**EARTHQUAKE PREDICTION MODEL USING PYTHON**

## PHASE 4 (Development Part 2):

Dataset link:  <https://www.kaggle.com/datasets/usgs/earthquake-database>

**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

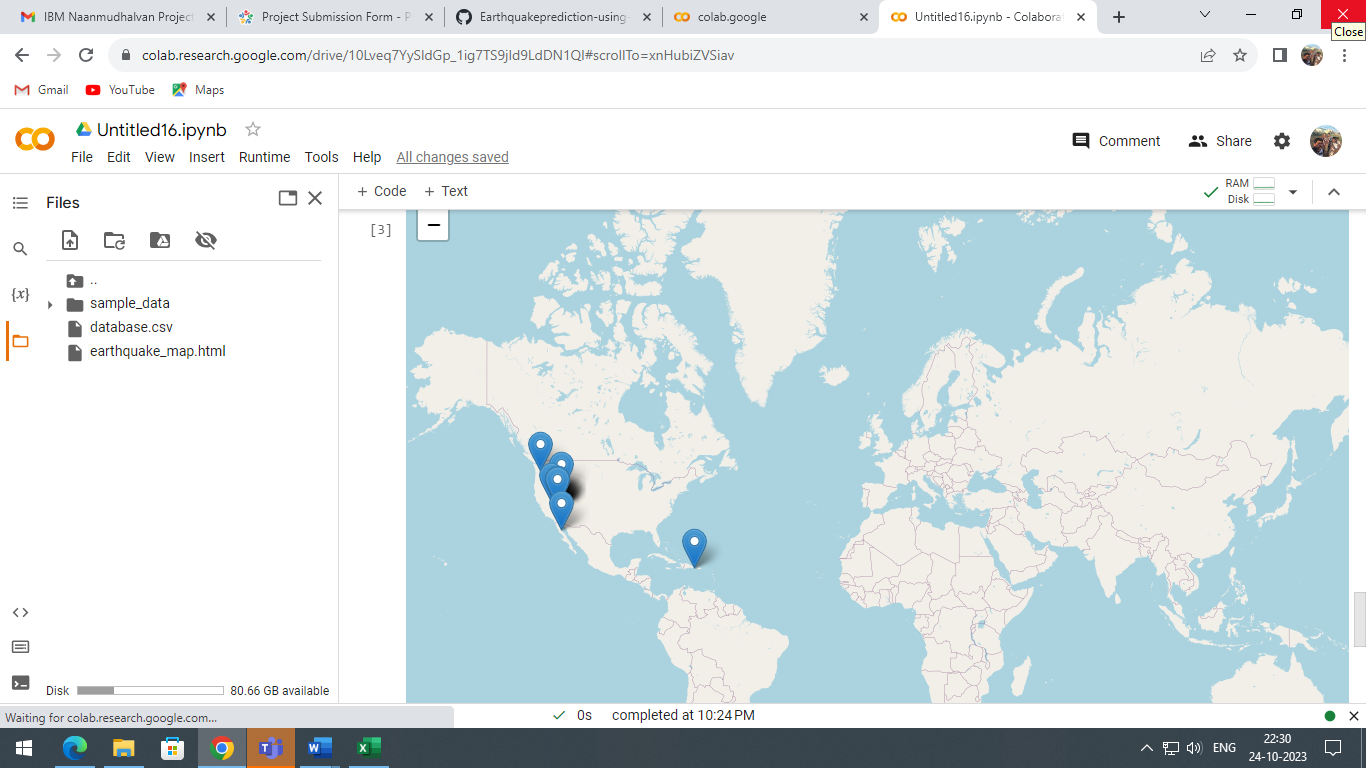
m = folium.Map(location=[0, 0], zoom\_start=2)

# 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:

## 

## 