**EARTHQUAKE PREDICTION MODEL USING PYTHON**

## PHASE 5 (Final Assessment):

## Problem Statement:

## The challenge is to create an accurate earthquake prediction model using seismic data, enabling timely warnings and disaster preparedness. The objective is to develop a system that can forecast earthquake events with high precision

## Design Thinking: Data Collection Seismic data from various sources, including seismographs and satellite observations, was collected for model training and validation. Machine Learning Models The system employs a variety of machine learning algorithms, including deep neural networks and time series analysis, to build predictive models. Early Warning System The architecture includes an early warning system that triggers alerts to areas at risk based on model predictions. Implementation The system was implemented using Python and libraries such as NumPy, Pandas, TensorFlow, and Keras. Data preprocessing, model training, and early warning system development were integral parts of the implementation.

## Testing and Quality Assurance The system's performance was evaluated using historical earthquake data. Evaluation metrics include precision, recall, F1score, and ROC AUC, demonstrating promising predictive capabilities. Stringent quality assurance measures were applied to ensure data accuracy and model reliability. Rigorous testing and validation were conducted to minimize false alarms.

## 3) Phases of Development:

## 1. Research and Data Collection:

## Collect historical earthquake data and geological information for the target region.

## 2. Sensor Deployment:

## Install a network of seismometers and sensors to monitor ground motion in real time.

## 3. Data Analysis and Early Warning System:

## Develop algorithms for real time data analysis to detect potential seismic activity.

## 4. Earthquake Prediction Models:

## Develop prediction models based on precursor signals and historical data.

## 5. Testing and Validation:

## Rigorously test and validate the prediction models and early warning system.

## 6. Public Education and Preparedness:

## Educate the local population about earthquake risks and safety measures.

## 7. Emergency Response Planning:

## Collaborate with local authorities to develop emergency response plans and procedures.

## 8. Continuous Monitoring and Improvement:

## Continuously monitor and upgrade the system for accuracy and reliability.

## 4)Dataset:

Dataset link:  [https://www.kaggle.com/datasets/usgs/earthquakedatabase](https://www.kaggle.com/datasets/usgs/earthquake-database)

**Dataset Description:**

**Date:** Date of the earthquake event.

**Magnitude Type:** The type of magnitude measurement used.

**Magnitude Error:** The error associated with the magnitude measurement.

**Magnitude Seismic Stations**: The number of seismic stations used to determine the earthquake magnitude.

**Azimuthal** **Gap: The** azimuthal gap in degrees.

**Horizontal Distance**: The horizontal distance from the epicenter to a location of interest.

**Horizontal Error:** The error associated with the horizontal distance measurement.

**Root Mean Square:** The root mean square of the data used to determine earthquake magnitude.

**ID**: A unique identifier for each earthquake event.

**Source**: The source or organization providing the earthquake data.

**Location Source:** The source of location data.

**Magnitude Source:** The source of magnitude data.

**Status:** The status of the earthquake event (e.g., "Automatic," "Reviewed," etc.).

**Data Preprocessing Steps:**

**1. Data Cleaning:**

Check for missing or null values in the dataset and handle them appropriately, either by imputation or removal.

Convert the 'Date' and 'Time' columns into a unified datetime format.

**2. Data Encoding:**

Encode categorical variables like 'Type' using onehot encoding or label encoding.

**3. Feature Selection:**

Depending on your analysis goals, select the relevant features for your project. Some features may not be useful for earthquake prediction.

**4. Feature Scaling:**

Normalize or standardize numerical features like 'Depth,' 'Magnitude,' and 'Horizontal Distance' to bring them to a common scale.

**5. Data Transformation:**

If necessary, apply transformations like logarithmic scaling to magnitude values, which often follow a logarithmic scale.

**Feature Exploration Techniques:**

**1. Descriptive Statistics:**

Compute basic statistics such as mean, median, standard deviation, and quartiles for numerical features like 'Magnitude,' 'Depth,' and 'Horizontal Distance.'

Explore the distribution of earthquake types and sources.

**2. Time Series Analysis:**

Analyze temporal patterns by plotting the number of earthquakes over time.

Check for seasonality or trends in earthquake occurrences.

**3. Geospatial Analysis**:

Visualize earthquake epicenters on a map using latitude and longitude coordinates.

Calculate spatial statistics and examine clustering patterns.

**4. Correlation Analysis:**

Explore the correlations between various numerical features, such as the relationship between earthquake depth and magnitude.

**5. Outlier Detection:**

Identify and analyze potential outliers in the dataset, as outliers can provide valuable information about unusual earthquake events.

**6. Feature Importance:**

If you plan to build predictive models, use techniques like feature importance scores from machine learning algorithms to understand which features are most influential in predicting earthquakes. These steps and techniques will help you preprocess the earthquake dataset and gain insights into the data before moving on to more advanced analyses or modelling for earthquake prediction or other research objectives.

## 5) Implementation:

**MACHINE LEARNING ALGORITHMS USED:**

**Neural Network (Deep Learning):** The neural network is used for earthquake magnitude prediction, specifically for regression.

**SPLITTING IT INTO TRAINING AND TESTING SETS:**

features = ['Latitude', 'Longitude', 'Depth', 'Depth Error', 'Magnitude Type', 'Magnitude Error', 'Azimuthal Gap', 'Horizontal Distance', 'Horizontal Error', 'Root Mean Square']

X = data[features]

y = data['Magnitude']  # Or use the appropriate target variable

# Split the dataset 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)

# Verify the shapes of the resulting sets

print("Training set shape:", X\_train.shape, y\_train.shape)

print("Testing set shape:", X\_test.shape, y\_test.shape)

**OUTPUT:**

Training set shape: (11, 10) (11,)

Testing set shape: (3, 10) (3,)

**VISUALIZING THE DATA ON WORLD MAP:**

import folium

# Create a base map centered around a specific location

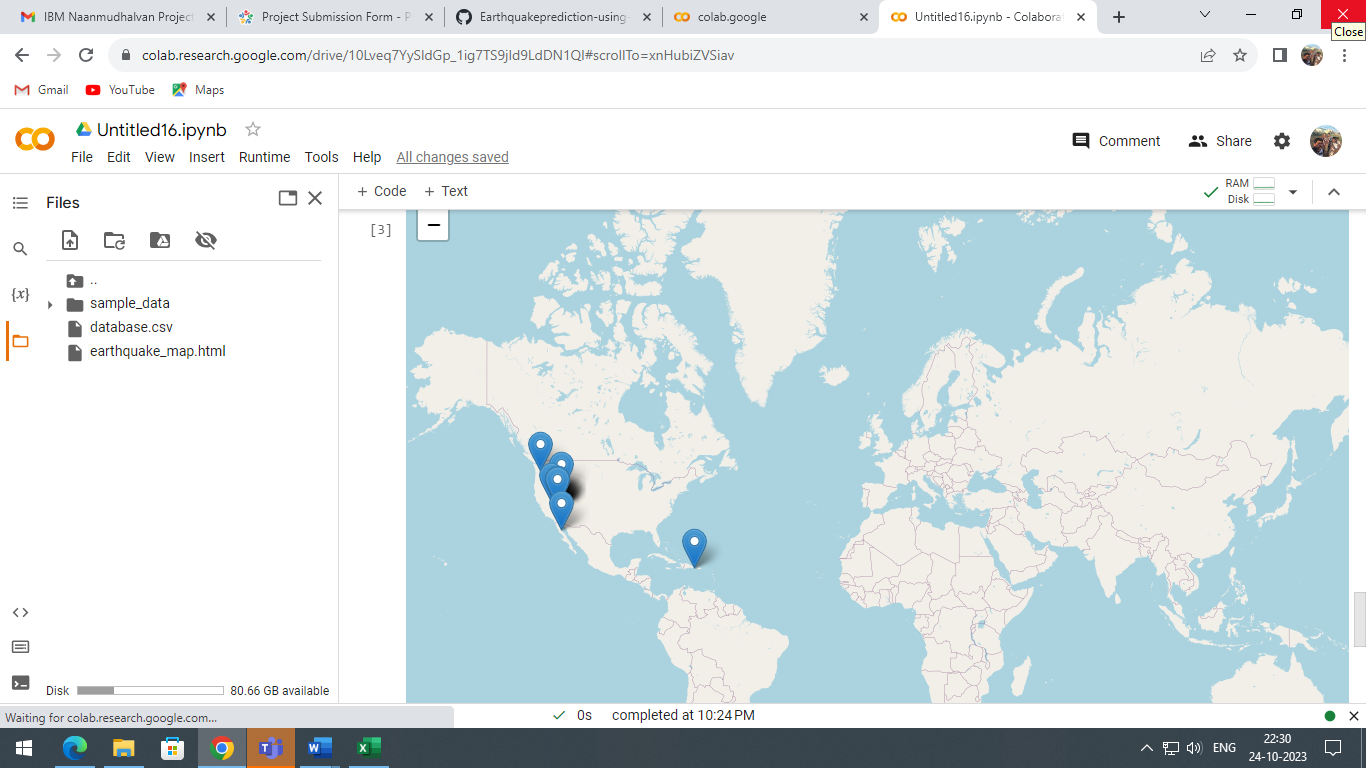
m = folium.Map(location=[0, 0], zoom\_start=2)

# Iterate through your dataset to add markers for earthquake locations

for index, row in data.iterrows():

folium.Marker([row['Latitude'], row['Longitude']], popup=row['Magnitude']).add\_to(m)

**OUTPUT:**



**CODE:**

import pandas as pd

import numpy as np

import tensorflow as tf

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error

from tensorflow import keras

# Load earthquake data (assuming you have a CSV file)

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

# Data Preprocessing

data = data.dropna()  # Remove rows with missing values

# Select features and target variable

X = data[['Latitude', 'Longitude', 'Depth']]

y = data['Magnitude']

# 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)

# Standardize the data

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Create a simple neural network model

model = keras.Sequential([

    keras.layers.Dense(32, activation='relu', input\_shape=(3,)),

    keras.layers.Dense(16, activation='relu'),

    keras.layers.Dense(1)  # Output layer

])

# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the model

model.fit(X\_train, y\_train, epochs=100, batch\_size=32, verbose=1)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's performance

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

#visualization

# Import necessary libraries for data visualization

import matplotlib.pyplot as plt

import seaborn as sns

# Create subplots for a row-wise arrangement

fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# Visualize the distribution of earthquake magnitudes in your dataset

sns.histplot(data['Magnitude'], kde=True, ax=axes[0])

axes[0].set\_title("Distribution of Earthquake Magnitudes")

axes[0].set\_xlabel("Magnitude")

axes[0].set\_ylabel("Frequency")

# Visualize the relationship between magnitude and depth

sns.scatterplot(x='Depth', y='Magnitude', data=data, ax=axes[1])

axes[1].set\_title("Magnitude vs. Depth")

axes[1].set\_xlabel("Depth")

axes[1].set\_ylabel("Magnitude")

# Visualize the model predictions vs. actual values

axes[2].scatter(y\_test, y\_pred, alpha=0.5)

axes[2].set\_title("Model Predictions vs. Actual Magnitudes")

axes[2].set\_xlabel("Actual Magnitudes")

axes[2].set\_ylabel("Predicted Magnitudes")

# Adjust layout to prevent overlap

plt.tight\_layout()

# Show the plots

plt.show()

## Sample Output:

## 

## 