Chettinad
College of Engineering & Technology
Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Domain Name: Artificial Intelligence – Group 4

Project Title: Earthquake Prediction Model using Python

1. **Name of the Student (s):**

S. No	Name of the Student
1.	Pavithra R
2.	Aarthi S
3.	Lathika M
4.	Loganayaki M

Name of the Guide: Ms. Suganya S

Department/ Designation: CSE/AP

Institutional Address: Chettinad College of Engineering & Technology
NH-67 Karur-Trichy Highway, Puliyur
CF, Karur

Phone No. & Mobile No: 6374527207

Problem Statement :

The problem is to develop an earthquake prediction model using a Kaggle dataset. The objective is to explore and understand the key features of earthquake data, visualize the data on a world map for a global overview, split the data for training and testing, and build a neural network model to predict earthquake magnitudes based on the given features**.**

PHASE 3 DEVELOPMENT PART 1

1.DEFINE OBJECTIVES:

The objective of earthquake prediction is to develop methods and systems for forecasting seismic events, with the aim of minimizing the impact of earthquakes on human life, infrastructure, and society.

2.DATA COLLECTION:

- Acquire seismic data from sources such as seismometers, GPS, and satellite imagery.
- Gather geological and geophysical data related to the region of interest, including fault lines, historical earthquake records, and other relevant geological features

CODE:

```
from obspy import UTCDateTime
from obspy.clients.fdsn import Client


# Define the data center and time window for data collection
client = Client("IRIS")  # Replace with the appropriate data center


# Define the start and end times for the data collection window
start_time = UTCDateTime(2023, 1, 1, 0, 0, 0)  # Customize the start time
end_time = UTCDateTime(2023, 1, 2, 0, 0, 0)    # Customize the end time
```

```
# Specify the network, station, location, and channel codes
network = "XX"  # Customize the network code
station = "XXX"  # Customize the station code
location = ""  # Customize the location code
channel = "BHZ"  # Customize the channel code

# Fetch the seismic data
stream = client.get_waveforms(network, station, location, channel, start_time, end_time)

# Save the data to a file or perform further analysis
stream.write("seismic_data.mseed")  # Save data to MiniSEED format
```

DATA PROCESSING:

- Clean and preprocess the collected data. This may involve removing noise, correcting for sensor errors, and aligning data from different sources.
- Convert raw data into a usable format, such as time series data or feature vectors.
- Annotate the data with earthquake events, including their location, magnitude, and time of occurrence.

CODE:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Generate synthetic earthquake data (you would replace this with real data)
# Features: geological data, fault activity, historical seismic data, etc.
X = np.random.rand(1000, 5)
# Target variable: 1 if earthquake occurred, 0 if not
y = np.random.choice([0, 1], size=1000)
```

```
# Split 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 machine learning model (Random Forest classifier)
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
predictions = model.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

Data Storage using MySQL:

```
CREATE DATABASE earthquake_data;
USE earthquake_data;
CREATE TABLE seismic_data (
   id INT AUTO_INCREMENT PRIMARY KEY,
   timestamp DATETIME,
   latitude DECIMAL(10, 6),
   longitude DECIMAL(10, 6),
   magnitude DECIMAL(4, 2),
   location VARCHAR(255)
);

# Define a SQL table creation query
```

```
create_table_query = """
CREATE TABLE web_traffic (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user VARCHAR(255),
    page VARCHAR(255),
    timestamp DATETIME
)
import mysql.connector
# Connect to the MySQL database
connection = mysql.connector.connect(
    host="your_host",
    user="your_username",
    password="your_password",
    database="earthquake_data"
)

# Create a cursor to interact with the database
cursor = connection.cursor()

# Sample seismic data
data = [
    ("2023-01-01 08:00:00", 34.0522, -118.2437, 5.0, "Los Angeles"),
    ("2023-01-02 14:30:00", 40.7128, -74.0060, 4.5, "New York"),
    # Add more data here
]

# Insert data into the table
insert_query = "INSERT INTO seismic_data (timestamp, latitude, longitude,
magnitude, location) VALUES (%s, %s, %s, %s, %s)"

cursor.executemany(insert_query, data)
```

```
# Commit changes and close the connection

connection.commit()

connection.close()
```

Install necessary libraries:

#Create a Jupyter Notebook to write and run your Python code. You can start a Jupyter Notebook by running:

```
jupyter notebook
```

Import the necessary libraries:

#In your Jupyter Notebook, start by importing the required libraries:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

import matplotlib.pyplot as plt
```

Load and preprocess earthquake data.

```
# Split the data into training and testing sets:

X = earthquake_data.drop('earthquake_occurred', axis=1)  # Features

y = earthquake_data['earthquake_occurred']  # Target variable


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

Create and train a machine learning model:

Use a simple Random Forest classifier as an example:

```
model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)
```

Make predictions and evaluate the model:

```
predictions = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, predictions)
print("Model Accuracy:", accuracy)
```

Matplotlib for earthquake prediction:

```
import matplotlib.pyplot as plt
import pandas as pd

# Load earthquake data from a CSV file (replace with your data source)
earthquake_data = pd.read_csv('earthquake_data.csv')

# Extract relevant columns (e.g., latitude, longitude, and magnitude)
latitude = earthquake_data['latitude']
longitude = earthquake_data['longitude']
magnitude = earthquake_data['magnitude']

# Create a scatter plot to visualize earthquake locations and magnitudes
plt.figure(figsize=(10, 8))
plt.scatter(longitude, latitude, c=magnitude, cmap='viridis', s=magnitude * 10, alpha=0.6)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Earthquake Locations and Magnitudes')
plt.colorbar(label='Magnitude')
plt.grid(True)
plt.show()
```

Deployment and Real-time Monitoring:

Early Warning System:

Integrate the trained model into an early warning system that can analyze real-time seismic data and provide alerts or forecasts.

Continuous Monitoring:

Continuously monitor and update the model with new data, adapting to changing seismic conditions.

Collaboration:

Collaborate with relevant authorities, seismologists, and disaster management agencies to ensure that the developed model is part of a larger earthquake preparedness and response strategy.

Model Development:

Model Selection:

Choose a suitable machine learning or statistical model for earthquake forecasting. Common choices include recurrent neural networks (RNNs), convolutional neural networks (CNNs), Long Short-Term Memory (LSTM) networks, support vector machines (SVM), and more.

Training Data:

Split the preprocessed data into training and validation sets. Use labeled earthquake data to train the model, taking into account location, magnitude, and time of occurrence.

Model Training:

Train the chosen model on the training data, optimizing for the chosen prediction criteria (e.g., earthquake magnitude, location, probability, or time to event).