Phase 3
Earthquake detection and prediction using python

Date	16 october 2023
Team ID	PROJ-212712_team_1
Project Name	Earthquake prediction model using python
Maximum Marks	

Data Visualization:

Data visualization provides a good, organized pictorial representation of the data which makes it easier to understand, observe, analyze. In this tutorial, we will discuss how to visualize data using Python.

Libraries used

Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs. In this tutorial, we will be discussing four such libraries.

- Matplotlib
- Seaborn
- Plotly

We will discuss these libraries one by one and will plot some most commonly used graphs.

Code:

libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px


df.describe()


Output:Code:

plt.subplots(figsize=(15,5))
plt.subplot(1,2,1)
sns.distplot(df['Depth'])
plt.show()
```

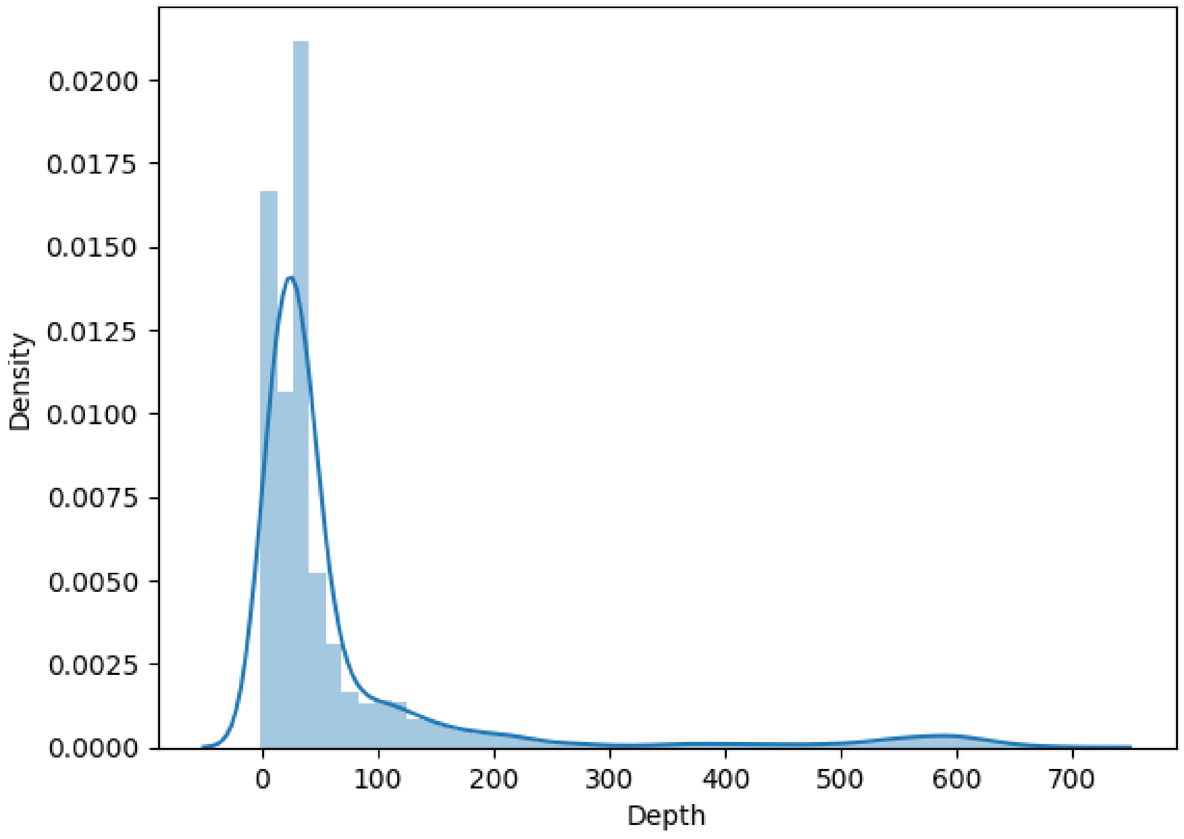
Matplotlib

Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility.

To import this type the below command in the terminal.

```
import matplotlib.pyplot as plt
```

Output:



Code:

```
plt.subplot(1,2,2)
sns.boxplot(df['Depth'])
plt.show()
```

Seaborn

Seaborn is a high-level interface built on top of the Matplotlib. It provides beautiful design styles and color palettes to make more attractive graphs.

To install seaborn type the below command in the terminal.

```
import seaborn as sns
```

Seaborn is built on the top of Matplotlib, therefore it can be used with the Matplotlib as well. Using both Matplotlib and Seaborn together is a very simple process. We just have to invoke the Seaborn Plotting function as normal, and then we can use Matplotlib's customization function.

Note: Seaborn comes loaded with dataset such as tips, iris, etc. but for the sake of this tutorial we will use Pandas for loading these datasets.

Scatter Plot

Scatter plot is plotted using the **scatterplot()** method. This is similar to Matplotlib, but additional argument data is required.

Line Plot

Line Plot in Seaborn plotted using the **lineplot()** method. In this, we can pass only the data argument also.

Bar Plot

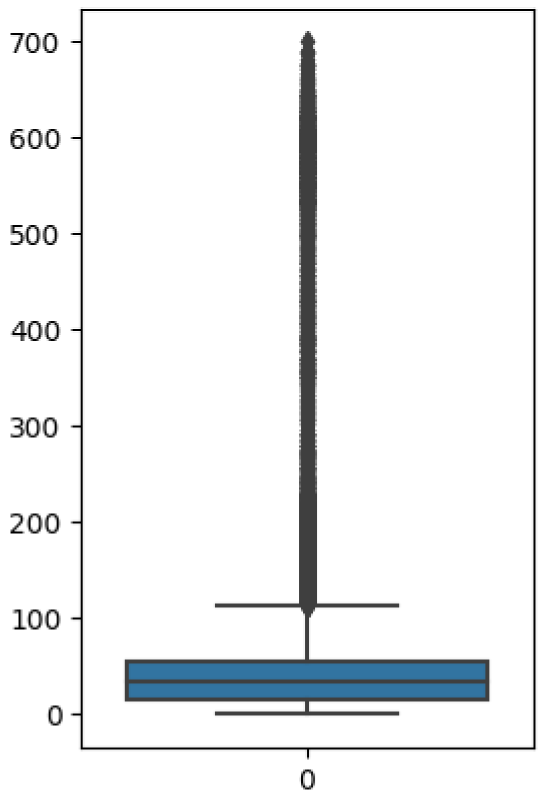
Bar Plot in Seaborn can be created using the **barplot()** method.

Histogram

The histogram in Seaborn can be plotted using the **histplot()** function.

After going through all these plots you must have noticed that customizing plots using Seaborn is a lot more easier than using Matplotlib. And it is also built over matplotlib then we can also use matplotlib functions while using Seaborn.

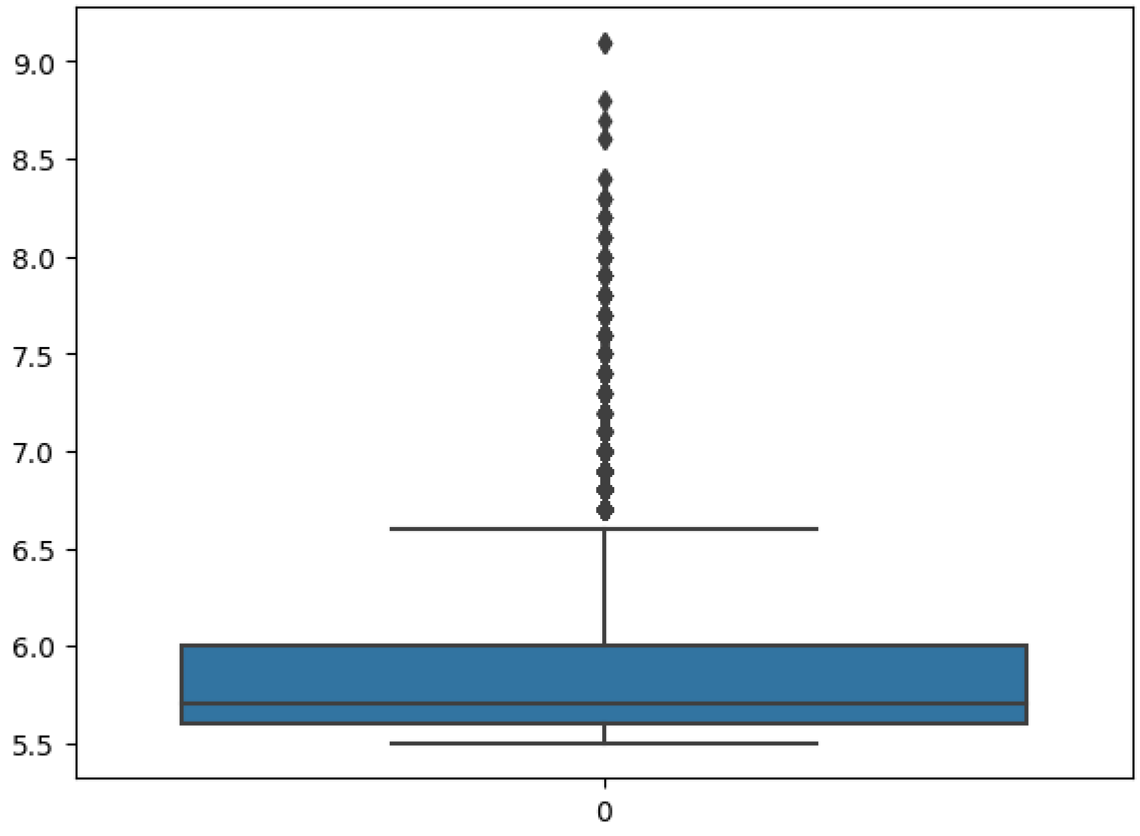
Output:



Code:

```
plt.subplots(figsize=(15,5))
plt.subplot(1,2,1)
sns.boxplot(df['Magnitude'])
plt.show()
```

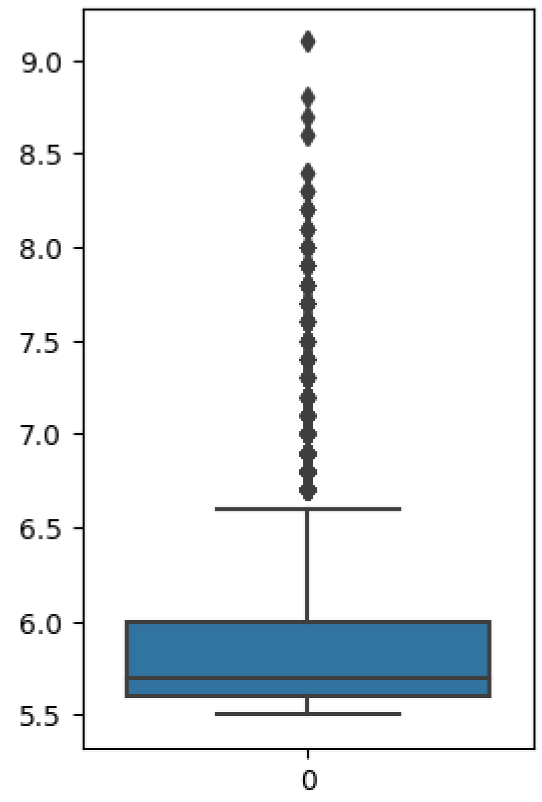
Output:



Code:

```
plt.subplot(1,2,2)
sns.boxplot(df['Magnitude'])
plt.show()
```

Output:



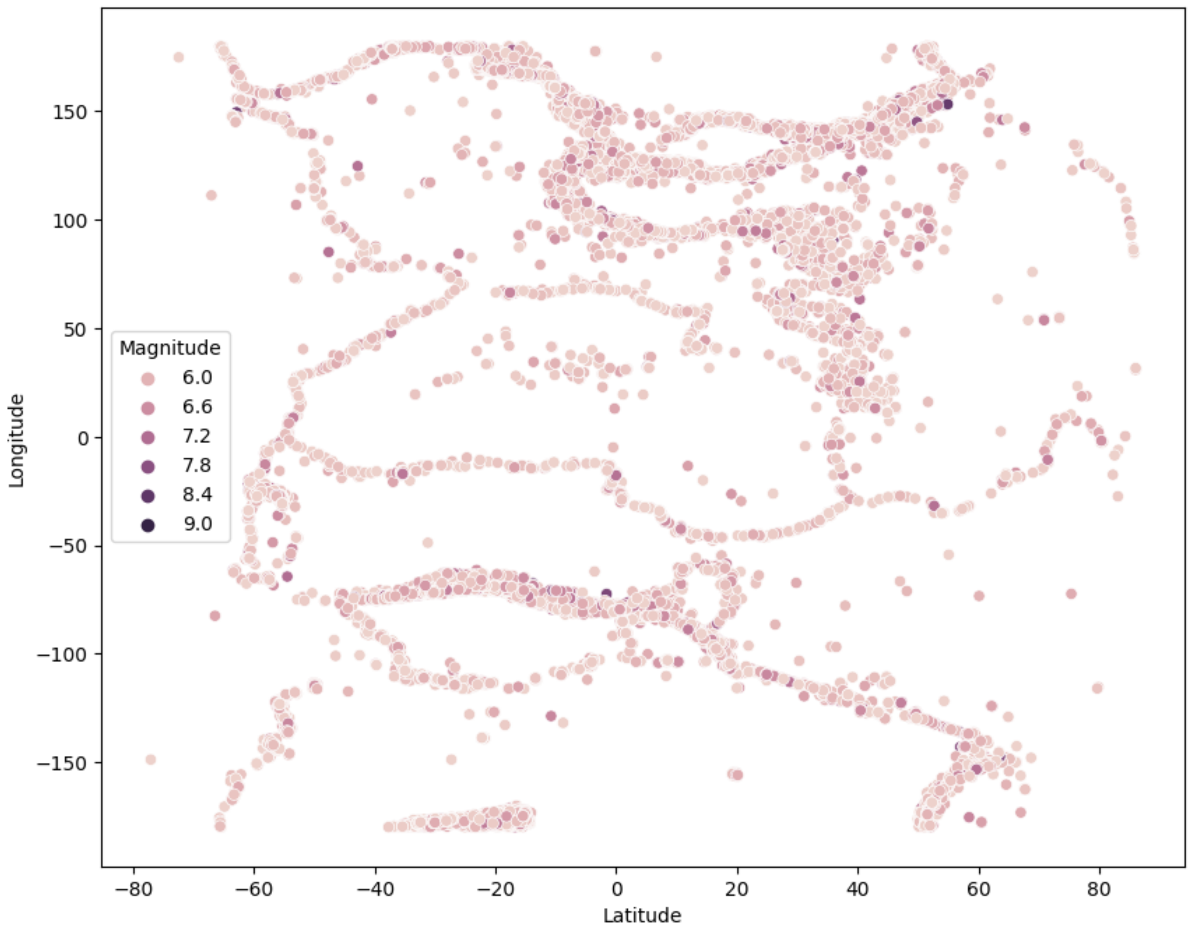
Code:

```
plt.figure(figsize=(10,8))
sns.scatterplot(data=df,x='Latitude',y='Longitude',hue=
'Magnitude')
plt.show()
```

Scatter Plot

Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them. The **scatter()** method in the matplotlib library is used to draw a scatter plot.

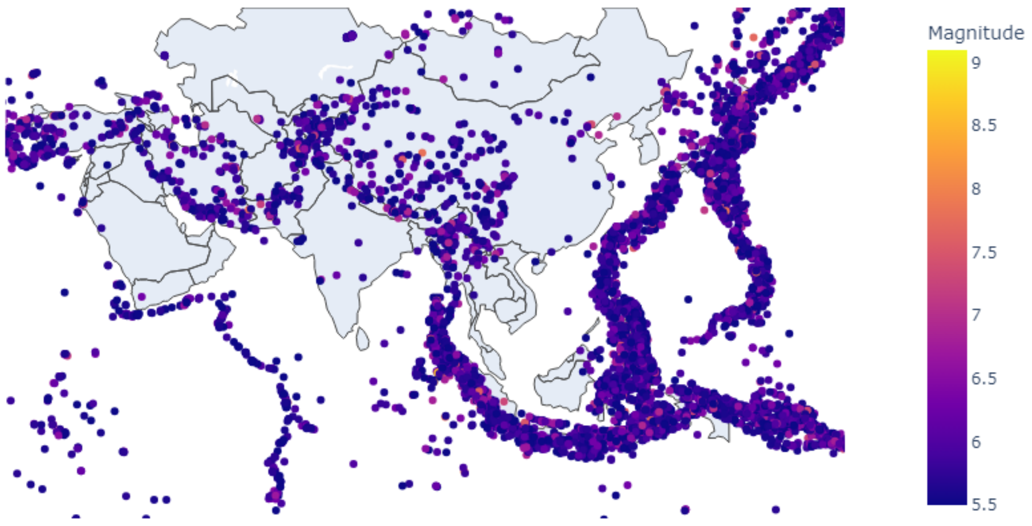
Output:



Code:

```
fig=px.scatter_geo(df,lat='Latitude',lon='Longitude',co
lor='Magnitude',scope='asia')
fig.show()
```

Output:



Code:

```
fig1=plt.figure()
x1=fig1.add_axes([.1,.1,2,1])
x1.plot(df['Magnitude'])
```

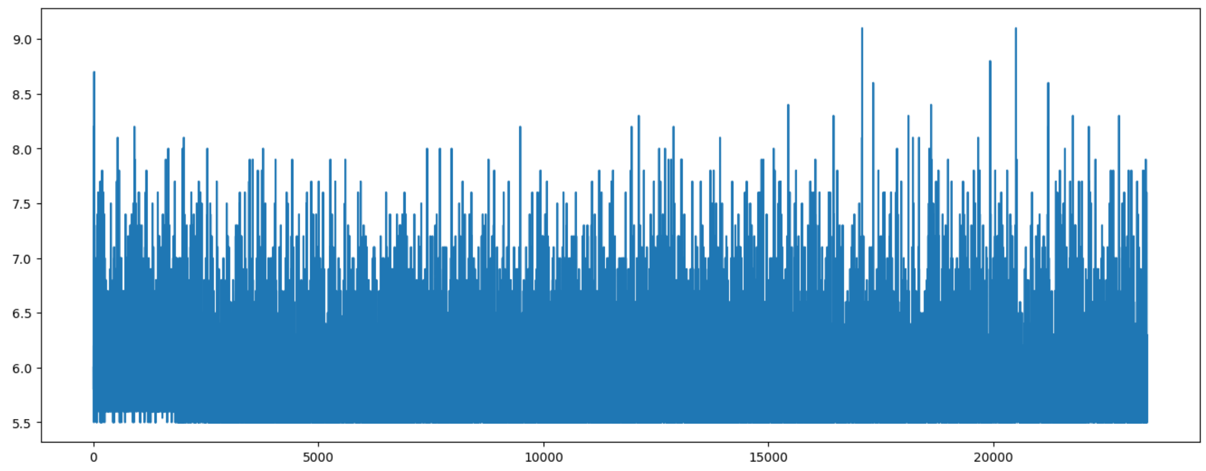
Line Chart

Line Chart is used to represent a relationship between two data X and Y on a different axis. It is plotted using the **plot()** function. Let's see the below example.

Bar Chart

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. It can be created using the **bar()** method.

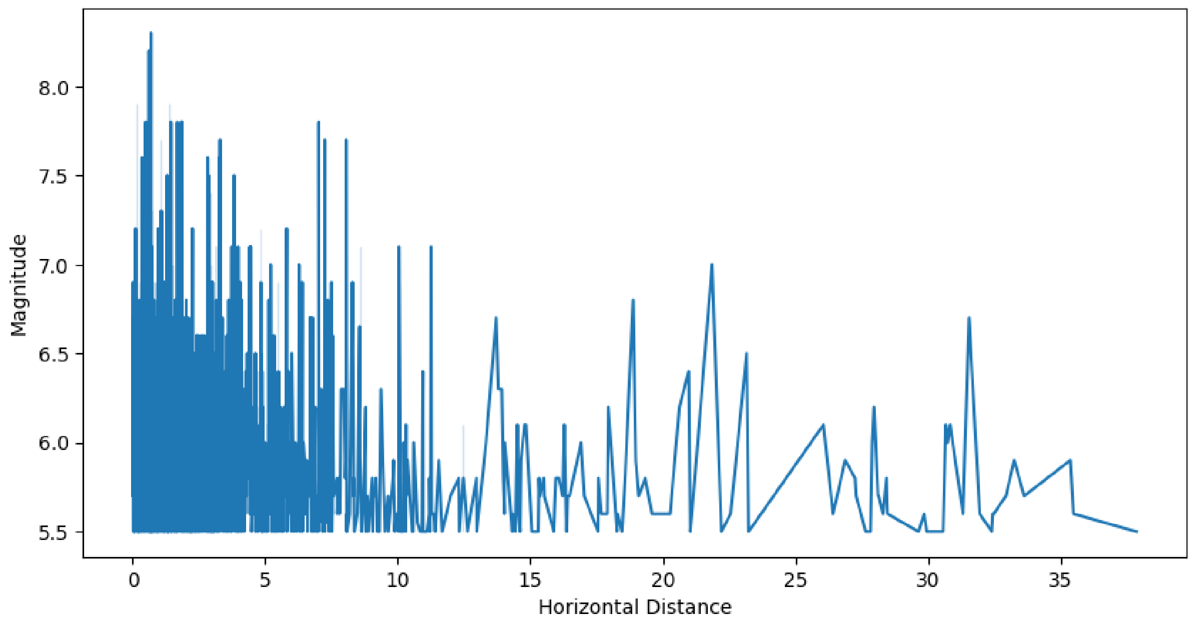
Output:



Code:

```
plt.figure(figsize=(10,5))
sns.lineplot(data=df,x='Horizontal
Distance',y='Magnitude')
plt.show()
```

Output:



Conclusion:

Good dataset visualization should communicate a data set clearly and effectively by using graphics. The best visualizations make it easy to comprehend data at a glance.Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.Data visualization allows business users to gain insight into their vast amounts of data. It benefits them to recognize new patterns and errors in the data. Making sense of these patterns helps the users pay attention to areas that indicate red flags or progress. This process, in turn, drives the business ahead.