if necessary.

Feature Engineering:

Extract relevant features from the seismic data, such as magnitude, depth, location, and time of occurrence.

Machine Learning Model:

Choose an appropriate machine learning algorithm. Common choices include decision trees, random forests, support vector machines, or deep learning models like neural networks.

Training and Testing:

Split your data into a training set and a testing set to train and evaluate the model's performance. Ensure you have a sufficient amount of labeled earthquake and non-earthquake data.

Model Evaluation:

Use appropriate metrics to evaluate your model, such as accuracy, precision, recall, and F1 score.

Hyperparameter Tuning:

Fine-tune your model by adjusting hyperparameters to improve its performance.

Deployment:

If your model performs well, consider deploying it to monitor and predict earthquakes in real-time. However, this is a sensitive application and should involve collaboration with experts in the field.

Continuous Monitoring and Updates:
Earthquake prediction models may require continuous monitoring and regular updates as new data becomes available.

CODING:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load your preprocessed data and split it into features (X) and labels (y).
X, y = load_and_preprocess_data()

# 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 a Random Forest classifier and train it.
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set.
y_pred = model.predict(X_test)
```

```
# Evaluate the model.
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```