ARTIFICIAL INTELLIGENCE

Earthquake prediction model using python

PHASE 3

Development part 2

Building the earthquake prediction model by loading and preprocessing the dataset.

Notebook link:https://colab.research.google.com/drive/1IHe_-veRrUX6y4RuHVhBIvX-_oGMLYa_?usp=sharing

LOADING THE DATASET

import pandas as pd
data = pd.read_csv('database.csv')

- Importing the dataset into a variable using built-in function in pandas package in python.

data.head()

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	...	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
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	...	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
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	...	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

5 rows × 21 columns

- The data is loaded an can be used by accessing the variable give ie.data

PREPROCESSING THE DATA

- The imported data should be preprocessed before using in to train the model.
- On analyzing the data we might come to know about the features in the dataset and the relationships between features in the dataset.

```
# Checking the Shape of the Dataset
data.shape
```

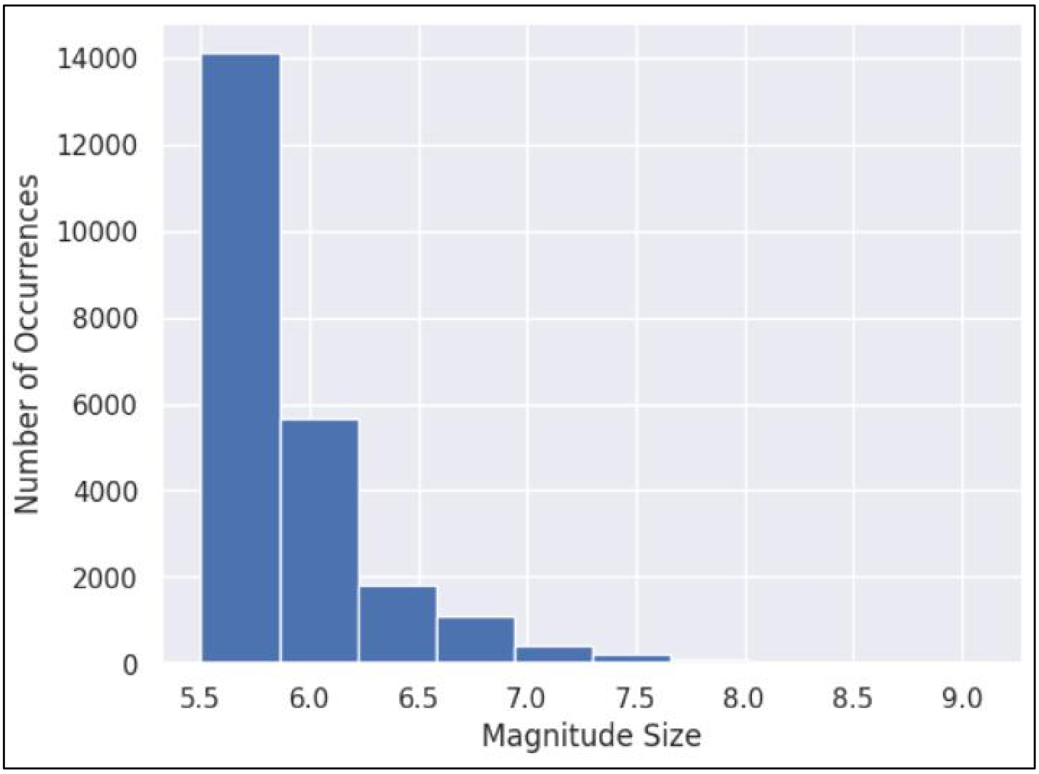
```
(23412, 21)
```

```
# Checking the Number of Entities
data.columns
```

```
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')
```

- Using the columns method we get the features in the dataset and with those details the features required for our need can be taken and used.

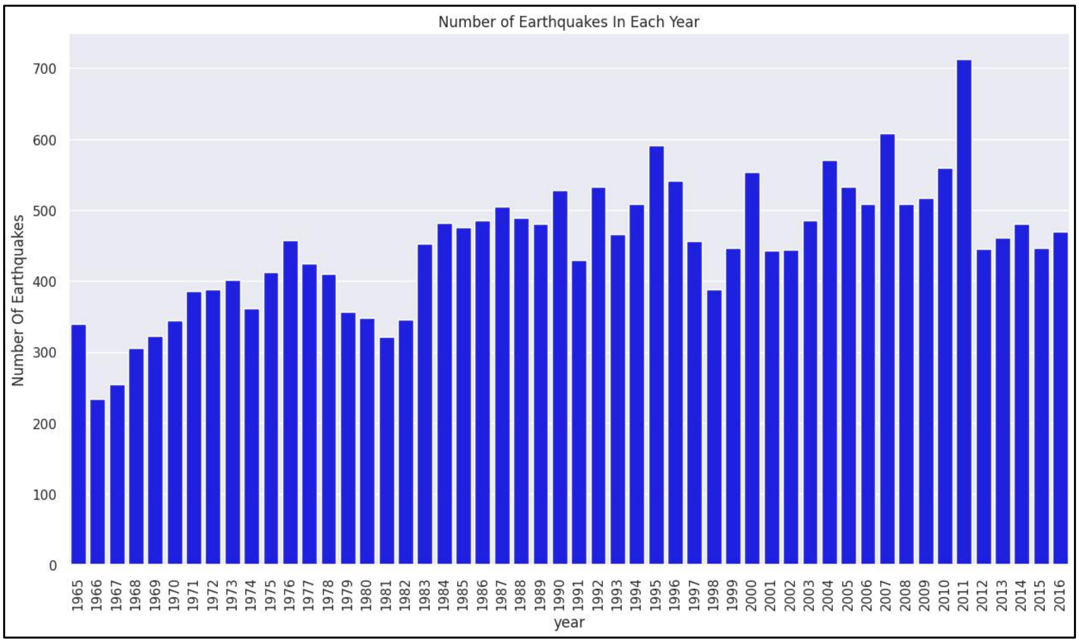
```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import seaborn as sns
plt.hist(data['Magnitude'])
plt.xlabel('Magnitude Size')
plt.ylabel('Number of Occurrences')
```



- On analysis on the magnitude feature of the dataset we get the number of occurrences that the earthquake occurred in past days.
- Using the basemap function in the mpl_toolkits package in python we can plot the occurrences of earthquake in real world map to visualize the distribution of the affected areas.
- With this we can analyse which part of the world is most affected and analysis of vulnerable areas can be spotted.

```
import datetime
data['date'] = data['Date'].apply(lambda x: pd.to_datetime(x))
data['year'] = data['date'].apply(lambda x: str(x).split('-')[0])
```

```
plt.figure(figsize=(15, 8))
sns.set(font_scale=1.0)
ax = sns.countplot(x="year", data=data, color = "blue")
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.ylabel('Number Of Earthquakes')
plt.title('Number of Earthquakes In Each Year')
```



- Upon another analysis with the date and the earthquake occurrence we come to know that the feature date is very closely related to the the event of earthquake occurrence.
- On the analysis of the dataset given we finally get that the features [‘Latitude’,’Longitude’,’Magnitude’,’Depth’] are the important features in the dateset.

```
tailored_data = data[['Latitude','Longitude','Magnitude','Depth']]
tailored_data.head()
```

	Latitude	Longitude	Magnitude	Depth
0	19.246	145.616	6.0	131.6
1	1.863	127.352	5.8	80.0
2	-20.579	-173.972	6.2	20.0
3	-59.076	-23.557	5.8	15.0
4	11.938	126.427	5.8	15.0

```
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:
        timestamp.append('ValueError')
tailored_data['Timestamp']=pd.Series(timestamp)
tailored_data= tailored_data[tailored_data.Timestamp!='ValueError']
tailored_data.dropna()
tailored_data.head()
```

- We need to add a new feature ie.Timestamp by combining the features Date and Time.
- The Timestamp data is a object datatype and is easier for processing using neural networks.

	Latitude	Longitude	Magnitude	Depth	Timestamp
0	19.246	145.616	6.0	131.6	-157630542.0
1	1.863	127.352	5.8	80.0	-157465811.0
2	-20.579	-173.972	6.2	20.0	-157355642.0
3	-59.076	-23.557	5.8	15.0	-157093817.0
4	11.938	126.427	5.8	15.0	-157026430.0