ARTIFICIAL INTELLIGENCE

Earthquake prediction model using python

PHASE 4

Development part 2

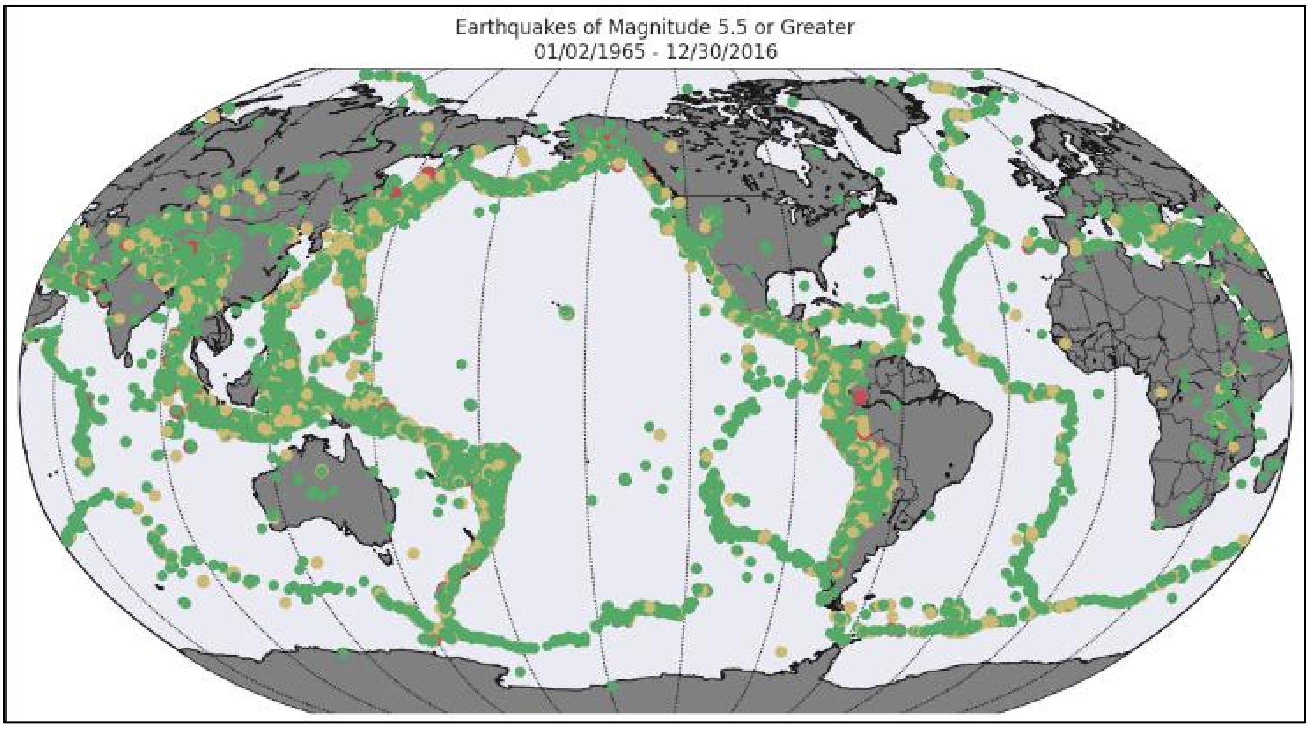
Visualizing the data on a world map
Splitting it into training and testing sets

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

DATA VISUALIZATION

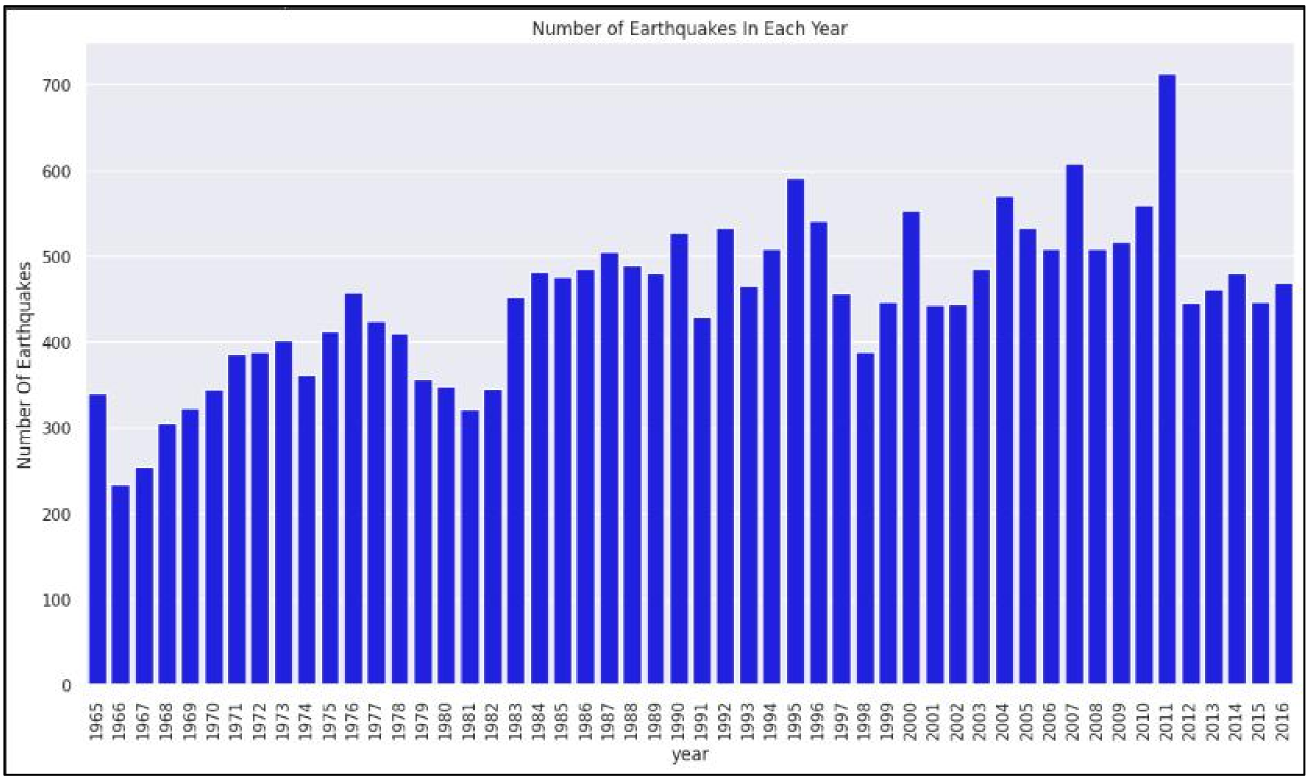
- Visualizing the data will help us to understand the dataset completely.
- With the given dataset we can explore the data by plotting the features in a bar chart or scatter plot.
- By mapping the Longitude and Latitude feature data in a world map we can easily see the distribution of the earthquake occurrence among the different countries in the world.

```
import pandas as pd
import numpy as np
data  = pd.read_csv('database.csv')
data.head()
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import seaborn as sns
def get_marker_color(magnitude):
    if magnitude < 6.2:
        return ('go')
    elif magnitude < 7.5:
        return ('yo')
    else:
        return ('ro')
plt.figure(figsize=(14,10))
eq_map = Basemap(projection='robin', resolution = 'l',lat_0=0, lon_0=-
130)
eq_map.drawcoastlines()
eq_map.drawcountries()
eq_map.fillcontinents(color = 'gray')
eq_map.drawmapboundary()
eq_map.drawmeridians(np.arange(0, 360, 30))
lons = data['Longitude'].values
lats = data['Latitude'].values
magnitudes = data['Magnitude'].values
timestrings = data['Date'].tolist()
min_marker_size = 0.5
for lon, lat, mag in zip(lons, lats, magnitudes):
    x,y = eq_map(lon, lat)
    msize = mag # * min_marker_size
    marker_string = get_marker_color(mag)
    eq_map.plot(x, y, marker_string, markersize=msize)
title_string = "Earthquakes of Magnitude 5.5 or Greater\n"
title_string += "%s - %s" % (timestrings[0][:10], timestrings[-1][:10])
plt.title(title_string)
plt.show()
```

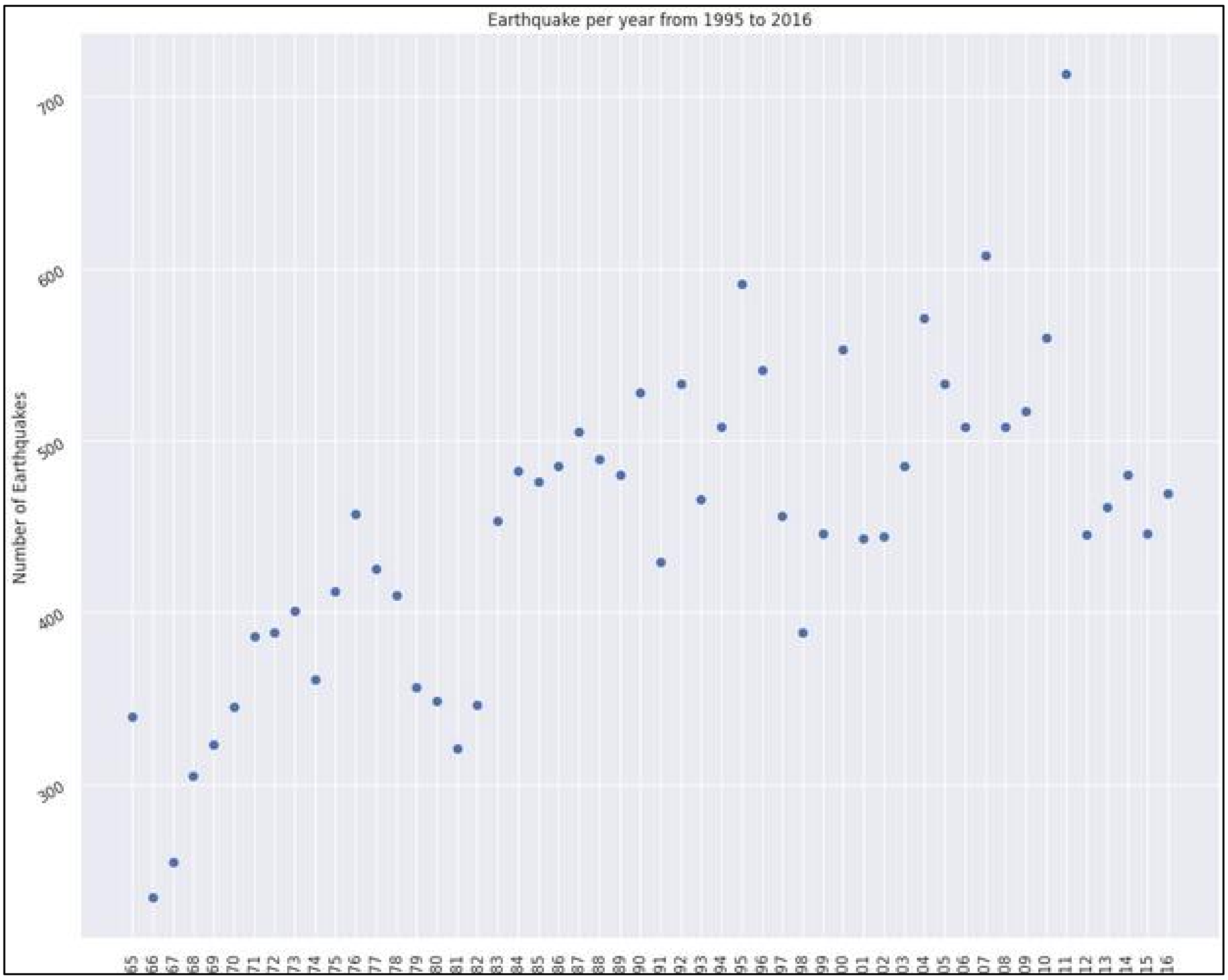


- The Date feature plays an important role in the occurences of the earthquake and plotting the bar graph for Year vs No. of occurrences shows the frequency of earthquake in each year.

```
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')
```



```
x = data['year'].unique()
y = data['year'].value_counts()
count = []
for i in range(len(x)):
    key = x[i]
    count.append(y[key])
#Scatter Plot
plt.figure(figsize =(15,12))
plt.scatter(x,count)
plt.title("Earthquake per year from 1995 to 2016")
plt.xlabel("Year")
plt.xticks(rotation=90)
plt.ylabel("Number of Earthquakes")
plt.yticks(rotation=30)
plt.show()
```



DATA SPLITTING FOR TRAINING AND TRAINING

- Among the important features we need to identify the features that can be used as input for the model and the features that can be used as a target/output for the model we are going to training.
- Once the input and output features are identified the data futher neeeded to be split into training data and testing data.
- We here split 80% of the data as training data and remainig 20% data as testing data which is used for validation part in the process.

```
x = tailored_data[['Longitude','Latitude','Timestamp','Depth']]
y = tailored_data[['Magnitude']]
from sklearn.model_selection import train_test_split as tts
x_train,x_test,y_train,y_test=tts(x,y,test_size=0.2,random_state =42)
```

- Here the x variable has the features that can be used as input for the model we are going to used.
- The Y variable has the feature that is used as a target for the model ie. which is predicted by the model.
- The spliiting is done by a built-in function in the sklearn.model_selection that is called as train_test_split().
- Using random state as a paramater for function will make sure that the same split for the data is used throughout the program even we run it again.