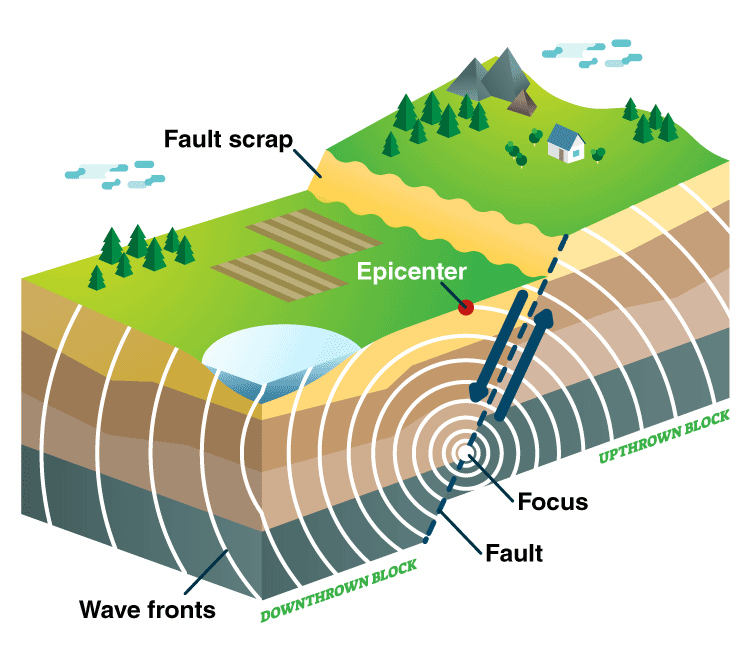
****AI-EARTHQUAKE PREDICTION MODEL USING PYTHON****

**Phase 4: Development Part 3**

**Name: ARAVINTHKUMAR.R**

**Reg.No:961221205009**

**Introduction:**

In this phase, we aim to visualize the earthquake data on a world map to gain geographic insights, split the data into training and testing sets for model evaluation, select a suitable machine learning algorithm, train the model using the training data, and evaluate its performance using appropriate metrics. 

**Procedure:**

**1. Visualizing Data on a World Map:**

Utilize geospatial visualization libraries like `folium` to plot earthquake data points on a world map. Use colors, sizes, or heat maps to represent earthquake magnitudes. This visualization provides a global overview of earthquake occurrences.

**Program:**

import folium

Assuming 'data' is your Data Frame containing earthquake data with columns 'latitude', 'longitude', and 'magnitude'

import pandas as pd

import folium

# Load your dataset (replace "data.csv" with your actual data file)

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

# Assuming 'data' is your DataFrame containing earthquake data with columns 'latitude', 'longitude', and 'magnitude'

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

for index, row in data.iterrows():

folium.CircleMarker(

location=[row['latitude'], row['longitude']],

radius=row['magnitude'], # Adjust the radius based on magnitude

color='crimson',

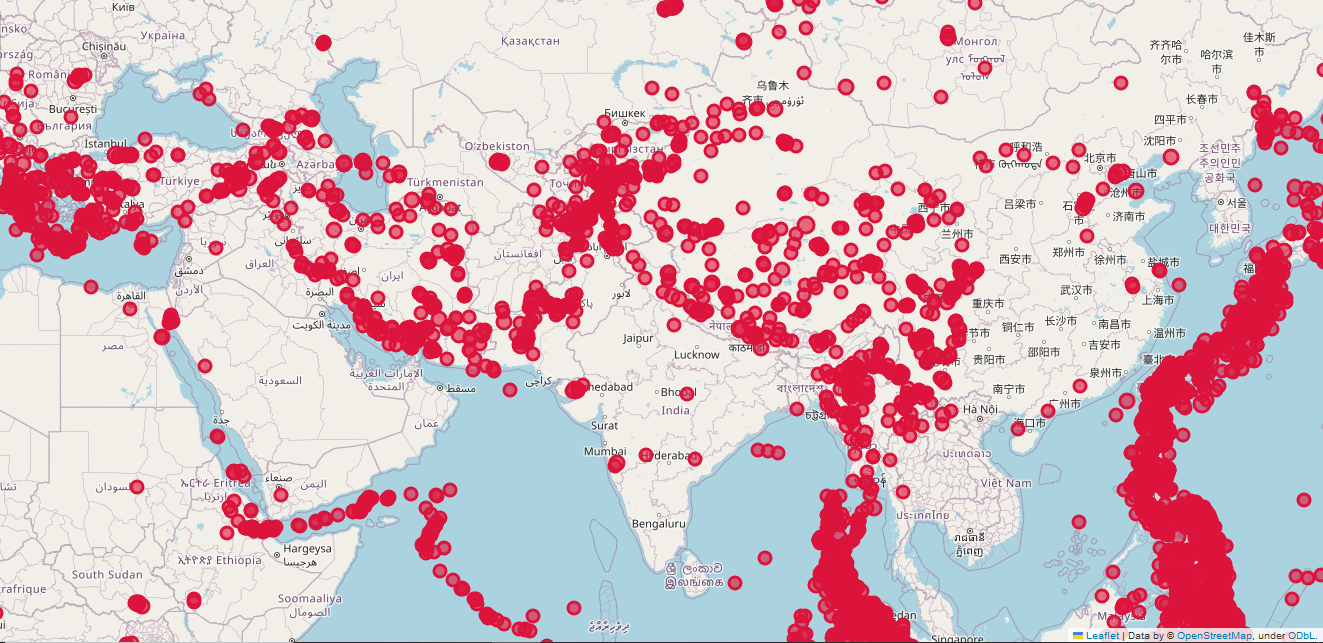
fill=True,

fill\_color='crimson',

fill\_opacity=0.6

).add\_to(map)

map.save("earthquake\_map.html")

**Output:**

This code will save an interactive HTML map showing earthquake occurrences.

**2. Splitting Data into Training and Testing Sets:**

Use `train\_test\_split` from scikit-learn to split the data into training and testing sets for model evaluation.

**Program:**

import pandas as pd

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

# Load your dataset (replace "data.csv" with your actual data file)

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

# Define features and target variable

features = ['latitude', 'longitude', 'depth', 'hour'] # Adjust features as needed

target = 'magnitude'

# Extract features (X) and target variable (y)

X = data[features]

y = data[target]

# Split the data into training and testing sets

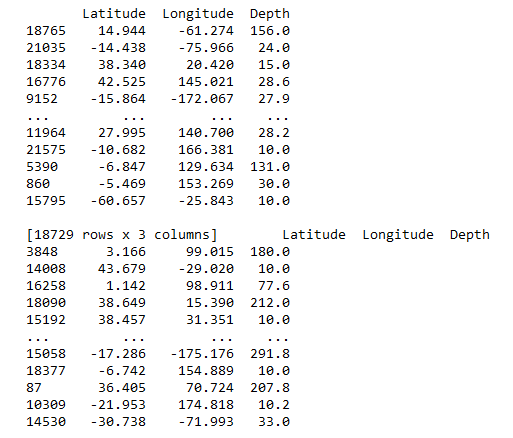
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

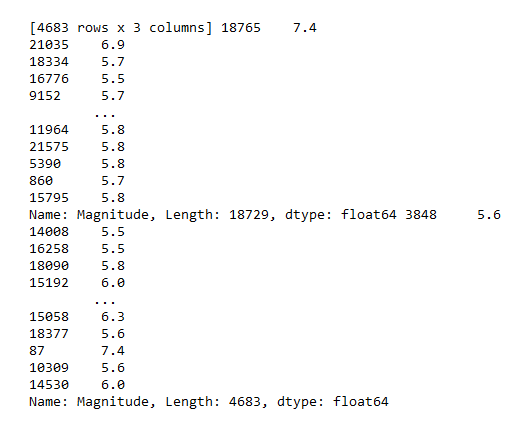
# Now, X\_train, X\_test, y\_train, and y\_test are available for further processing

# For example, you can train your machine learning model using X\_train and y\_train

# and evaluate its performance using X\_test and y\_test

**Output:**





**3. Selecting a Machine Learning Algorithm, Training, and Evaluation:**

Choose a machine learning algorithm (e.g., Random Forest, Gradient Boosting) and train the model using the training data. Evaluate the model using appropriate metrics (e.g., Mean Squared Error for regression tasks).

**Program:**

import pandas as pd

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

# Load your dataset (replace "data.csv" with your actual data file)

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

# Define features and target variable

features = ['latitude', 'longitude', 'depth', 'hour'] # Adjust features as needed

target = 'magnitude'

# Extract features (X) and target variable (y)

X = data[features]

y = data[target]

# 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 regression model

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

# Train the model using the training data

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

# Make predictions on the test set

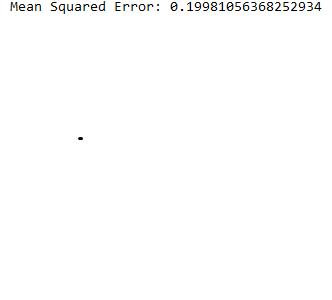
predictions = model.predict(X\_test)

# Calculate Mean Squared Error

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

print("Mean Squared Error:", mse)

**Output:**



**Conclusion:**

In this phase, we successfully visualized earthquake data on a world map, split it into training and testing sets, selected a Random Forest regression model, trained the model, and evaluated its performance using Mean Squared Error. Visualizing the data geographically provides valuable insights, and the model's evaluation metrics help assess its accuracy in predicting earthquake magnitudes.