GRT INSTITUTE OF
ENGINEERING AND
TECHNOLOGY, TIRUTTANI - 631209
Approved by AICTE, New Delhi Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PHASE 4 – DEVELOPMENT PART 2

PROJECT TITLE

EARTHQUAKE PREDICTION MODEL USING PYTHON

COLLEGE CODE : 1103

SARANYA R

3rd yr, 5th sem

Reg no. : 110321106047

rajeshsaranya37@gmail.com

INTRODUCTION:

EARTHQUAKE PREDICTION

Earthquake prediction is a challenging problem requiring a great deal of interdisciplinary research. In principle, there are two major approaches to developing a methodology for earthquake forecasting. The first approach is to search for intricate cause-effect links between earthquakes and the events accompanying them, trying to establish some functional relation among quantitative characteristics describing the events. The second approach is to study the statistical data pertaining to earthquake occurrence in order to find out how preceding and accompanying events influence the frequency of earthquakes. These two approaches do not contradict each other and can be combined in a joint decision-making forecasting system. A corresponding flowchart for the decision-making process is presented in Figure 14-1. This decision-making process was applied in the Tokai District of eastern Japan and at Parkfield, California.

PROJECT OVERVIEW:

Countless dollars and entire scientific careers have been dedicated to predicting where and when the next big earthquake will strike. But unlike weather forecasting, which has significantly improved with the use of better satellites and more powerful mathematical models, earthquake prediction has been marred by repeated failure due to highly uncertain conditions of earth and its surroundings. Now, with the help of artificial intelligence, a growing number of scientists say changes in the way they can analyze massive amounts of seismic data can help them better understand earthquakes, anticipate how they will behave, and provide quicker and more accurate early warnings. This helps in hazzard assessments for many builders and real estate business for infrastructure planning from business perspective. Also many lives can be saved through early warning. This project aims a simple solution to above problem by predicting or forecasting likely places to have earthquake in next 7 days. For user-friendly part, this project has a web application that extracts live data updated every minute by USGS.gov and predicts next likely place world wide to get hit by an earthquake, hence a realtime solution is provided.

DATA SET:

Title	magnitude	Datetime	cdi	mmi	alert	Tsunami	
M 7.0 - 18 km SW of Malango, Solomon Islands	7	22-11-2022 02:03	8	7	green	1	
M 6.9 - 204 km SW of Bengkulu, Indonesia	6.9	18-11-2022 13:37	4	4	green	0	
M 7.0 -	7	########	3	3	green	1	
M 7.3 - 205 km ESE of Neiafu, Tonga	7.3	########	5	5	green	1	
M 6.6 -	6.6	########	0	2	green	1	
M 7.0 - south of the Fiji Islands	7	########	4	3	green	1	
M 6.8 - south of the Fiji Islands	6.8	########	1	3	green	1	
M 6.7 - 60 km SSW of Boca Chica, Panama	6.7	20-10-2022 11:57	7	6	green	1	
M 6.8 - 55 km SSW of Aguililla, Mexico	6.8	22-09-2022 06:16	8	7	yellow	1	

Earthquake Prediction:

It is well known that if a disaster has happened in a region, it is likely to happen there again. Some regions really have frequent earthquakes, but this is just a comparative quantity compared to other regions. So, predicting the earthquake with Date and Time, Latitude and Longitude from previous data is not a trend which follows like other things, it is natural occuring.

Import the necessary libraries required for buidling the model and data analysis of the earthquakes.

```
In [1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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

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

Read the data from csv and also columns which are necessary for the model and the column which needs to be predicted.

```
In [2]:
data = pd.read_csv("../input/database.csv")
data.head()
```

```
Out[2]:
```

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	Magnitude Eror	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Eror	Root Mean Square	ID	Source	Location Source	Magnitude Source	Status
0	01/02/1965	13:44:18	19.2466	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860706	ISCGEM	ISCGEM	ISCGEM	Automatic
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860737	ISCGEM	ISCGEM	ISCGEM	Automatic
2	01/05/1965	18:05:58	-20.5579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860762	ISCGEM	ISCGEM	ISCGEM	Automatic
3	01/08/1965	18:49:43	-59.0776	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860856	ISCGEM	ISCGEM	ISCGEM	Automatic
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860890	ISCGEM	ISCGEM	ISCGEM	Automatic

```
In [3]:
data.columns

Out[3]:
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth
Error',
       'Depth Seismic Stations', 'Magnitude', 'Magnitude Type',
       'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
       'Horizontal Distance', 'Horizontal Error', 'Root Mean Square',
'ID',
       'Source', 'Location Source', 'Magnitude Source', 'Status'],
      dtype='object')
```

Figure out the main features from earthquake data and create a object of that features, namely, Date, Time, Latitude, Longitude, Depth, Magnitude.

```
In [4]:
data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]
data.head()

Out[4]:
```

	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

Here, the data is random we need to scale according to inputs to the model. In this, we convert given Date and Time to Unix time which is in seconds and a numeral. This can be easily used as input for the network we built.

```
In [5]:
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')
```

```
In [6]:
timeStamp = pd.Series(timestamp)
data['Timestamp'] = timeStamp.values
```

```
In [7]:
final_data = data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

Out[7]:

	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

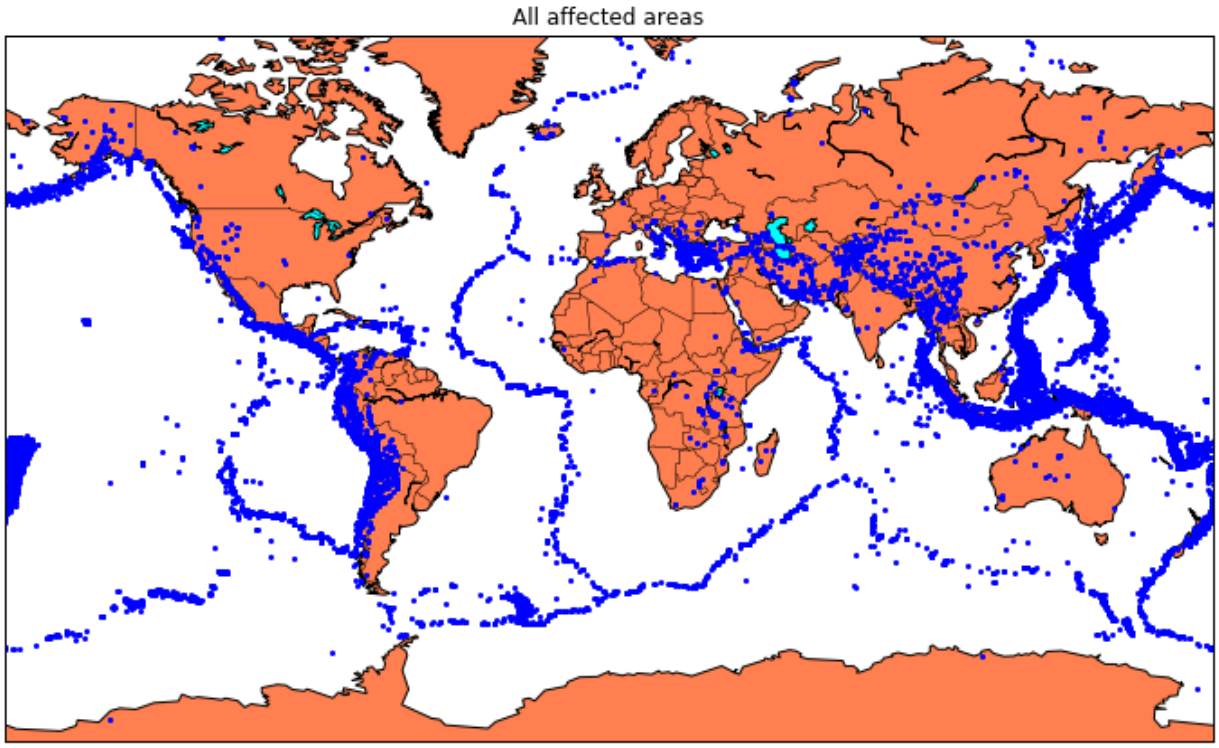
Visualization

Here, all the earthquakes from the database in visualized on to the world map which shows clear representation of the locations where frequency of the earthquake will be more.

```
In [8]:
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)

In [9]:
```

```
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()
```

```
/opt/conda/lib/python3.6/site-
packages/mpl_toolkits/basemap/__init__.py:1704:
MatplotlibDeprecationWarning: The axesPatch function was deprecated in
version 2.1. Use Axes.patch instead.
  limb = ax.axesPatch
/opt/conda/lib/python3.6/site-
packages/mpl_toolkits/basemap/__init__.py:1707:
MatplotlibDeprecationWarning: The axesPatch function was deprecated in
version 2.1. Use Axes.patch instead.
  if limb is not ax.axesPatch:
```



Splitting the Data

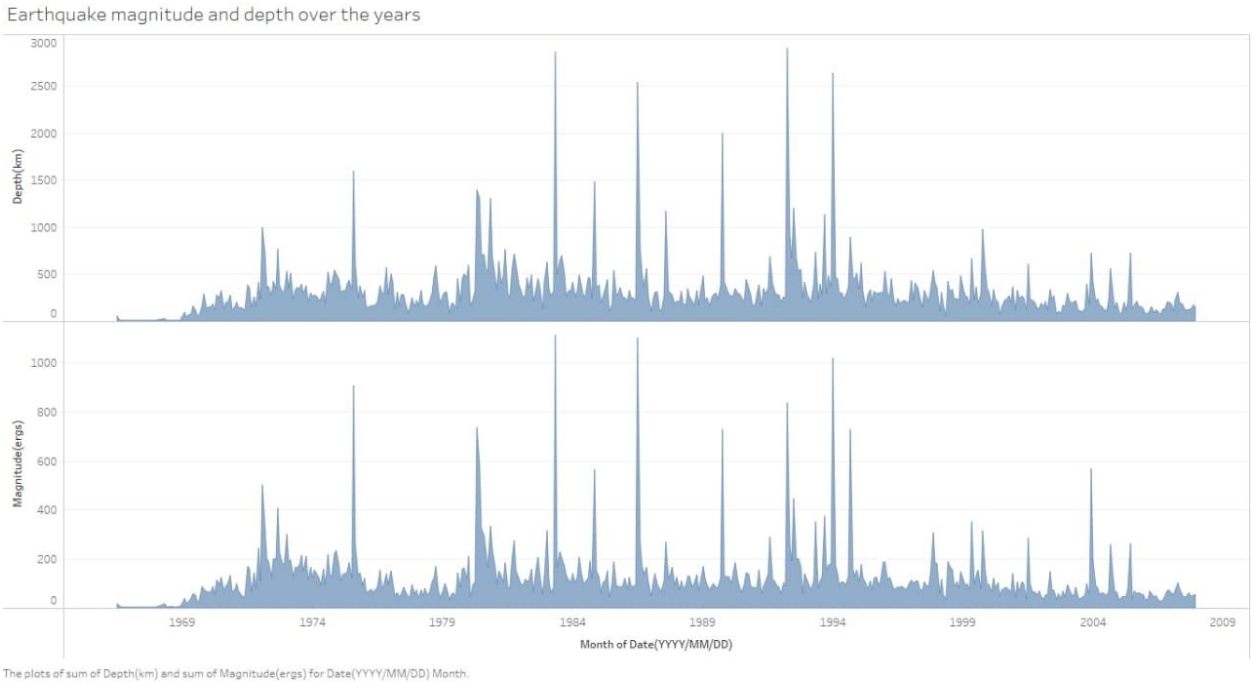
Firstly, split the data into Xs and ys which are input to the model and output of the model respectively. Here, inputs are TImestamp, Latitude and Longitude and outputs are Magnitude and DepEarthquake prediction: You can use this dataset to build a model that predicts when and where an earthquake might occur based on past earthquake data. You could use techniques such as time series analysis, clustering, or classification to identify patterns in the data and make predictions.

Magnitude prediction: You can use this dataset to build a model that predicts the magnitude of an earthquake based on other factors such as location, depth, or the number of seismic stations that recorded the earthquake. You could use regression techniques to build this model.

Risk assessment: You can use this dataset to identify areas that are at higher risk of earthquakes based on historical earthquake data. You could use clustering or classification techniques to identify patterns in the data and identify areas with similar characteristics.

Anomaly detection: You can use this dataset to detect anomalies or outliers in the data, which could represent earthquakes that are unusual or unexpected. You could use techniques such as clustering or classification to identify patterns in the data and detect anomalies.

Data visualization: You can use this dataset to create visualizations of earthquake data, which could help you identify patterns and relationships in the data. You could use techniques such as scatter plots, heat maps, or geographic information systems (GIS) to visualize the data.th. Split the Xs and ys into train and test with validation. Training dataset contains 80% and Test dataset contains 20%.




```
In [10]:
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)
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validation.py:41:
DeprecationWarning: This module was deprecated in version 0.18 in favor of
the model_selection module into which all the refactored classes and
functions are moved. Also note that the interface of the new CV iterators
are different from that of this module. This module will be removed in
0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Here, we used the RandomForestRegressor model to predict the outputs, we see the strange prediction from this with score above 80% which can be assumed to be best fit but not due to its predicted values.

```
In [12]:
from sklearn.ensemble import RandomForestRegressor

reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
reg.predict(X_test)
```

```
/opt/conda/lib/python3.6/site-
packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning:
numpy.core.umath_tests is an internal NumPy module and should not be
imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

```
Out[12]:
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]])
```

```
In [13]:
reg.score(X_test, y_test)
```

```
Out[13]:
0.8614799631765803
```

```
In [14]:
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)
```

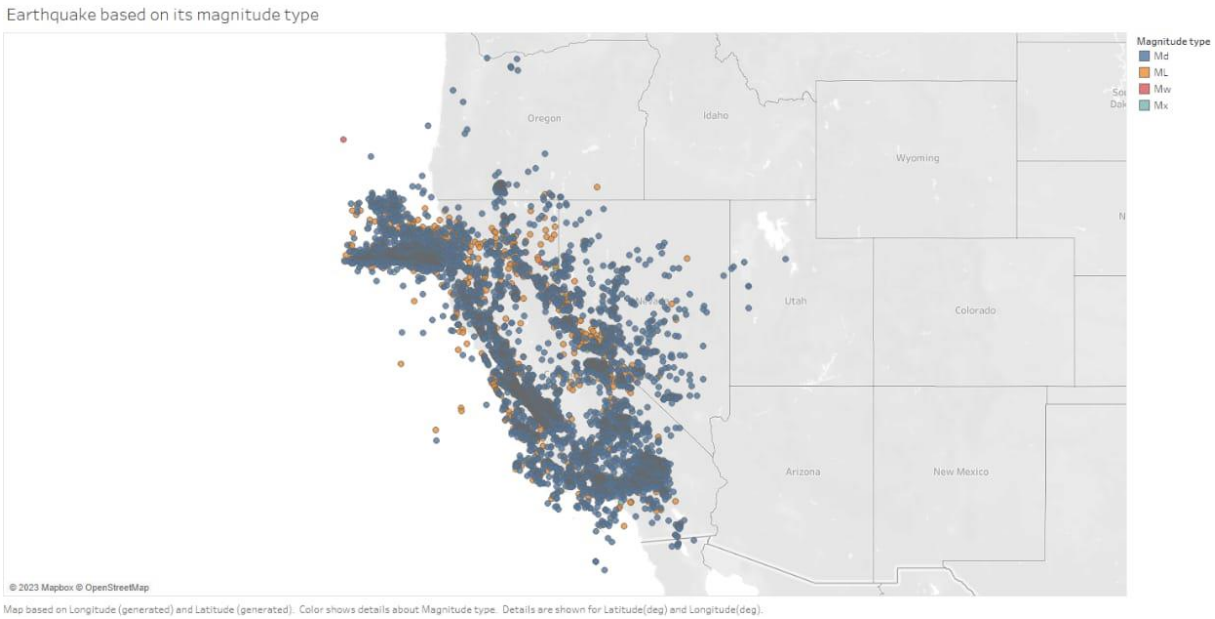
```
Out[14]:
array([[  5.8888 ,  43.532  ],
       [  5.8232 ,  31.71656],
       [  6.0034 ,  39.3312 ],
       ...,
       [  6.3066 ,  23.9292 ],
       [  5.9138 , 592.151  ],
       [  5.7866 ,  38.9384 ]])
```

```
In [15]:
best_fit.score(X_test, y_test)
```

```
Out[15]:
0.8749008584467053
```

Neural Network model

In the above case it was more kind of linear regressor where the predicted values are not as expected. So, Now, we build the neural network to fit the data for training set. Neural Network consists of three Dense layer with each 16, 16, 2 nodes and relu, relu and softmax as activation function.




```
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
```

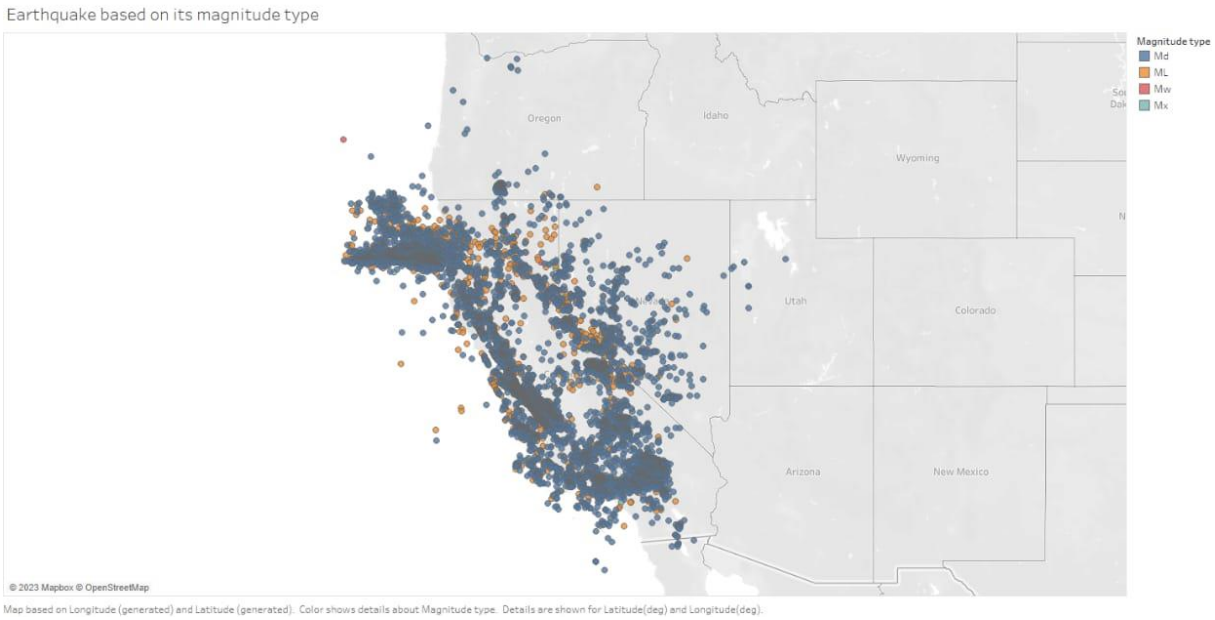
```
Using TensorFlow backend.
```

In this, we define the hyperparameters with two or more options to find the best fit.

```
In [17]:
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)
```

linkcode

Here, we find the best fit of the above model and get the mean test score and standard deviation of the best fit model.



CONCLUSION :

Predicting earthquakes accurately is an extremely challenging task, and it's important to note that there is no proven method for reliably predicting earthquakes. While some research exists on statistical and machine learning models for seismic data analysis, their predictive capabilities are limited to short-term forecasts and general trends. Earthquake prediction involves complex geological and seismological factors, and it's an active area of research but not yet fully predictable.

In Python, you can work with seismic data using libraries like ObsPy for data retrieval and processing, and scikit-learn for building predictive models. However, any predictions made should be approached with caution, understanding the limitations and uncertainties involved in earthquake prediction. It's crucial to rely on established seismic monitoring systems and follow the guidance of relevant authorities for any necessary preparedness and response measures.