Earthquake prediction model-Data preprocessing and Visualisation

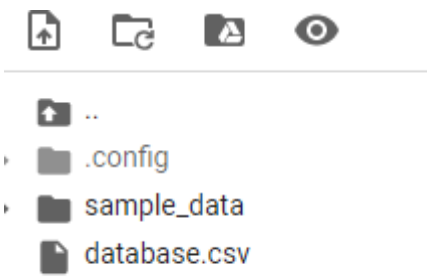
Introduction:

In this phase we need to start our project.So let us start to feed the data into the compiler and dtart preprocessing.

Data preprocessing:

Fed the data into the compiler:

Now let us the upload the data into the google colab.



Now import the pandas and read the file.As It is a csv file,we can use read_csv file.

```
import pandas as pd
df=pd.read_csv('database.csv')
print(df)
```

```
            Date      Time  Latitude  Longitude        Type   Depth  \
0      01/02/1965  13:44:18   19.2460   145.6160  Earthquake  131.60
1      01/04/1965  11:29:49    1.8630   127.3520  Earthquake   80.00
2      01/05/1965  18:05:58  -20.5790  -173.9720  Earthquake   20.00
3      01/08/1965  18:49:43  -59.0760   -23.5570  Earthquake   15.00
4      01/09/1965  13:32:50   11.9380   126.4270  Earthquake   15.00
...           ...       ...       ...        ...         ...     ...
23407  12/28/2016  08:22:12   38.3917  -118.8941  Earthquake   12.30
23408  12/28/2016  09:13:47   38.3777  -118.8957  Earthquake    8.80
23409  12/28/2016  12:38:51   36.9179   140.4262  Earthquake   10.00
23410  12/29/2016  22:30:19   -9.0283   118.6639  Earthquake   79.00
23411  12/30/2016  20:08:28   37.3973   141.4103  Earthquake   11.94

       Depth Error  Depth Seismic Stations  Magnitude Magnitude Type  ...  \
0              NaN                     NaN        6.0             MW  ...
1              NaN                     NaN        5.8             MW  ...
2              NaN                     NaN        6.2             MW  ...
```

Now let us see the what are the columns available in the dataset.

```
df.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', 'Timestamp'],
      dtype='object')
```

We need to not all the columns.We only need date, time, latitude, longitude, depth, magnitude.

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

To prepare the random data for input into the model, we need to standardize it. This involves converting the provided date and time into Unix time, which is represented in seconds as a numerical value. This Unix time format serves as a suitable input for the neural network we've constructed.

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

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

Missing Data:

Now we check whether any data is missing in the data.

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

	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

Now we check any nan values.

```
missing_values = data.isnull().sum()
print("Missing values in each column:")
print(missing_values)
nan_values = data.isna().sum()
print("Number of NaN values in 'Magnitude' column:", nan_values)
```

```
Missing values in each column:
Time         0
Latitude     0
Longitude    0
Depth        0
Magnitude    0
dtype: int64
Number of NaN values in 'Magnitude' column: Time         0
Latitude     0
Longitude    0
Depth        0
Magnitude    0
dtype: int64
```

As we can see there is no missing and nan values so we don’t need to do actions need to be taken for handling missing values.

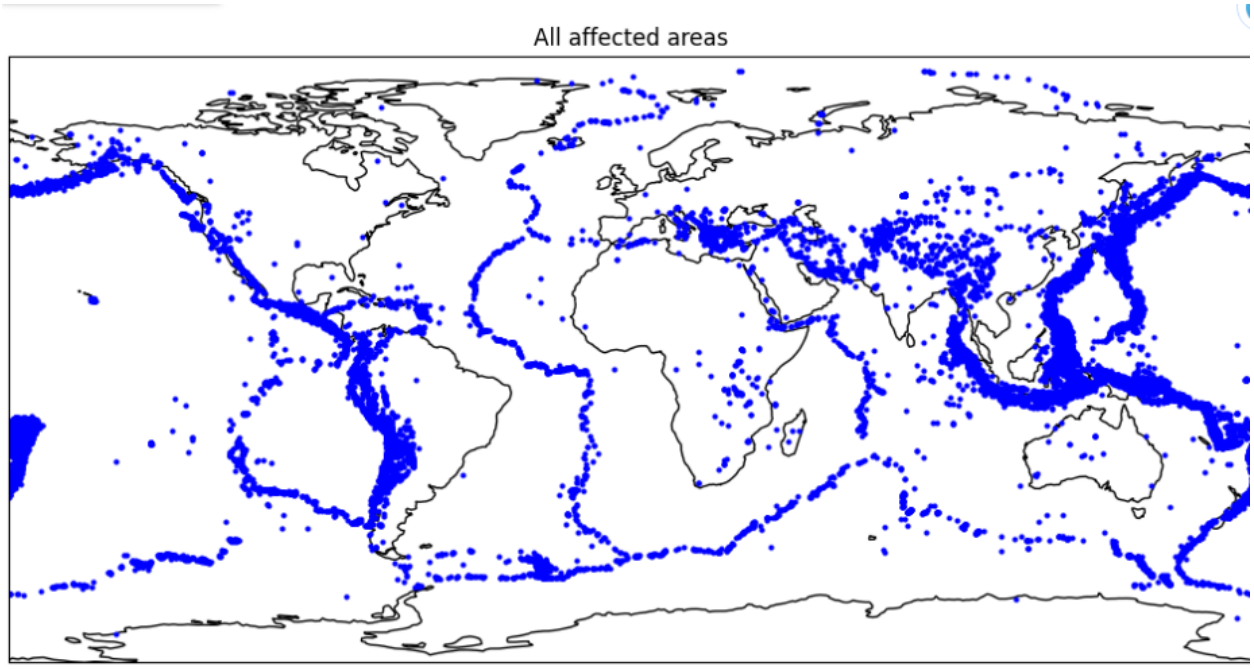
Data Visualization:

Data visualization is the practice of representing data graphically to gain insights and communicate information effectively. By creating charts, graphs, and other visual representations, complex data sets can be made more understandable, revealing patterns,

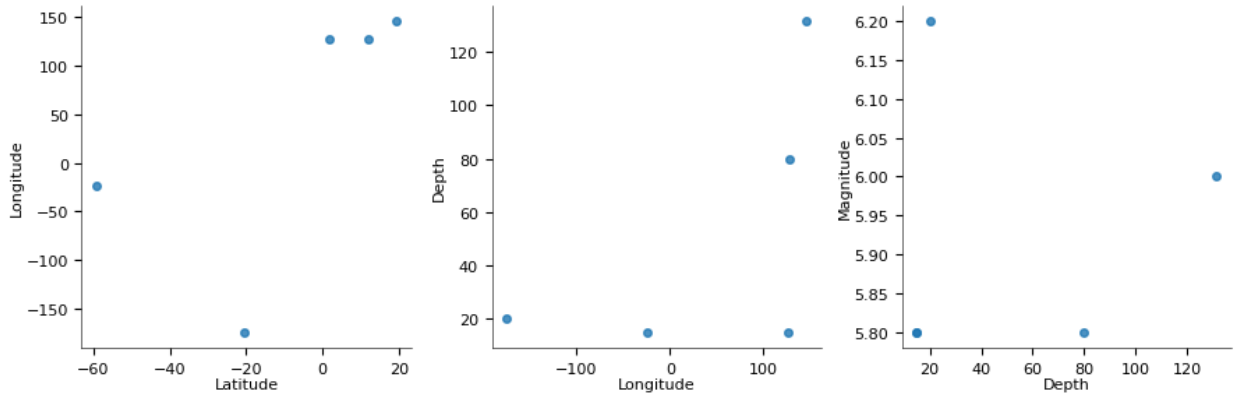
trends, and relationships, enabling informed decision-making and clear data storytelling.So now let us see where are the place earthqauke hits most.

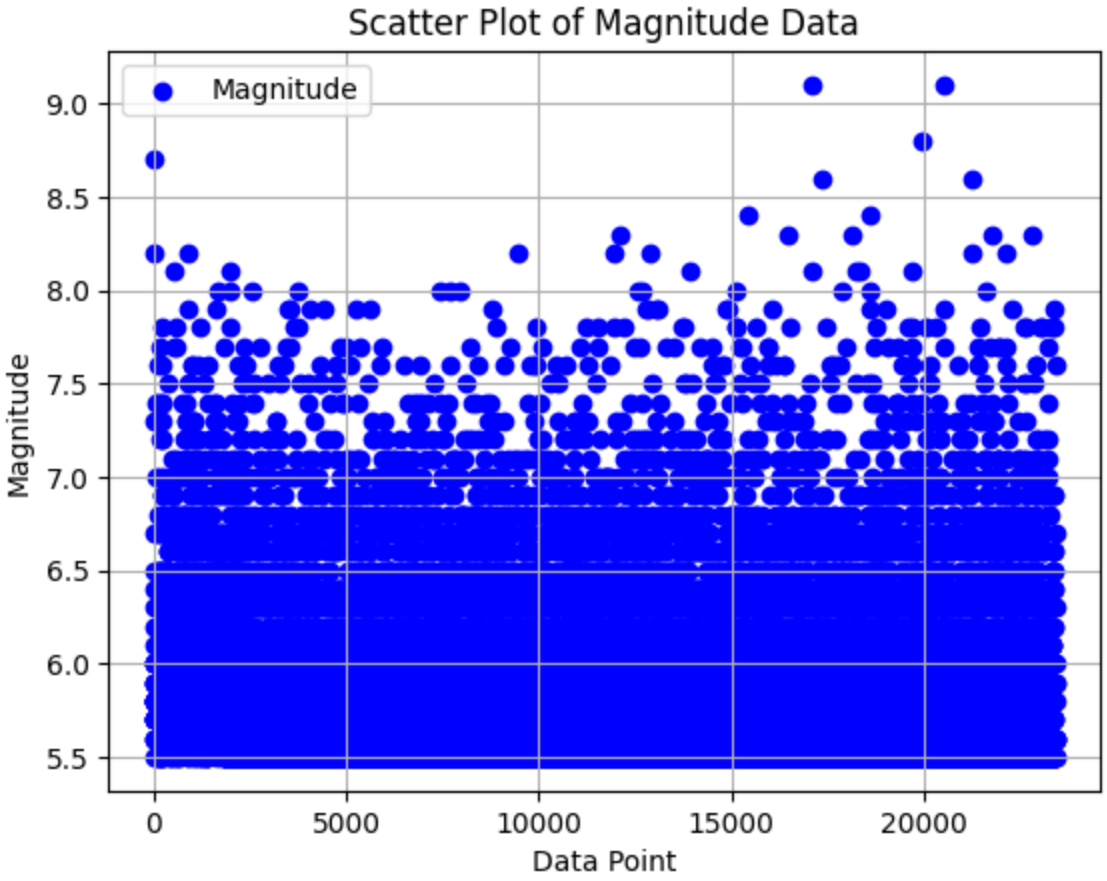
```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs

# Create a Cartopy projection
projection = ccrs.PlateCarree()

# Define the figure and axis
fig, ax = plt.subplots(subplot_kw={'projection': projection}, figsize=(12, 10))
ax.set_title("All affected areas")
longitudes = df["Longitude"].tolist()
latitudes = df["Latitude"].tolist()
# Plot the data on the map
ax.plot(longitudes, latitudes, 'bo', markersize=2, transform=projection)
```

We use cartopy instead basemap because, I don’t basemap and I cant download.Sad life bro.Jokes apart we use cartopy instead of basemap.Let us see which part of the world is most affected.

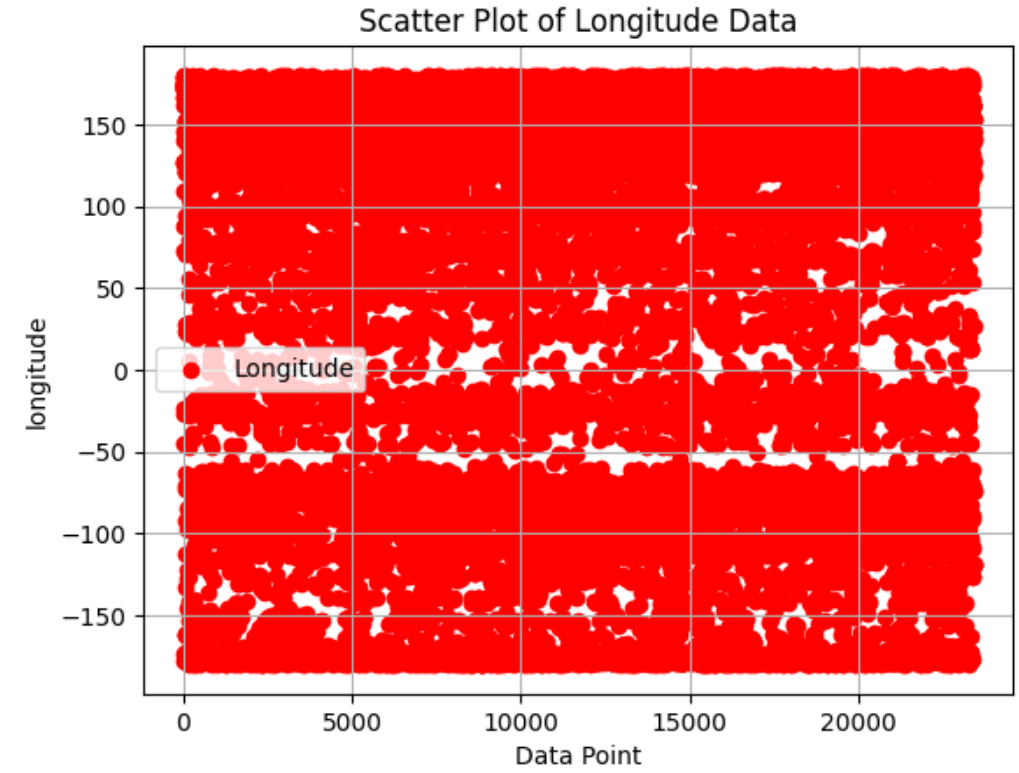


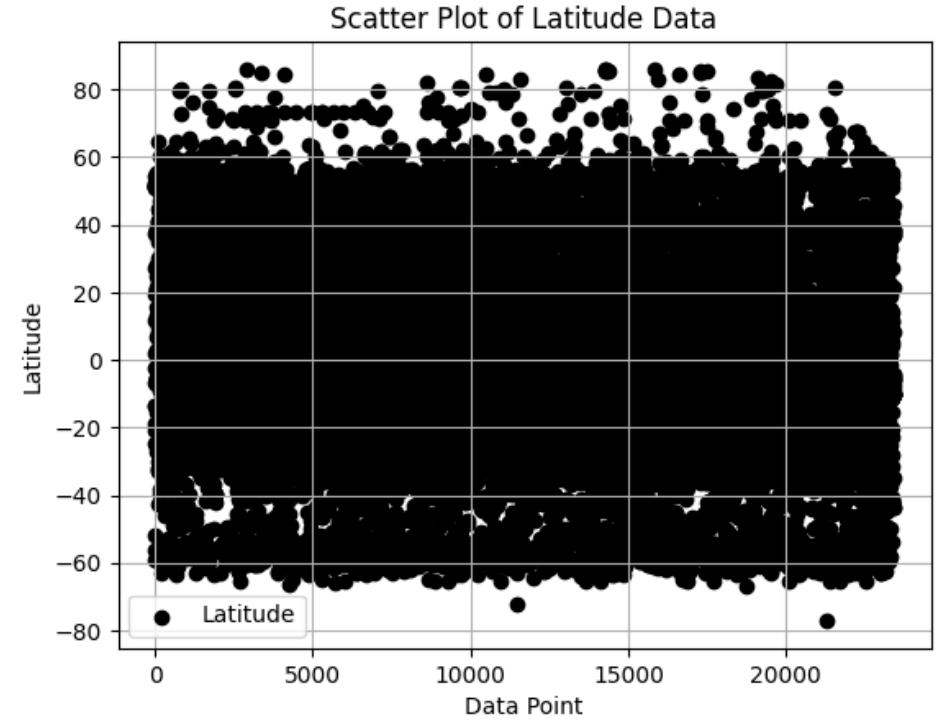
As we can see the east border of china,Russia and west border of North and south America is affected most.This is visual detail we can see from the data.let’s us see distribution of data.





We can see the data is clustered but we can see there are few points are scattered above the cluster .We may any details when we feed the data in the model.





We can see that data is clustered .It is not useful as the latitude and longitude are vary by a little bit.so we get many clustered points.

Conclusion:

As result here we separate the data column we needed and checked whther there is any missing values and visually displayes the data.Thus we completed Data preprocessing and now the data is now ready to be feed into the model.