NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

CBCPC09

OPERATING SYSTEM

PROJECT REPORT

EARTHQUAKE PREDICTION

Submitted by:

Adarsh singh 2020UEA6592

Kunal 2020UEA6603

COURSE INSTRUCTOR

Ms.Somiya Rani

Branch & sec: ECAM-2

Semester : 6th

INDEX

ABSTRACT

Per the statistics received from BBC, data varies for every earthquake occurred till date. Approximately, up to thousands are dead, about 50,000 are injured, around 1-3 Million are dislocated, while a significant amount go missing and homeless. Almost 100% structural damage is experienced. It also affects the economic loss, varying from 10 to 16 million dollars. A magnitude corresponding to 5 and above is classified as deadliest.The consequences of earthquakes are devastating and are not limited to loss and damage of living as well as non-living, but it also causes a significant amount of change-from surrounding and lifestyle to economic. Every such parameter is devoted to forecasting earthquakes. A couple of minutes' notice and individuals can act to shield themselves from damage and demise; can decrease harm and monetary misfortunes, and property, characteristic assets can be secured.

In the current scenario, an accurate forecaster is designed and developed, a system that will forecast the catastrophe. It focuses on detecting early signs of earthquakes by using machine learning algorithms. Systems are entitled to the basic steps of developing learning systems along with the life cycle of data science. Data-sets for the Indian sub-continental along with the rest of the World are collected from government sources. Pre-processing of data is followed by construction of a stacking model that combines Random Forest and Support Vector Machine Algorithms. Algorithms develop this mathematical model reliant on "training data-set". Model looks for pattern that leads to catastrophe and adapt to it in its building, so as to settle on choices and forecasts without being expressly customized to play out the task

INTRODUCTION

Earthquake's association with structural damage and loss of life is one that keeps on enduring and thus is the focal point of consideration for many fields, say, seismological research and environmental engineering yet not limited to these. Its significance is stretched out to human life too, to sustain and to survive. A prediction that can be accurate and relied on is a requisite for all the areas prone to disasters and as well as for locations that have less to none chances. It will get us ready for all the worst possible scenarios and for necessary measures as well that can be taken beforehand to solve the upcoming crisis. As the technology is evolving and helping humans for a better and a convenient lifestyle, the possibility of saving life is taken up with the help of efficient ML algorithms and Data Science to give accurate forecasts. Machine Learning is a subset of Artificial Intelligence. It permits the system to adapt to a behavior of a particular kind based on its own learning and possesses the ability to improve itself naturally solely from experience without any explicit programming, human mediation or help. Initialisation of a machine learning process starts with feeding an honest quality data-set to the algorithms, so as to build a ML prediction model. Algorithms perform knowledge discovery and statistical evaluation, determining patterns and trends in data. Selection of algorithms relies on data and on the task that requires automation.

Our target is foreseeing catastrophic events and improving the manner in which we react to them. Great forecasts and admonitions spare lives. A notice of an approaching calamity can be issued well ahead of time as it will help in reducing both death occurrence and structural loss.

ML algorithms construct two types of predictive models, Regression and Classification models. Each of them approaches data in a different way. Concerned system makes use of a

regression model whose core idea is forecasting a numerical value.

Anticipating a seismic event is viewed as an impossible phenomenon. It is a troublesome errand due to non-linearity of the event and unreliability in it yet the ability of ML algorithms to assemble prescient models has transformed it into a potential wonder.

Earthquake forecasts for the Indian subcontinent along with the rest of the World requires employing their earthquake catalog aka data-set. An earthquake catalog refers to a complete list of earthquake location, time, magnitude and depth that have happened in the past. Methodology relies on the sequence of these past earthquakes,recognising suitable, necessary and appropriate parameters, identifying patterns in these parameters and understanding correlations between actual earthquakes from the past so as to predict future occurrence.

Various Random Forest-Grid search CV ensemble models are studied, modeled.

METHODOLOGY

1.Data Acquisition

Data acquisition is the process for bringing data for production use either from source outside the system and into the system, or from data produced by the system. This is the underlying advance to start and alludes to gathering required information. We obtain required data sets from government provided website such as

USGS.gov (United States Geological Survey)

IMD.gov (India Meteorological Department)

Google Acquired Kaggle contains data-set collected from different agencies of different governments.

Importing required libraries

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

Reading CSV file with the help of pandas

```
raw_data = pd.read_csv("major_earthquake_data.csv")
raw_data.head(6)
```

[4]:

	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
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	NaN	NaN	NaN	NaN	NaN	NaN
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
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	NaN	NaN	NaN	NaN	NaN	NaN
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	NaN	NaN	NaN
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
5	01/10/1965	13:36:32	-13.405	166.629	Earthquake	35.0	NaN	NaN	6.7	MW	NaN	NaN	NaN	NaN	NaN	NaN

To see columns and dimensions of our dataset

[5]:

```
print(raw_data.shape)
print(raw_data.columns)
```

```
(23412, 21)
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')
```

2.Data Featuring

Real-World Data could be found as incorrect, invalid, out-of-range, off-base, impossible as well as missing data which influence the outcomes causing them to be deceiving, misleading and incorrect. Irrelevant and unreliable data can make pattern recognition and knowledge discovery in the training phase progressively troublesome.

We took only few features which are important for training our model

[7]:
```
main_data = raw_data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]
main_data.head()
```

[7]:

	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

Checking for any null value

[8]:
```
main_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23412 entries, 0 to 23411
Data columns (total 6 columns):
Date          23412 non-null object
Time          23412 non-null object
Latitude      23412 non-null float64
Longitude     23412 non-null float64
Depth         23412 non-null float64
Magnitude     23412 non-null float64
dtypes: float64(4), object(2)
memory usage: 1.1+ MB
```

```
main_data.describe()
```

[9]:

	Latitude	Longitude	Depth	Magnitude
count	23412.000000	23412.000000	23412.000000	23412.000000
mean	1.679033	39.639961	70.767911	5.882531
std	30.113183	125.511959	122.651898	0.423066
min	-77.080000	-179.997000	-1.100000	5.500000
25%	-18.653000	-76.349750	14.522500	5.600000
50%	-3.568500	103.982000	33.000000	5.700000
75%	26.190750	145.026250	54.000000	6.000000
max	86.005000	179.998000	700.000000	9.100000

Since our data contain date and time columns whose values are unique so we convert them to UNIX timestamp to train our model better

[10]:
```
import datetime
import time
timestamp = []
for d, t in zip(main_data['Date'],main_data['Time']):
    try:
        ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
        timestamp.append(time.mktime(ts.timetuple()))
    except ValueError:
        timestamp.append('Error')
```

Adding timestamp value columns in our data if its not null

[12]:
```
timeStamp = pd.Series(timestamp)
main_data['Timestamp'] = timeStamp.values
final_data = main_data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'Error']
final_data.head()
```

[12]:

	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

[13]:
```
final_data = main_data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'Error']
final_data.head()
```

[13]:

	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

3.Visualization of Data

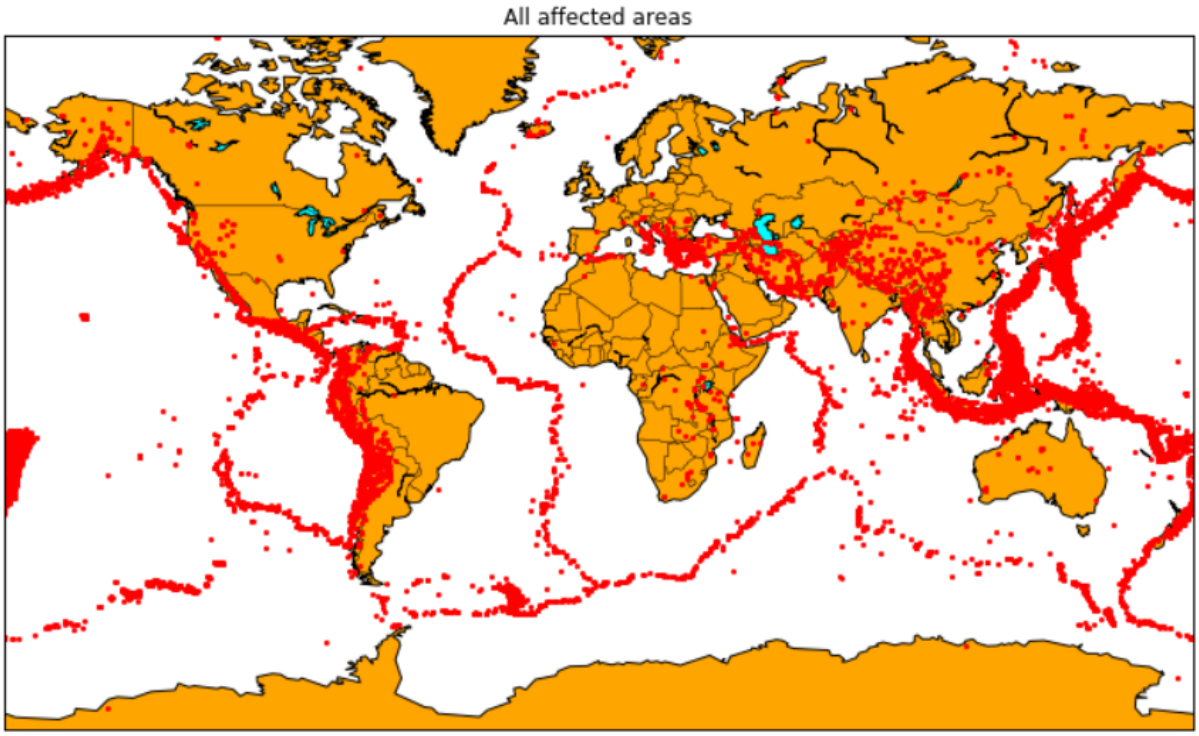
We plotted all the earthquake on world map

[25]:
```
from mpl_toolkits.basemap import Basemap

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

longitudes = final_data["Longitude"].tolist()
latitudes = final_data["Latitude"].tolist()
x,y = m(longitudes,latitudes)
```

[26]:
```
fig = plt.figure(figsize=(12,10))
plt.title("All affected areas")
m.plot(x, y, "o", markersize = 2, color = 'red')
m.drawcoastlines()
m.fillcontinents(color='orange',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```



From the figure it can be seen that most earthquake occurred near coastal region,

4.Spilliting of data into training and testing data

We used train_test_split to split our data into x and y, where x contains features like latitude,longitude and timestamp.y have target values for magnitude and depth of earthquake.

[12]:
```
x = final_data[['Timestamp', 'Latitude', 'Longitude']]
y = final_data[['Magnitude', 'Depth']]
print(x.shape)
print(y.shape)
```

```
(23409, 3)
(23409, 2)
```

[13]:
```
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, random_state=42)
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(18727, 3) (4682, 3) (18727, 2) (4682, 2)
```

5.Traning our Model

We are using RandomForestregressor to test and predict our output

[38]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

rf = RandomForestRegressor(random_state=42)
rf.fit(x_train, y_train)
rf_pred=rf.predict(x_test)
```

We used GridSearchCV to select the best hyperparameter for our model to improve its accuracy.

[28]:
```
from sklearn.model_selection import GridSearchCV

parameters = {'n_estimators':[10, 20, 50, 100, 200, 500]}

grid_obj = GridSearchCV(rf, parameters)
grid_fit = grid_obj.fit(x_train, y_train)
best_fit = grid_fit.best_estimator_
best_fit.predict(x_test)
```

[28]:
```
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 ]])
```

6.Evaluation of models

RandomForestregressor

```
print('Training score',rf.score(x_train,y_train))
print('Testing score',rf.score(x_test, y_test))
```

```
Training score 0.9728845600346301
Testing score 0.8614799631765803
```

GridSearchCV

[43]:
```
print(best_fit.score(x_train,y_train))
print(best_fit.score(x_test, y_test))
```

```
0.9811809977313877
0.8749008584467053
```

With GridsearchCV

7.Predicting Earthquake

We defined a function to convert our input date into unix time format

[40]:
```
from datetime import datetime
epoch = datetime(1970, 1, 1)

def mapdateTotime(x):
    try:
        dt = datetime.strptime(x, "%m/%d/%Y")
    except ValueError:
        dt = datetime.strptime(x, "%Y-%m-%dT%H:%M:%S.%fZ")
    diff = dt - epoch
    return diff.total_seconds()
```

After entering latitude, longitude and date we can predict magnitude and depth of earthquake

[47]:
```
latitude = float(input("Enter Latitude between -77 to 86:"))
longitude = float(input("Enter Longitude between -180 to 180:"))
date = input("Enter the date (Month/Day/Year format):")
Input_values = np.asarray([[mapdateTotime(date),latitude,longitude]],dtype=np.float32)
output_values=rf.predict(Input_values)
print('magnitude depth(km)',output_values)
```

We can also plot the location on a map to know better about the place.

[48]:

```
from mpl_toolkits.basemap import Basemap

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

longitudes = longitude
latitudes = latitude
x,y = m(longitudes,latitudes)
```

[49]:

```
fig = plt.figure(figsize=(12,10))
plt.title("on map")

m.plot(x, y, "o", markersize = 2, color = 'red')
m.drawcoastlines()
m.fillcontinents(color='yellow',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```

Conclusion

We are able to predict the magnitude and depth of earthquakes with latitude and longitude.

```
Enter Latitude between -77 to 86: 11.938
Enter Longitude between -180 to 180: 126.427
Enter the date (Month/Day/Year format): 01/08/2002
magnitude depth(km) [[ 5.86  31.433]]
```

```
Enter Latitude between -77 to 86: -24
Enter Longitude between -180 to 180: -120
Enter the date (Month/Day/Year format): 08/29/2021
magnitude depth(km) [[ 5.78  71.354]]
```

References

https://earthquake.usgs.gov/earthquakes/search/

https://riseq.seismo.gov.in/riseq/earthquake

https://www.kaggle.com/

https://link.springer.com/article/10.1007/s11069-016-2579-3

https://dl.acm.org/doi/abs/10.1145/3292500.3330919