DEVELOPMENT PART- 2 (PHASE 4)

Presented By:

1. S.MANICKAM ,
2. C.KARTHIGAILAKSHMI ,
3. S.SELVAVELRAJ ,
4. G.NANTHAKUMAR ,
5. B.AYYAPPAN .

Topic:

Continue building the earthquake prediction model by visualizing the data on a world map and splitting it into training and testing sets.

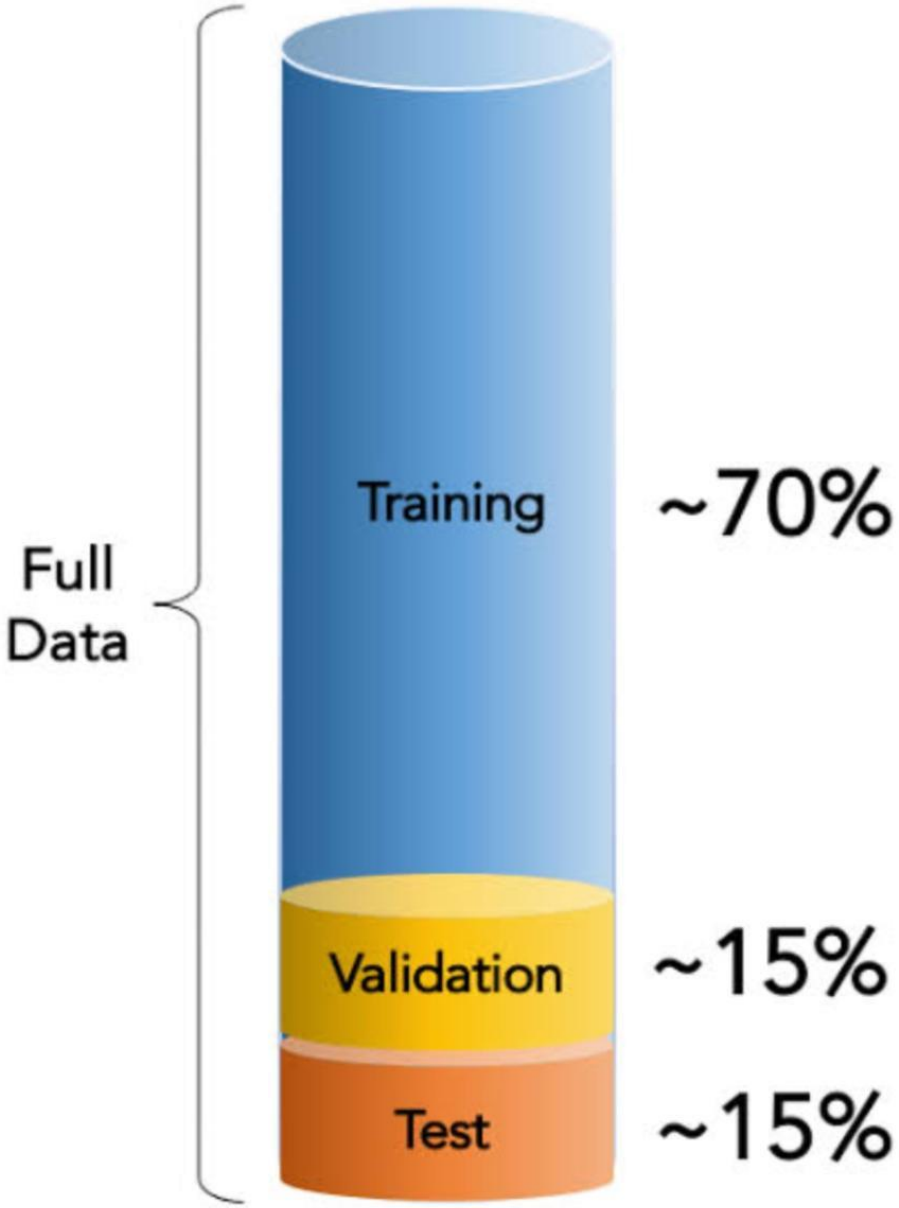
Definition :

An earthquake prediction model is a scientific approach that aims to forecast the occurrence, location, and strength of earthquakes based on geological and seismological data. These models help assess earthquake risk but do not provide precise short-term predictions.

Abstract :

"In the process of building an earthquake prediction model, data visualization plays a crucial role. Geospatial data visualization tools are employed to map earthquake occurrences on a world map, aiding in the understanding of geographical patterns. Subsequently, the dataset is partitioned into training and testing sets, with typically 70% allocated for training,15% allocated for validation and 15% for testing. This division ensures the model is trained on historical earthquake data and tested on unseen data, a fundamental step in model development."

Three Types Of Earthquake Prediction Model by visualization:



Spliting The Data :

Firstly, split the data into Xs and ys which are input to the model and output of the model respectively. Here, inputs are TImestamp, Latitude and Longitude and outputs are Magnitude and Depth. Split the Xs and ys into train and test with validation. Training dataset contains 70%, Validation dataset contains 15% and Test dataset contains 15%.

Components Of Data Split :

(1) Test Set
(2) Validation Set
(3) Training Set

Test Set:

The test set is a separate set of data used to test the model after completing the training**.**

Validation Set:

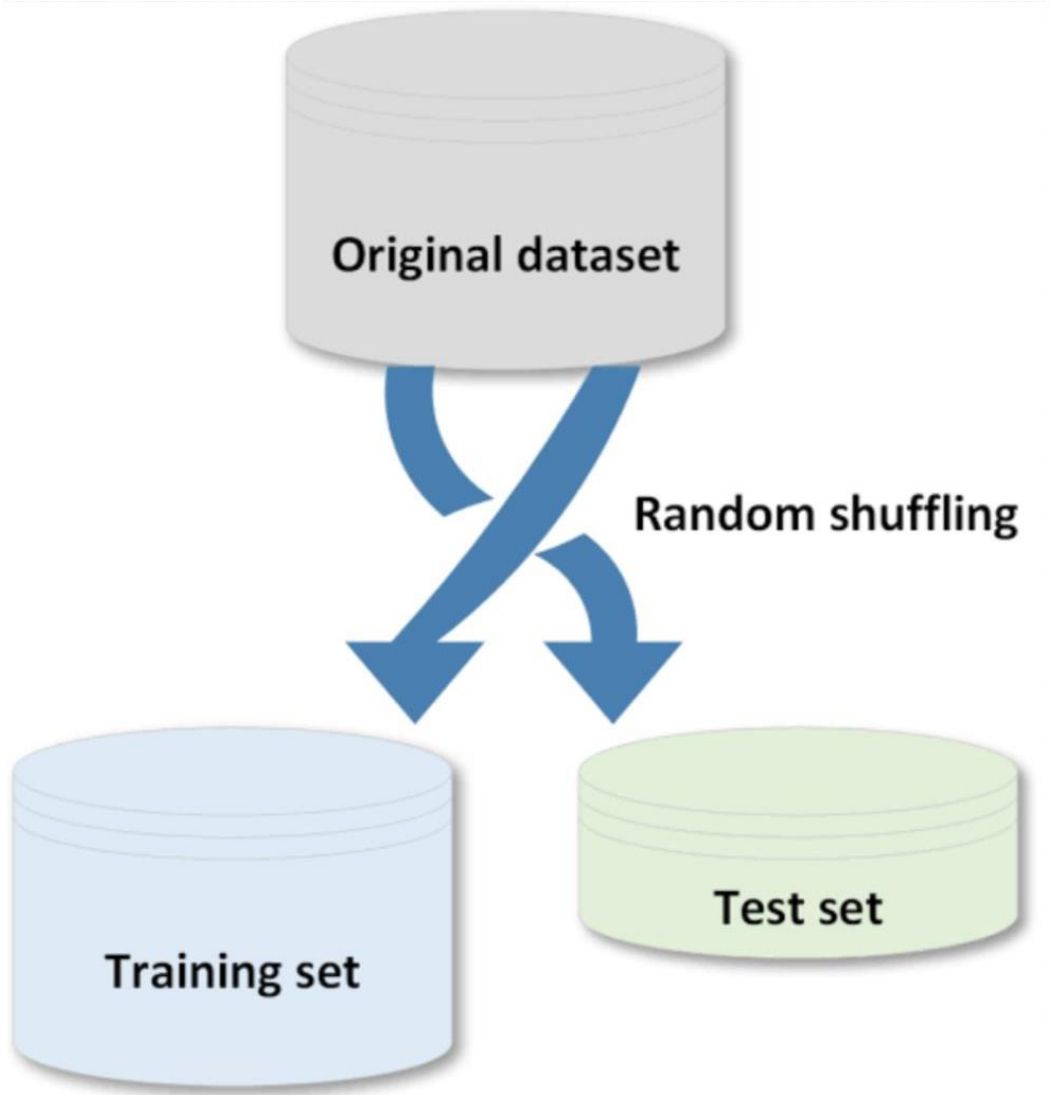
The validation set is a set of data, separate from the training set, that is used to validate our model performance during training.

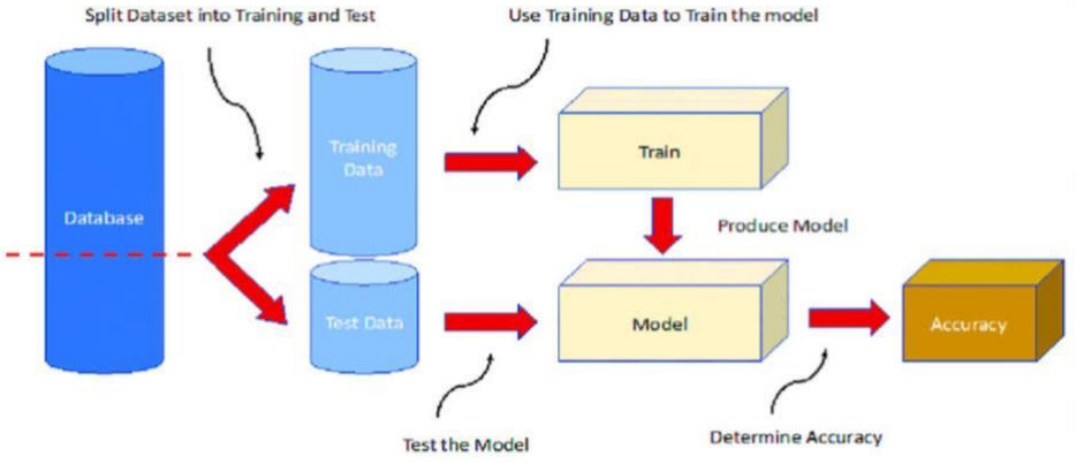
Training Set:

It is the set of data that is used to train and make the model learn the hidden features/patterns in the data.

Random Shuffling :

Random shuffling is a crucial step in preparing your data for training and testing in machine learning. It ensures that your data is randomly ordered, reducing the risk of bias and allowing for a representative split into training and test sets. Here are a few lines of Python code to perform random shuffling and define your training and test sets,


Original dataset
Random shuffling
Training set
Test set


Split Dataset into Training and Test
Use Training Data to Train the model
Database
Training Data
Test Data
Train
Produce Model
Model
Accuracy
Test the Model
Determine Accuracy

A New Technology To Predict Earthquakes is Invented:

In today's twenty-first century, technology has developed beyond our imagination.

New inventions are being introduced to the world day by day.

They make the daily activities of the people easier and benefit the lives of the people of the world.

Seismic Potential:

In that way, University of Texas researchers have successfully invented and tested an earthquake detection algorithm using artificial intelligence technology.

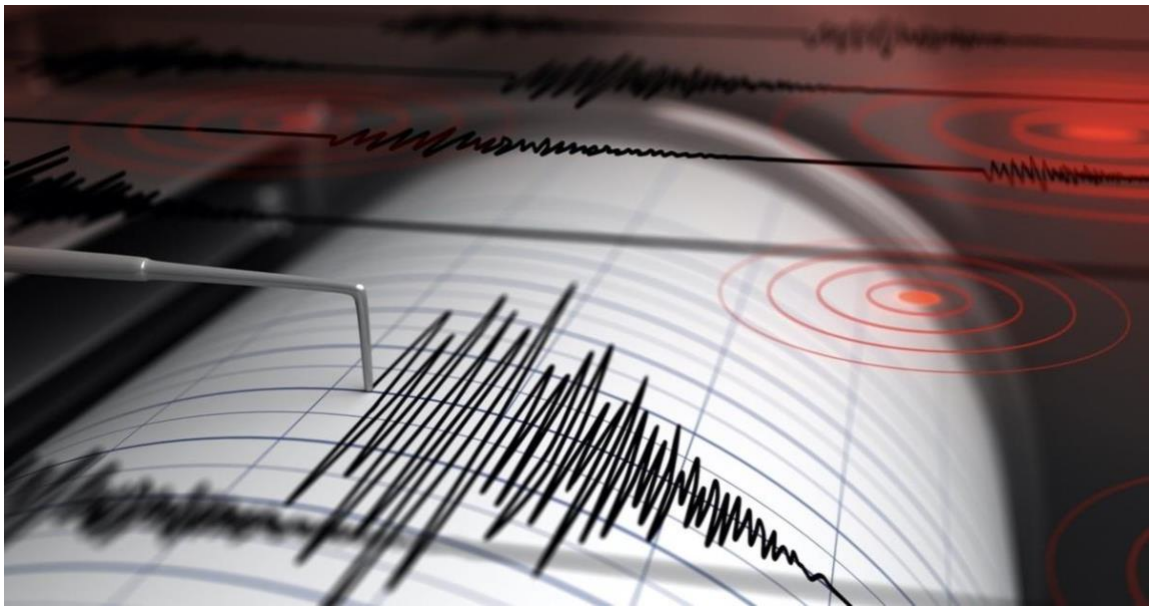
It is also said that it can be used to identify the zone where an earthquake is likely to occur a week in advance.

Distance :

Scientists say that this artificial intelligence system can even predict the size of an earthquake that can occur at a distance of approximately 321KM.

And to gauge its accuracy, scientists say tests are being conducted in seismically-prone countries like Italy, Japan and Greenland.

Diagram Of New Technology:



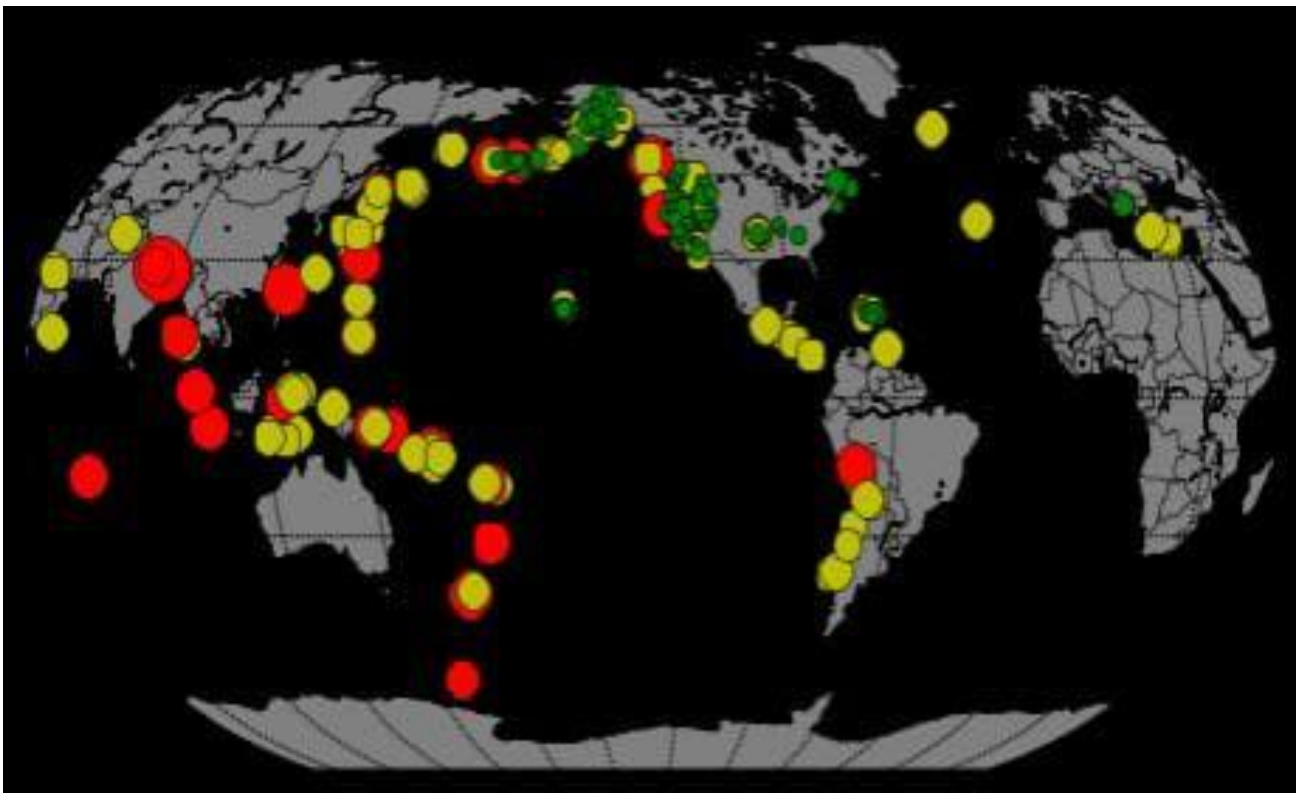
Program : 1

```
import csv

    # Open the earthquake data file.
    filename = 'datasets/earthquake_data.csv'

    # Create empty lists for the data we are interested in.
    lats, lons = [], []
    magnitudes = []

    # Read through the entire file, skip the first line,
    #  and pull out just the lats and lons.
    with open(filename) as f:
```

```
    # Create a csv reader object.
    reader = csv.reader(f)

    # Ignore the header row.
    next(reader)

    # Store the latitudes and longitudes in the appropriate lists.
    for row in reader:
        lats.append(float(row[1]))
        lons.append(float(row[2]))
        magnitudes.append(float(row[4]))

# --- Build Map ---
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np

def get_marker_color(magnitude):
    # Returns green for small earthquakes, yellow for moderate
    #  earthquakes, and red for significant earthquakes.
    if magnitude < 3.0:
        return ('go')
    elif magnitude < 5.0:
        return ('yo')
    else:
        return ('ro')

eq_map = Basemap(projection='robin', resolution = 'l',
area_thresh = 1000.0,
            lat_0=0, lon_0=-130)
eq_map.drawcoastlines()
eq_map.drawcountries()
```

```
eq_map.fillcontinents(color = 'gray')
eq_map.drawmapboundary()
eq_map.drawmeridians(np.arange(0, 360, 30))
eq_map.drawparallels(np.arange(-90, 90, 30))

min_marker_size = 2.5
for lon, lat, mag in zip(lons, lats, magnitudes):
    x,y = eq_map(lon, lat)
    msize = mag * min_marker_size
    marker_string = get_marker_color(mag)
    eq_map.plot(x, y, marker_string, markersize=msize)

plt.show()
```

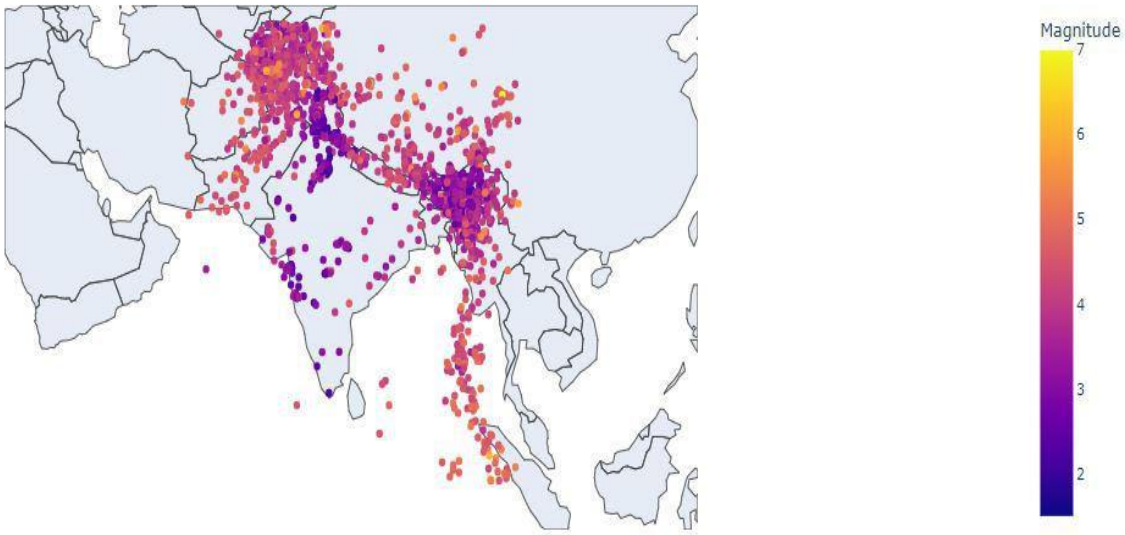
Output :



Program : 2

```
import plotly.express as px

import pandas as pd

fig = px.scatter_geo(df, lat='Latitude',

      lon='Longitude',

     color="Magnitude",

     fitbounds='locations',

      scope='asia')

fig.show()
```

Output :

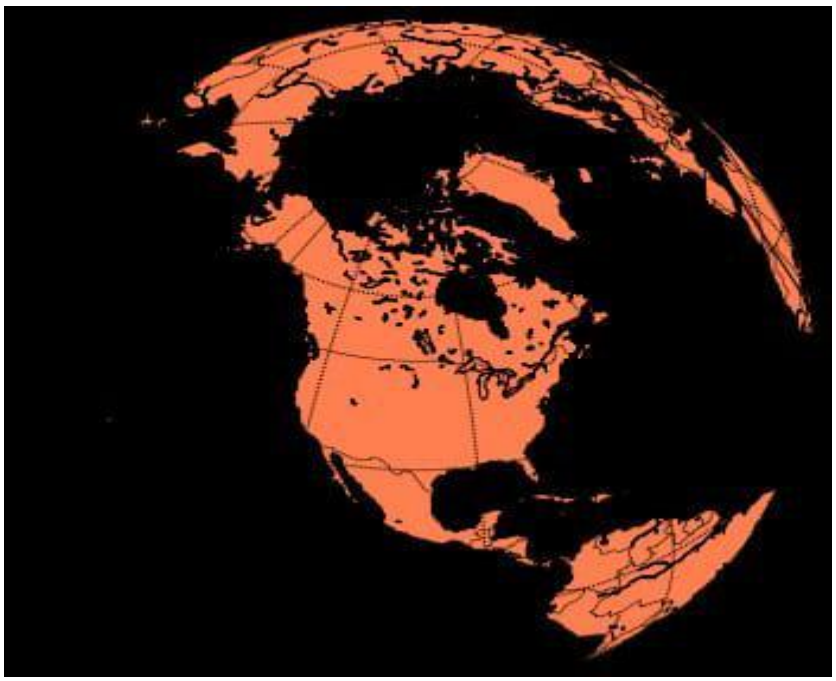


Program : 3

```
from mpl_toolkits.basemap import Basemap

import matplotlib.pyplot as plt

import numpy as np


# make sure the value of resolution is a lowercase L,

#  for 'low', not a numeral 1

my_map = Basemap(projection='ortho', lat_0=50, lon_0=-
100,

            resolution='l', area_thresh=1000.0)


my_map.drawcoastlines()

my_map.drawcountries()

my_map.fillcontinents(color='coral')

my_map.drawmapboundary()
```

```
my_map.drawmeridians(np.arange(0, 360, 30))

my_map.drawparallels(np.arange(-90, 90, 30))


plt.show()
```

Output :



Program : 4

```
# import modules


import pandas as pd


from sklearn.linear_model import LinearRegression


from sklearn.model_selection import train_test_split


# read the dataset


df = pd.read_csv('Real estate.csv')


# get the locations

X = df.iloc[:, :-1]
```

```
y = df.iloc[:, -1]

# split the dataset

X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.05, random_state=0)
```

Output :

```
1 X_train
```

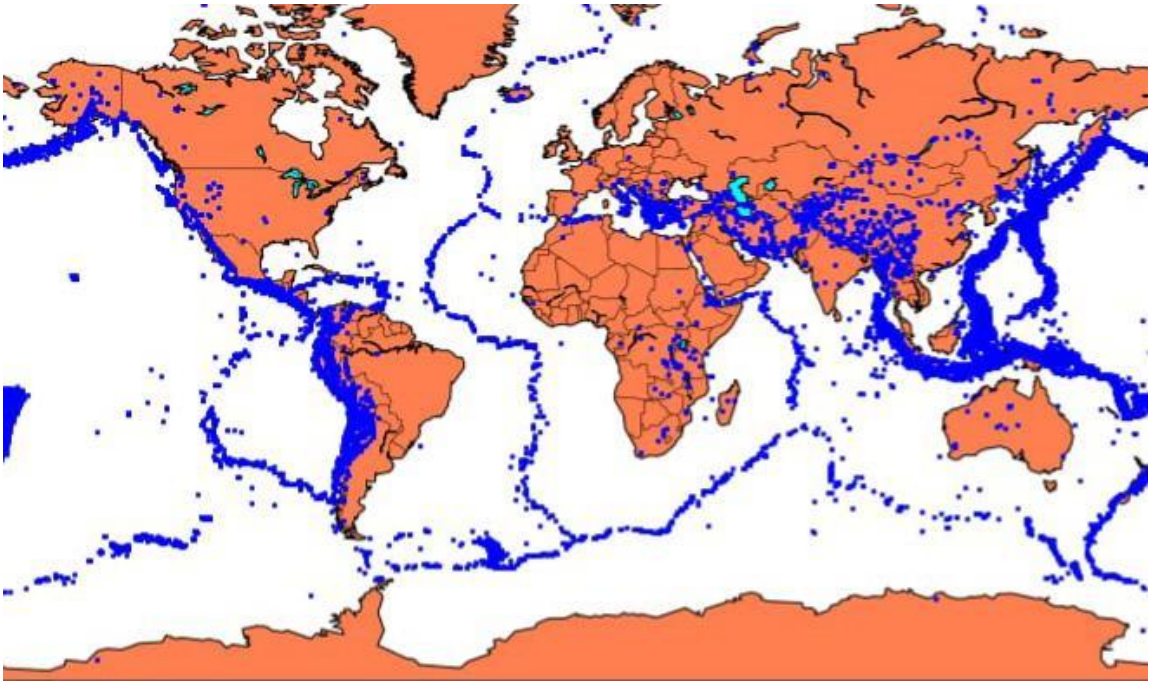
	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude
37	38	2013.167	12.0	1360.13900	1	24.95204	121.54842
334	335	2012.917	30.0	1013.34100	5	24.99006	121.53460
54	55	2013.083	16.1	289.32480	5	24.98203	121.54348
145	146	2012.917	2.1	451.24380	5	24.97563	121.54694
284	285	2012.917	15.0	383.28050	7	24.96735	121.54464
...	...	...	...	...	...	...	...
323	324	2013.417	28.6	197.13380	6	24.97631	121.54436
192	193	2013.167	43.8	57.58945	7	24.96750	121.54069
117	118	2013.000	13.6	4197.34900	0	24.93885	121.50383
47	48	2013.583	35.9	640.73910	3	24.97563	121.53715
172	173	2013.583	6.6	90.45606	9	24.97433	121.54310

Program : 5

```
# Feature Engineering

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score


# Load and preprocess your data

data = pd.read_csv('your_data.csv')

data = data.dropna()  # Handle missing values

data = pd.get_dummies(data,
columns=['categorical_feature'])  # Encode categorical
variables


# Define features (X) and target (y)

X = data.drop('target_column', axis=1)
```

```
y = data['target_column']


# Model Training

X_train, X_val, y_train, y_val = train_test_split(X, y,
test_size=0.2, random_state=42)


# Initialize and train the model

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

model.fit(X_train, y_train)


# Make predictions on the validation set

y_pred = model.predict(X_val)


# Evaluate the model

accuracy = accuracy_score(y_val, y_pred)

print(f'Validation Accuracy: {accuracy}')
```

Output :



Program : 6

```
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=5000,
n_features=20, n_informative=15)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=True)

print(len(X),len(X_train),len(X_test))
```

Output :

```
    Total Train Test
 5000  4000  1000
```

Program : 7

```
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
data =
pd.read_csv('http://apmonitor.com/pds/uploads/Main/tclab_
data6.txt')

data.set_index('Time',inplace=True)



# Split into train and test subsets (20% for test)

train, test = train_test_split(data, test_size=0.2,
shuffle=False)



print('Train: ', len(train))

print(train.head())

print('Test: ', len(test))

print(test.head())
```

Output:

```
Train:  2880

        Q1   Q2     T1     T2
```

Time

0.0 0.0 0.0 16.06 16.00

1.0 0.0 0.0 16.06 15.97

2.0 0.0 0.0 16.06 16.03

3.0 0.0 0.0 16.03 16.00

4.0 0.0 0.0 16.03 15.94

Test: 721

Q1 Q2 T1 T2

Time

2880.0 0.0 0.0 59.25 27.02

2881.0 0.0 0.0 59.22 27.02

2882.0 0.0 0.0 58.99 27.02

2883.0 0.0 0.0 58.93 27.02

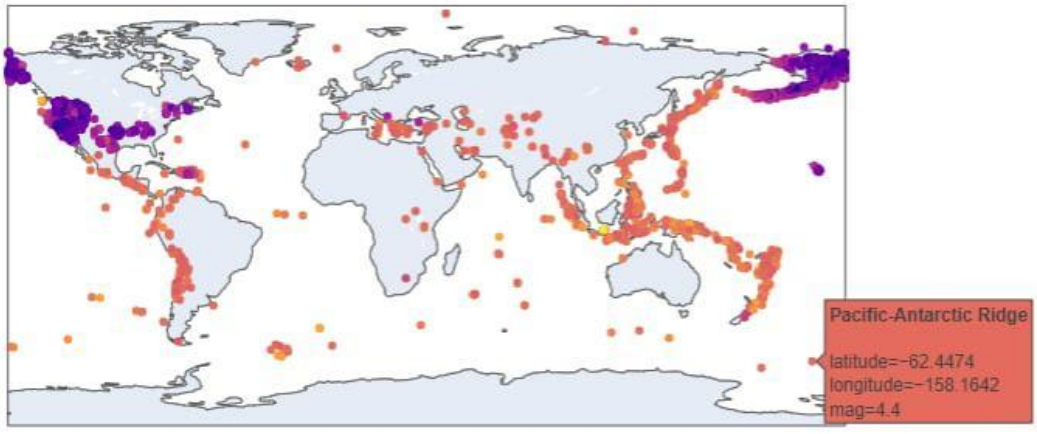
2884.0 0.0 0.0 58.93 27.02

Program : 8

```
import plotly.express as px

import pandas as pd



# Import data from USGS

data =
pd.read_csv('https://earthquake.usgs.gov/earthquakes/feed
/v1.0/summary/all_month.csv')



# Drop rows with missing or invalid values in the 'mag'
column

data = data.dropna(subset=['mag'])

data = data[data.mag >= 0]
```

```
# Create scatter map

fig = px.scatter_geo(data, lat='latitude', lon='longitude',
color='mag',

                    hover_name='place', #size='mag',

                    title='Earthquakes Around the World')

fig.show()
```

Output :



Feature Engineering instructions :

In the feature engineering phase, follow your project's instructions by:

1. Analyzing Data: Carefully study your dataset to understand its characteristics and domain-specific insights.

2. Selecting Features: Choose the most relevant features based on the project's objectives and data analysis.

3. Transforming Data: Apply necessary transformations such as scaling, encoding, and normalization to prepare features for modeling.

4. Creating New Features: Generate additional features if required to capture unique information.

5. Handling Missing Data: Implement strategies for managing missing values as per project instructions.

6. Managing Outliers: Address outliers according to the project's guidelines, which might involve treatment or removal.

These activities should align with your project's specific instructions and goals.

Model Training Of Instructions :

To continue building the project, follow the instructions provided in the project documentation or guidelines. This may involve tasks like data preprocessing, model training, and evaluation. Make sure to refer to the specific project instructions for detailed steps and code examples.

Evaluation Of Instructions :

In this section, continue building the project by focusing on model evaluation. Follow the instructions provided in the project guidelines to assess the performance of your model. This may involve metrics like accuracy, precision, recall, or others, depending on the nature of your project. Be sure to record and analyze the evaluation results to make informed decisions for further improvements in your project.

Conclusion :

Visualizing earthquake data on a world map provides valuable insights into the geographic distribution of seismic events, aiding in the model-building process. The partitioning of data into training and testing sets is essential for evaluating the model's performance. These initial steps set the foundation for the development of an accurate earthquake prediction model, whIch, when fine-tuned and deployed, can contribute to proactive disaster management and public safety.