EARTHQUAKE PREDICTION USING MACHINE LEARNING:

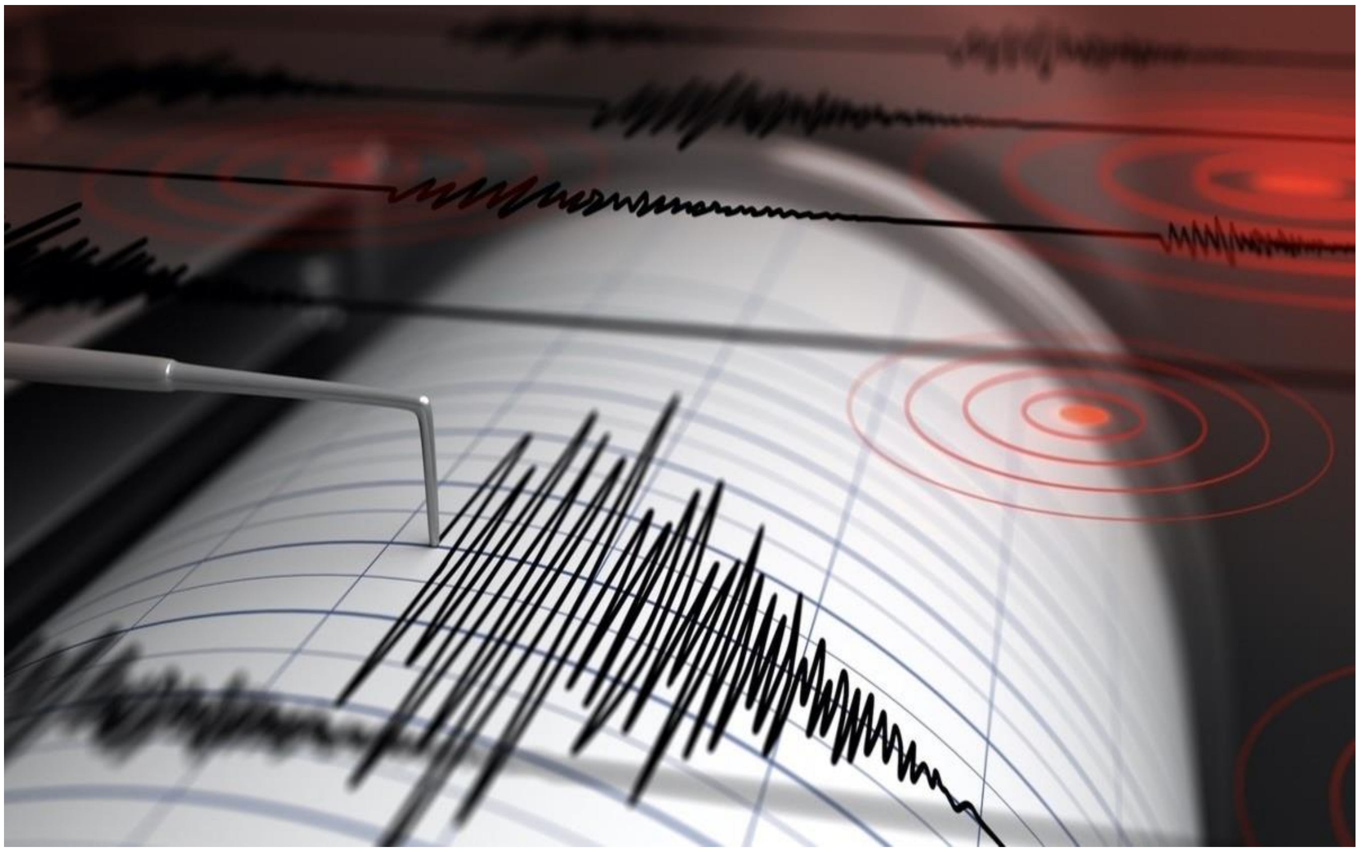
BATCH NUMBER

961621205017: NANDANA P A

PHASE 3 SUBMISSION DOCUMENT

PROJECT TITLE :EARTHQUAKE PREDICTION MODEL USING PYTHON

PHASE 3: DEVELOPMENT PART 1



Earthquake Prediction Using Machine Learning:

Machine learning has the ability to advance our knowledge of earthquakes and enable more accurate forecasting and catastrophe response. It's crucial to remember that developing accurate and dependable prediction models for earthquakes still needs more study as it is a complicated and difficult topic.

In order to anticipate earthquakes, machine learning may be used to examine seismic data trends. Seismometers capture seismic data, which may be used to spot changes to the earth's surface, like seismic waves brought on by earthquakes. Machine learning algorithms may utilize these patterns to forecast the risk of an earthquake happening in a certain region by studying these patterns and learning to recognize key traits that are linked to seismic activity.

So we will be predicting the earthquake fromDate and Time, Latitude, and Longitude from previous data is not a trend that follows like other things. It is naturally occurring.

Code:

Importing Libraries

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

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

Output:

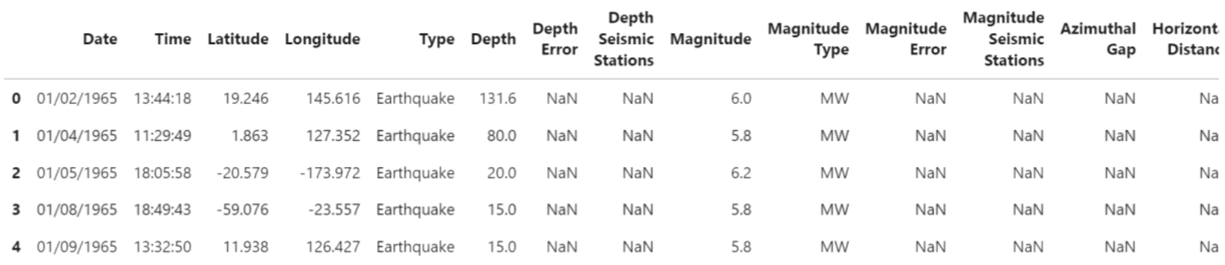
```
['database.csv']
```

Read the Dataset

Now we will read the dataset and look for the various features in the dataset.

```python
data = pd.read_csv("../input/database.csv")
```

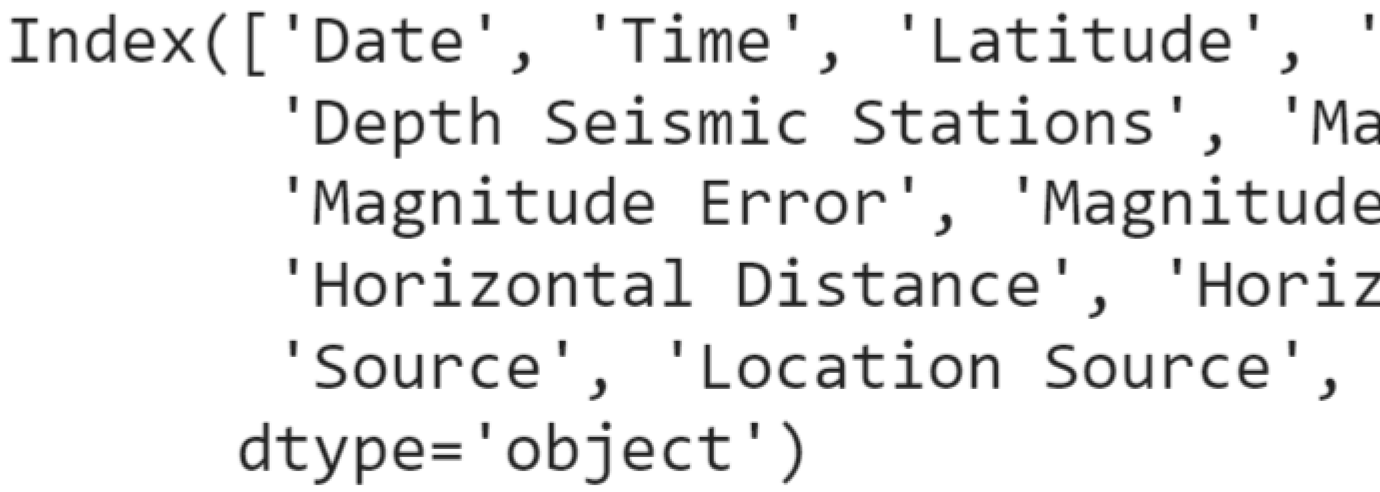
```
2. data.head()
```



	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	Magnitude Error	Magnitude Seismic Stations	Azimuthal Gap	Horizont Distanc
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	NaN	NaN	NaN	Na
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	Na
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	NaN	NaN	NaN	Na
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	Na
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	Na

```
1. data.columns
```

Output:

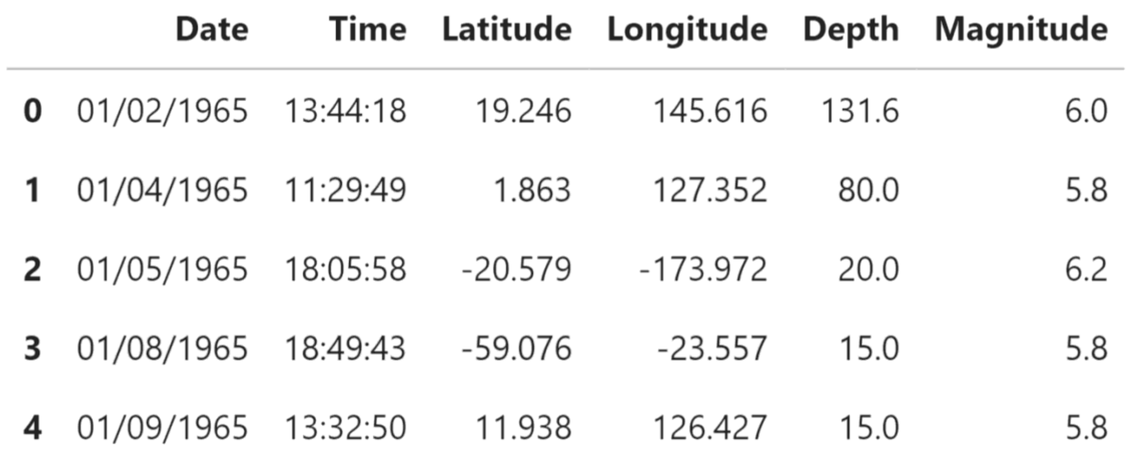


```
Index(['Date', 'Time', 'Latitude', '
       'Depth Seismic Stations', 'Ma
       'Magnitude Error', 'Magnitude
       'Horizontal Distance', 'Horiz
       'Source', 'Location Source',
      dtype='object')
```

We need to select the features that will be useful for our prediction.

```
1. data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]
2. data.head()
```

Output:



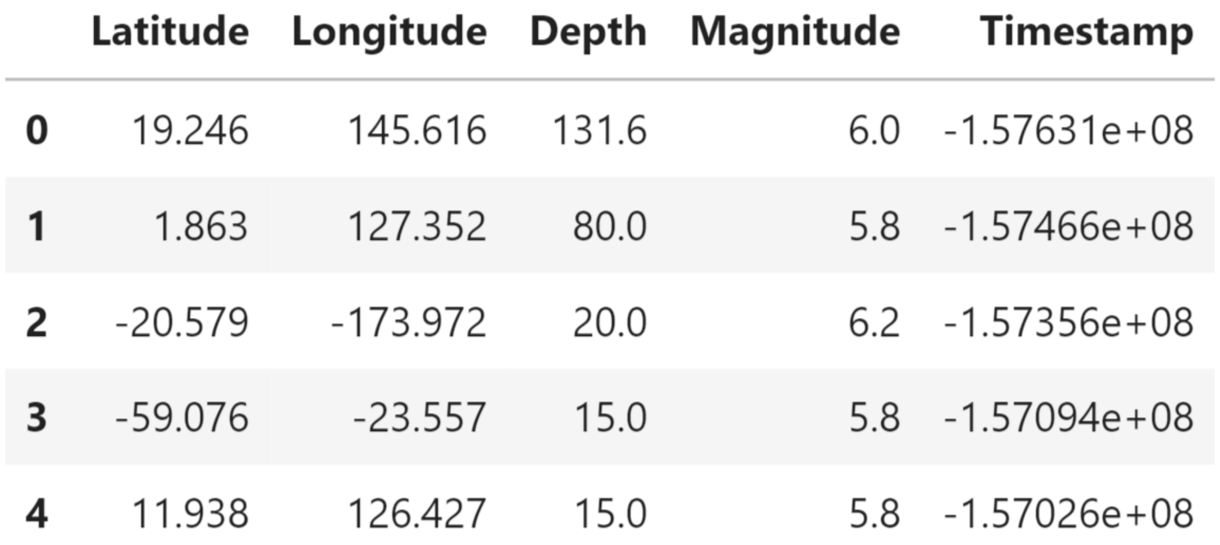
	Date	Time	Latitude	Longitude	Depth	Magnitude
0	01/02/1965	13:44:18	19.246	145.616	131.6	6.0
1	01/04/1965	11:29:49	1.863	127.352	80.0	5.8
2	01/05/1965	18:05:58	-20.579	-173.972	20.0	6.2
3	01/08/1965	18:49:43	-59.076	-23.557	15.0	5.8
4	01/09/1965	13:32:50	11.938	126.427	15.0	5.8

We will try to frame the time and place of the earthquake that has happened in the past on the world map.

```
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()
```

Output:



	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

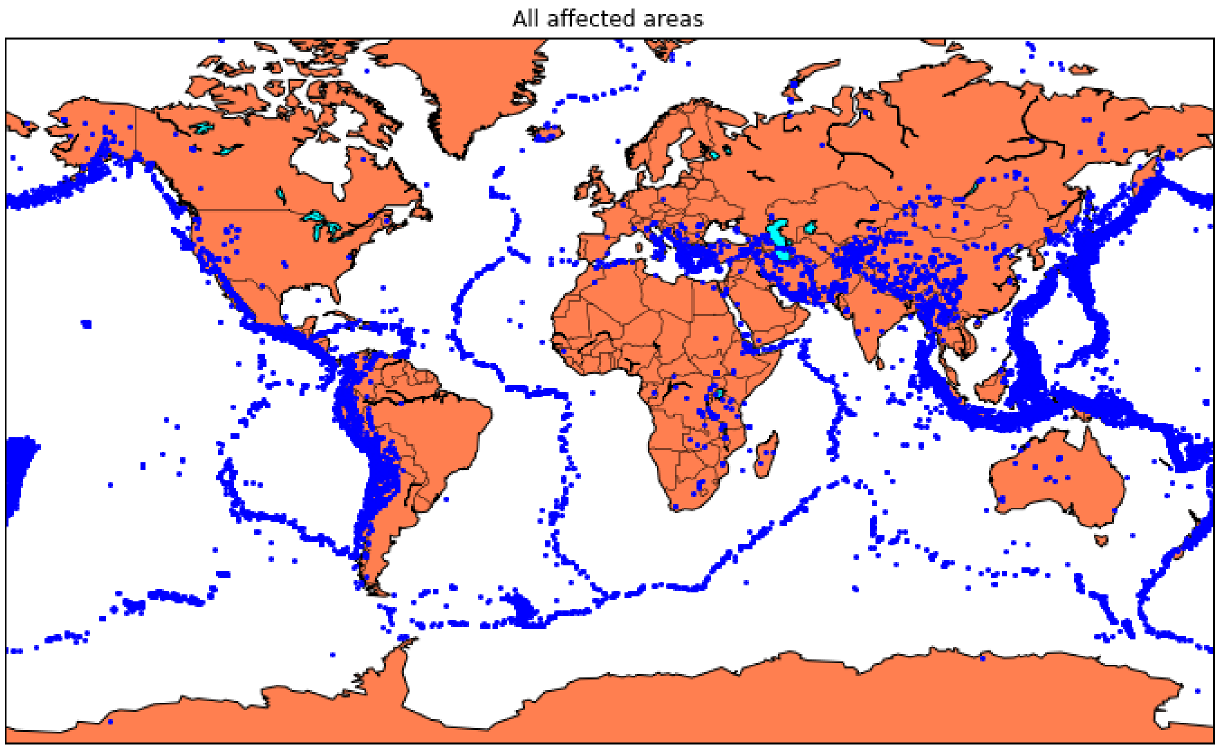
Visualization

Here, we will visualize the earthquakes that have occurred all around the world.

```
From mpl_toolkits.basemap import Basemap


M = Basemap(projection=’mill’,llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-
180,urcrnrlon=180,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)


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()
```

Output:



Splitting The Dataset:

Now we will split the dataset into a training and testing set.

```
X = final_data[[‘Timestamp’, ‘Latitude’, ‘Longitude’]]

Y = final_data[[‘Magnitude’, ‘Depth’]]
```

```
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)
```

Output:

Earthquake Prediction Using Machine Learning

We will be using the RandomForestRegressor model to predict the earthquake, here will look for its accuracy.

```
Reg = RandomForestRegressor(random_state=42)

Reg.fit(X_train, y_train)

Reg.predict(X_test)
```

Output:

Earthquake Prediction Using Machine Learning

```
Reg.score(X_test, y_test)

Reg.score(X_test, y_test)
```

Output:

Earthquake Prediction Using Machine Learning

86% of accuracy is quite high.

Now we will shift to GridSearch

```
From sklearn.model_selection import GridSearchCV
```

```
Parameters = {‘n_estimators’:[10, 20, 50, 100, 200, 500]}


Grid_obj = GridSearchCV(reg, parameters)

Grid_fit = grid_obj.fit(X_train, y_train)

Best_fit = grid_fit.best_estimator_

Best_fit.predict(X_test)
```

Output:

Earthquake Prediction Using Machine Learning

```
Best_fit.score(X_test, y_test)
```

Output:

Earthquake Prediction Using Machine Learning

Considering it’s a natural phenomenon, we have got a high accuracy number.

We will employ a neural network for predicting the earthquake.

Neural Network Model

A neural network model can be employed to forecast earthquakes by examining diverse elements and trends in seismic data. This model harnesses the capabilities of neural networks, which draw inspiration from the neural connections of the human brain, to analyze intricate data and reveal hidden relationships and patterns. By training the neural network on historical earthquake data, it can acquire the ability to identify precursor signals and patterns that indicate the probability of an upcoming earthquake.

```
From keras.models import Sequential

From keras.layers import Dense


Def create_model(neurons, activation, optimizer, loss):

   Model = Sequential()

   Model.add(Dense(neurons, activation=activation, input_shape=(3,)))
```

```
    Model.add(Dense(neurons, activation=activation))
    Model.add(Dense(2, activation='softmax'))

    Model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

    Return model

From keras.wrappers.scikit_learn import KerasClassifier

Model = KerasClassifier(build_fn=create_model, verbose=0)

# neurons = [16, 64, 128, 256]
Neurons = [16]
# batch_size = [10, 20, 50, 100]
Batch_size = [10]
Epochs = [10]
# activation = ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear', 'exponential']
Activation = ['sigmoid', 'relu']
# optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
Optimizer = ['SGD', 'Adadelta']
Loss = ['squared_hinge']

Param_grid = dict(neurons=neurons, batch_size=batch_size, epochs=epochs, activation=activation,
optimizer=optimizer, loss=loss)

Grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
Grid_result = grid.fit(X_train, y_train)

Print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Means = grid_result.cv_results_[‘mean_test_score’]

Stds = grid_result.cv_results_[‘std_test_score’]

Params = grid_result.cv_results_[‘params’]

For mean, stdev, param in zip(means, stds, params):

   Print(“%f (%f) with: %r” % (mean, stdev, param))
```

Output:

Earthquake Prediction Using Machine Learning

```
Model = Sequential()

Model.add(Dense(16, activation=’relu’, input_shape=(3,)))

Model.add(Dense(16, activation=’relu’))

Model.add(Dense(2, activation=’softmax’))

  Model.compile(optimizer=’SGD’, loss=’squared_hinge’, metrics=[‘accuracy’])

Model.fit(X_train, y_train, batch_size=10, epochs=20, verbose=1, validation_data=(X_test, y_test))
```

Output:

Earthquake Prediction Using Machine Learning

```
[test_loss, test_acc] = model.evaluate(X_test, y_test)

Print(“Evaluation result on Test Data : Loss = {}, accuracy = {}”
```

Conclusion

Understanding earthquakes and effectively responding to them remains a complex and challenging task, even with the latest technological advancements. However, leveraging the capabilities of machine learning can greatly enhance our comprehension of seismic events. By employing machine learning techniques to analyze seismic data, we can uncover valuable insights and patterns that contribute to a deeper understanding of earthquakes. These insights can subsequently inform more effective strategies for mitigating risks and responding to seismic events.

As we head towards the future, we might see new technologies that will precisely predict the place and time of the earthquake that will happen.