4222- SURYA GROUP OF INSTITUTIONS

VIKRAVANDI-605625

NAAN MUDHALVAN PROJECT

EARTHQUAKE PREDICTION MODEL USING PYTHON

AI- Phase 3 (DEVELOPMENT PART -1)

PREPARED BY

K.ARUN

422221106003

ECE DEPT

INTRODUCTION:

Earthquake prediction is a way of predicting the magnitude of an earthquake based on parameters such as longitude, latitude, depth, and duration magnitude, country, and depth using machine learning to give warnings of potentially damaging earthquakes early enough to allow appropriate response to the disaster, enabling people to minimize loss of life and property.

PREPROCESSING:

An earthquake preprocessing dataset is a meticulously curated collection of raw seismic data gathered from various sensors and sources before undergoing any analysis or interpretation. This dataset serves as the foundational information for earthquake researchers aseismologists. Through preprocessing, the raw data is cleaned, filtered, and transformed to enhance its quality and usability. This process involves removing noise, correcting inconsistencies, and standardizing formats, ensuring that the dataset is reliable and ready indepth analysis. Preprocessed earthquake datasets are invaluable resources, enabling scientists to conduct accurate studies, predict seismic events, and enhance our understanding of the Earth's dynamic geological processes.

PREPROCESSING STEP:

1. Data Collection: Gather raw seismic data from various seismometers and networks.

2. Data Conversion: Convert data from different formats into a standard format for consistency.

3. Noise Removal: Remove noise from the data to enhance the signal-to-noise ratio. This can involve using filters or statistical methods to identify and remove unwanted noise.

4. Instrument Response Removal: Correct for the instrument's response to seismic waves. Seismic instruments often have a specific frequency response that needs to be removed to obtain accurate earthquake signals.

5. Data Segmentation: Divide the continuous data into smaller segments for analysis. Common segments include hours or days.

6. Event Detection: Use algorithms to detect earthquake events within the segmented data. These algorithms can identify patterns associated with seismic events.

7. Event Association: Identify which seismic signals belong to the same earthquake event. This can involve clustering nearby seismic events in both time and space.

8. Phase Picking: Identify the arrival times of seismic waves (P, S, etc.) for each event. This is crucial for locating the epicenter and determining the magnitude.

9. Hypocenter Location: Calculate the earthquake's hypocenter (latitude, longitude, and depth) using the phase arrival times from different seismometers. Various methods, like triangulation, are used for this purpose.

10. Magnitude Estimation: Determine the earthquake's magnitude using the amplitude of seismic waves. Common magnitude scales include Richter scale and moment magnitude scale.

11. Data Integration: Integrate the earthquake location, magnitude, and other relevant information into a database or a suitable format for further analysis and visualization.

12. Quality Control: Perform quality checks at various stages of the preprocessing to ensure the accuracy and reliability of the results.

ML MODELS USED:

- Linear Regression
- Decision Tree
- K-Nearest Neighbors

STEP TAKEN:

1. Splitting the Datasets
2. Building ML models And Data Analysis
3. Visualization
4. Prediction

1.SPLITTING THE DATASETS:

```
X = final_data[['Timestamp', 'Latitude', 'Longitude']]
y = final_data[['Magnitude', 'Depth']]

In [11]:
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
=42)
print(X_train.shape, X_test.shape, y_train.shape, X_test.shape)
(18727, 3) (4682, 3) (18727, 2) (4682, 3)
from sklearn.ensemble import RandomForestRegressor

reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
reg.predict(X_test)
         array([[  5.96,   50.97],
                 [   5.88,   37.8 ],
                 [   5.97,   37.6 ],
                 ...,
                 [   6.42,   19.9 ],
                 [   5.73, 591.55],
                 [   5.68,   33.61]])
reg.score(X_test, y_test)
```

2.Building ML models And Data Analysis:

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

import os
print(os.listdir("../input"))
```

	Date	Time	Latitude	Langitude	Depth	Magnitude
0	01/02/1965	13:44:18	19.246	145.616	131.6	6.0
1	01/04/1965	11:29:49	1.863	127.352	80.0	5.8
2	01/05/1965	18:05:58	-20.579	-173.972	20.0	6.2
3	01/08/1965	18:49:43	-59.076	-23.557	15.0	5.8
4	01/09/1965	13:32:50	11.938	126.427	15.0	5.8

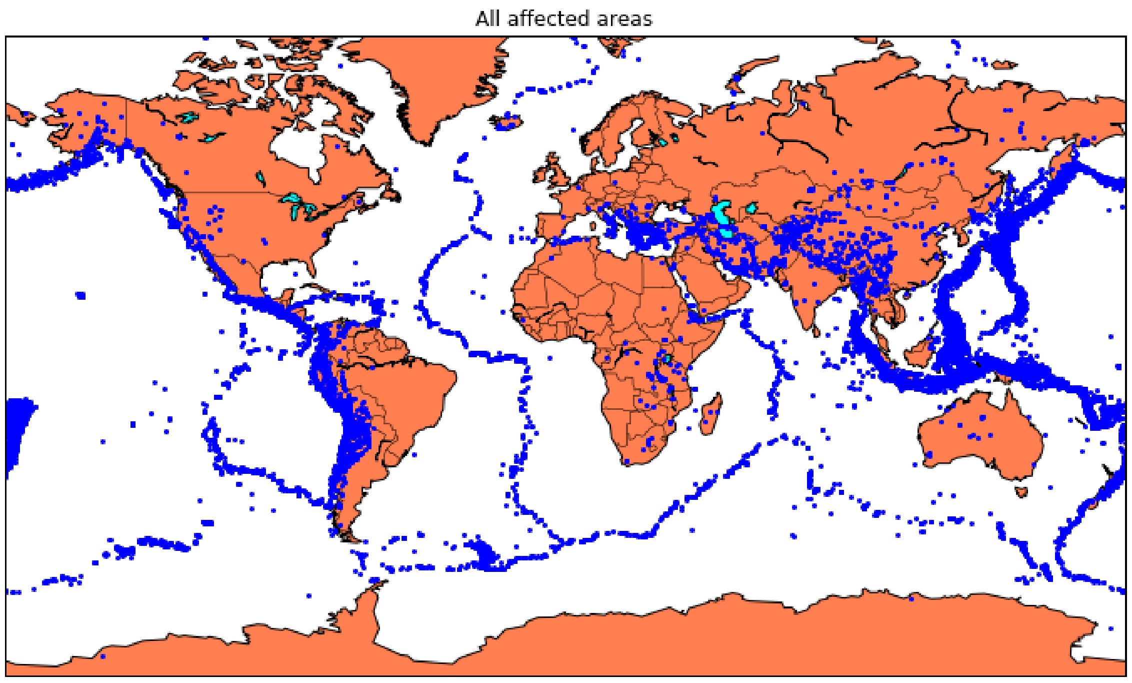
3.Visualization:

```
from mpl_toolkits.basemap import Basemap

m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=18
0,lat_ts=20,resolution='c')

longitudes = data["Longitude"].tolist()
latitudes = data["Latitude"].tolist()
#m = Basemap(width=12000000,height=9000000,projection='lcc',
            #resolution=None,lat_1=80.,lat_2=55,lat_0=80,lon_0=-107.)
x,y = m(longitudes,latitudes)
```

```
In [9]:
linkcode
fig = plt.figure(figsize=(12,10))
plt.title("All affected areas")
m.plot(x, y, "o", markersize = 2, color = 'blue')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```



4.Prediction:

```
import datetime
import time

timestamp = []
for d, t in zip(data['Date'], data['Time']):
    try:
        ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
        timestamp.append(time.mktime(ts.timetuple()))
    except ValueError:
        # print('ValueError')
        timestamp.append('ValueError')
```

```
  timeStamp = pd.Series(timestamp)
data['Timestamp'] = timeStamp.values
final_data = data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-1.57631e+08
1	1.863	127.352	80.0	5.8	-1.57466e+08
2	-20.579	-173.972	20.0	6.2	-1.57356e+08
3	-59.076	-23.557	15.0	5.8	-1.57094e+08
4	11.938	126.427	15.0	5.8	-1.57026e+08

CONCLUSION:

This work presents that the Random Forest Classifier algorithm has the highest accuracy in predicting the damage due to earthquakes, based on the F1 score calculated for each of the four algorithms previously mentioned in this work. KNearest Neighbors has been observed to be the second most preferred algorithm for earthquake damage prediction. On analysis of the materials that help curb damage to buildings during an earthquake, the work concludes that Reinforced Concrete is the material most suited to the cause.