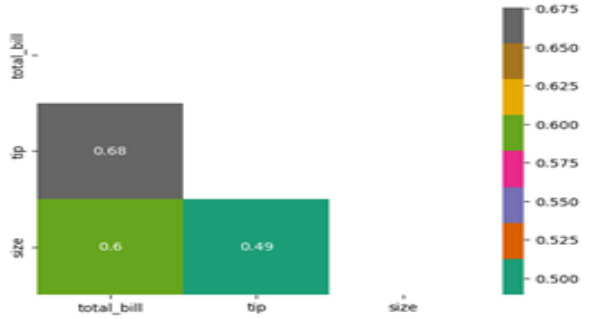
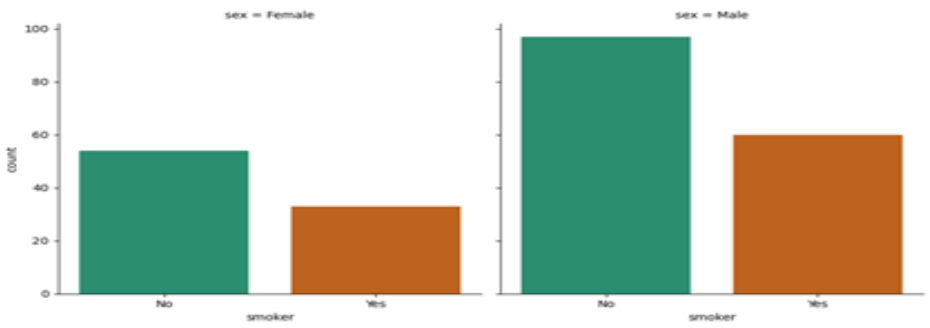
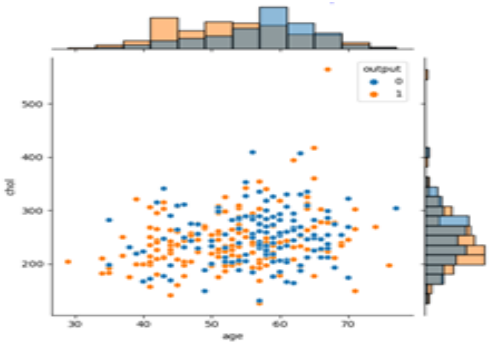
ML Visualizations

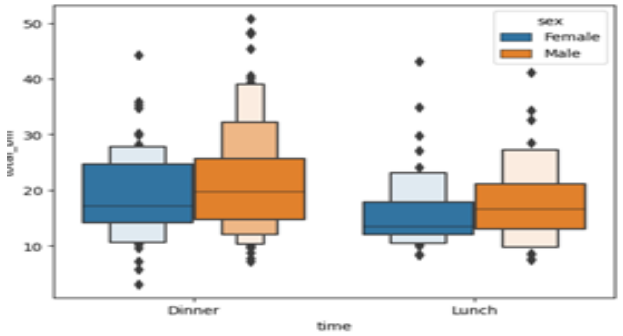
Himanshu K. Gajera
Department of Computer Science & Engineering
Pandit Deendayal Energy University, Gandhinagar

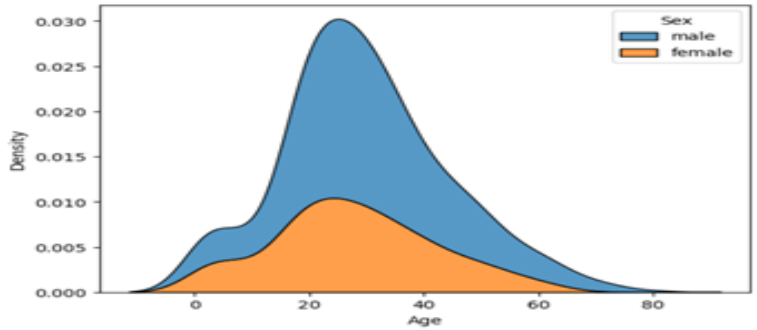
Introduction

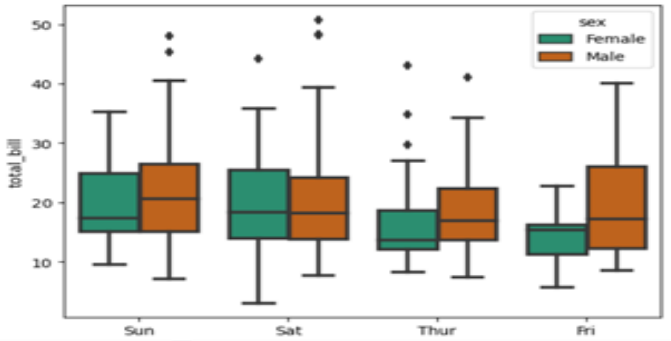


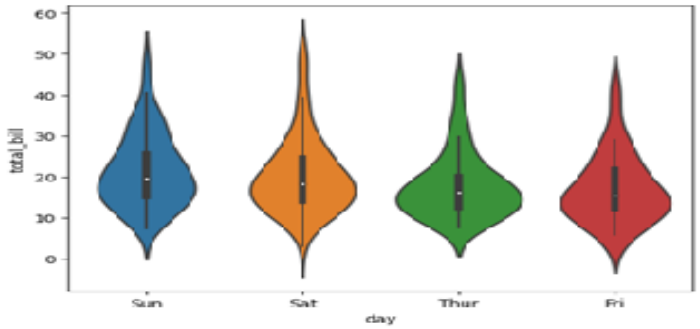


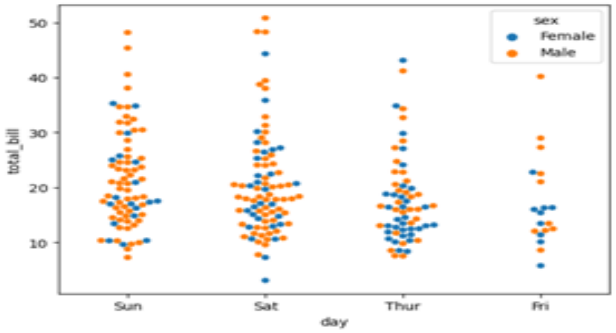


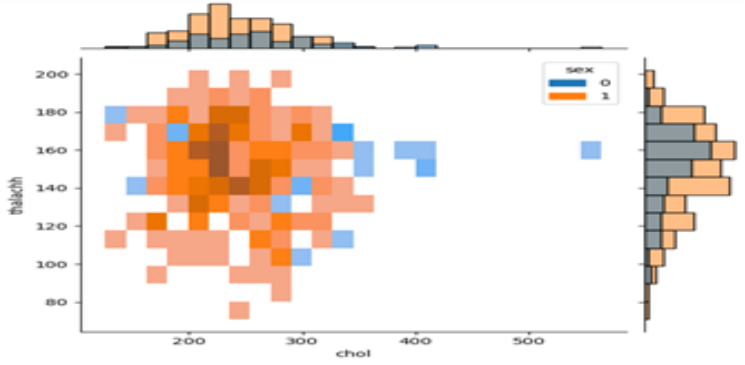








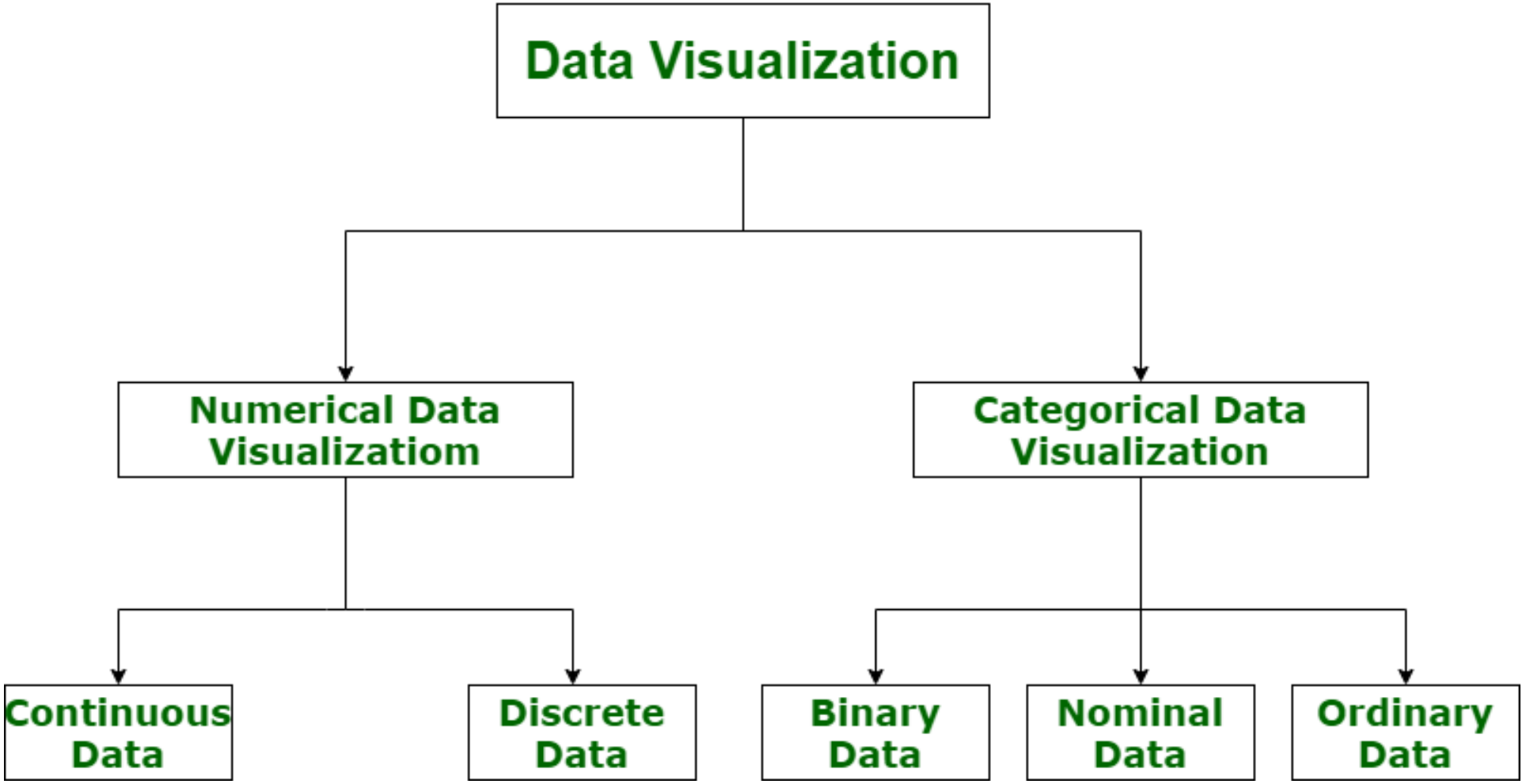




Introduction

- Data Visualization turns data into images that nearly anyone can understand making them invaluable for explaining the significance of digits to people who are more visually oriented.
- Not every time the numbers will sound meaningful to people working with data.
- This is where Data Visualization comes in.
- It is a technique of encoding those numbers into images which can be much more helpful to gain meaningful insights.
- It is one of the essential steps in every Data Science process.

Categories of Data Visualization



Top Data Visualization Tools

1. Tableau
2. Looker
3. Zoho Analytics
4. Sisense
5. IBM Cognos Analytics
6. Qlik Sense
7. Domo
8. Microsoft Power BI
9. Klipfolio
10. SAP Analytics Cloud

Top Data Visualization Libraries Available in Python, R, and Javascript

Python:

1. Matplotlib
2. Plotly
3. ggplot
4. Seaborn
5. Altair
6. Geoplotlib
7. Bokeh

R:

1. ggplot2
2. Plotly
3. Leaflet
4. Esquisse
5. Lattice

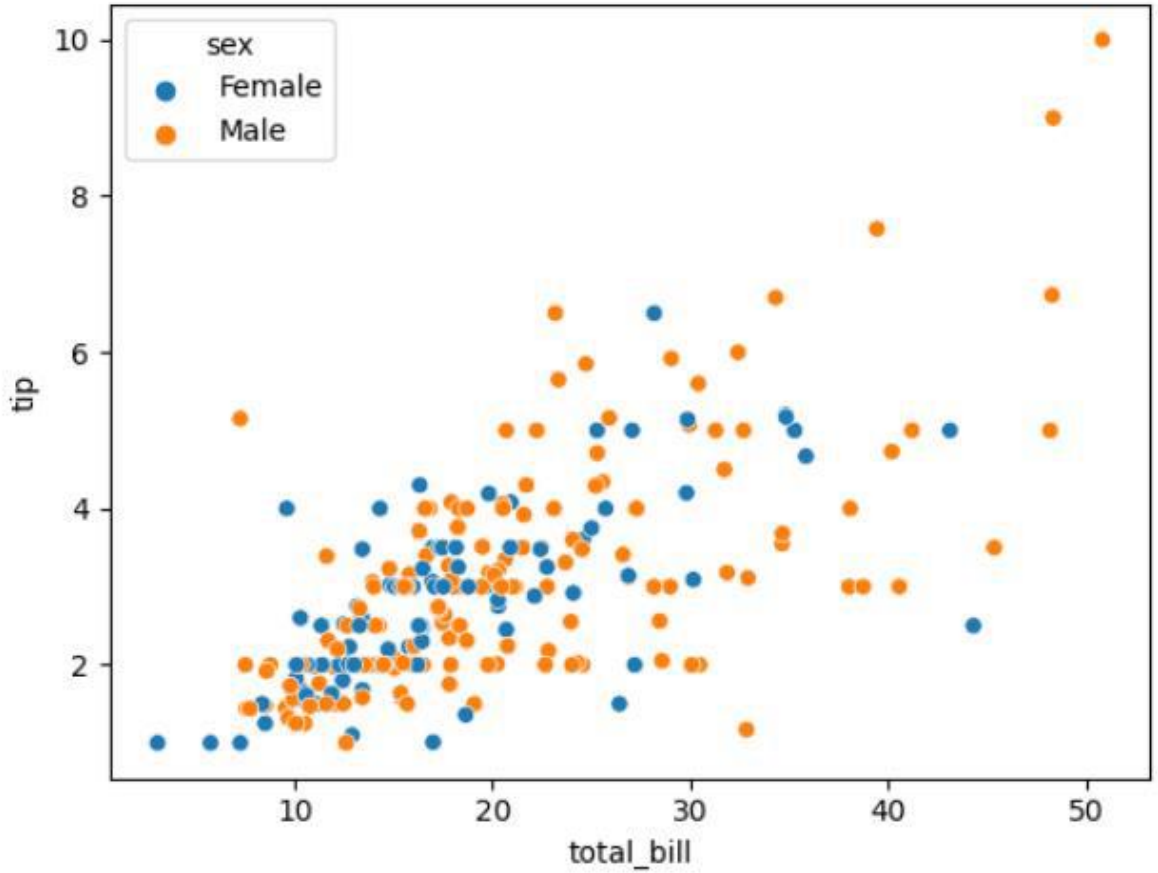
JavaScript:

1. D3.js
2. Chart.js
3. Plotly

Scatterplot

- This is used to find a relationship in a bivariate data. It is most commonly used to find correlations between two continuous variables.

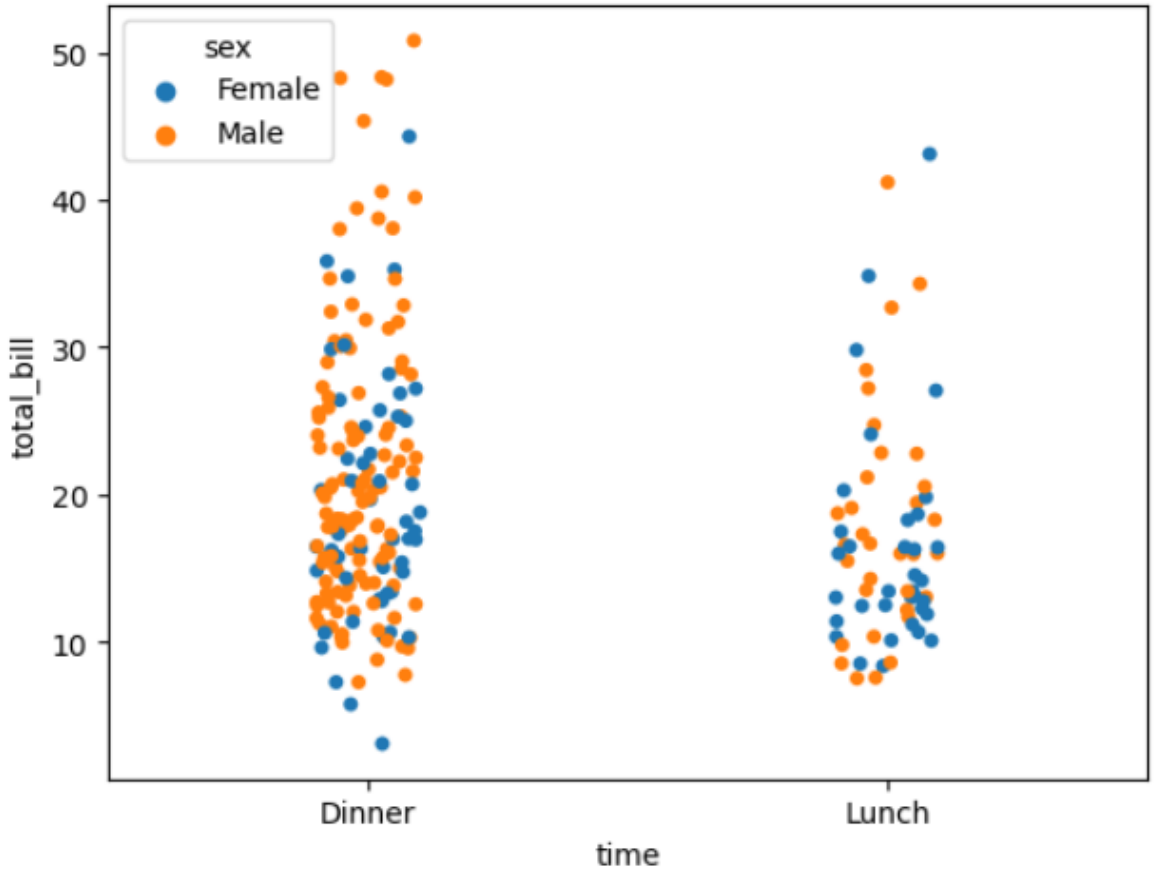
```
sns.scatterplot(x = 'total_bill', y = 'tip', hue = 'sex', data = tips);
```



Stripplot

- A strip plot is a single-axis scatter plot that is used to visualize the distribution of many individual one-dimensional values. The values are plotted as dots along one unique axis, and the dots with the same value can overlap.

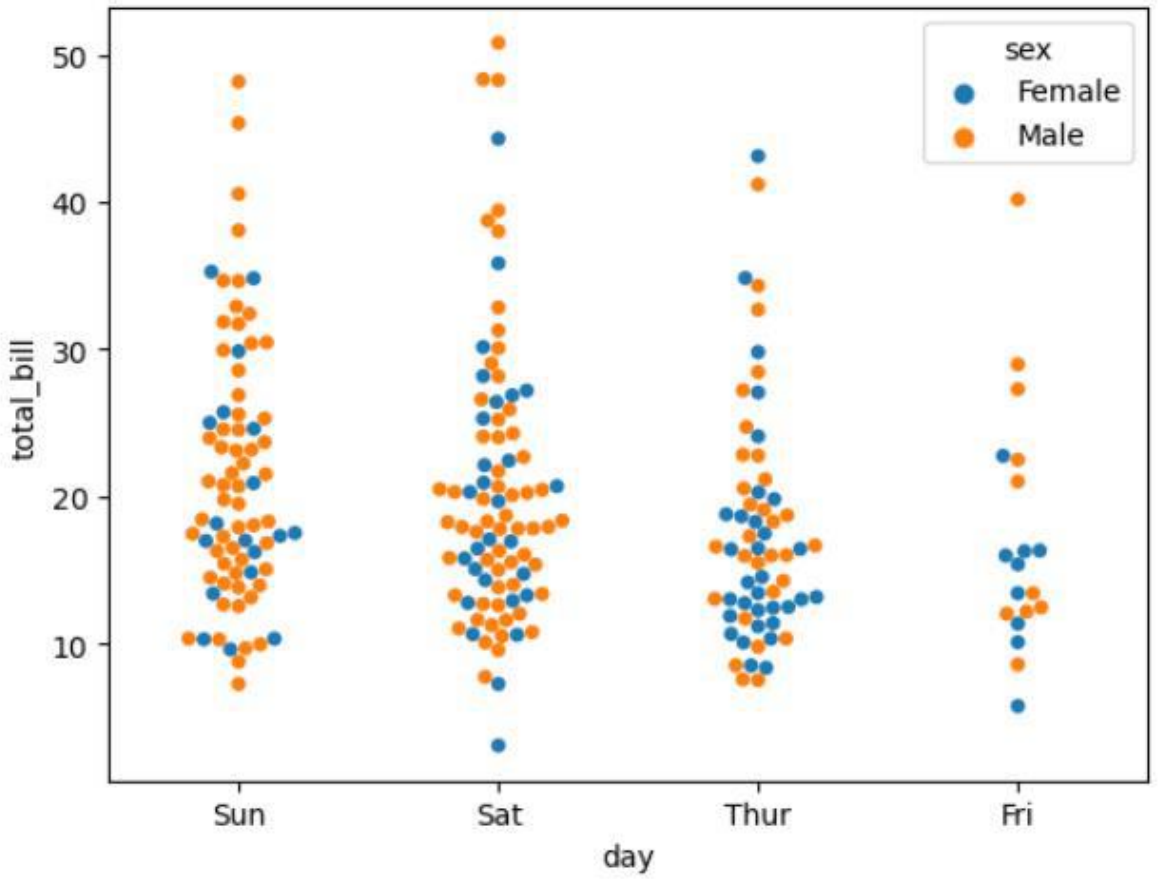
```
sns.stripplot(x="time", y="total_bill", hue="sex", data=tips);
```



Swarmplot

- swarmplot() method is used to draw a non-overlapping scatter plot where one of the variables is a categorical variable.

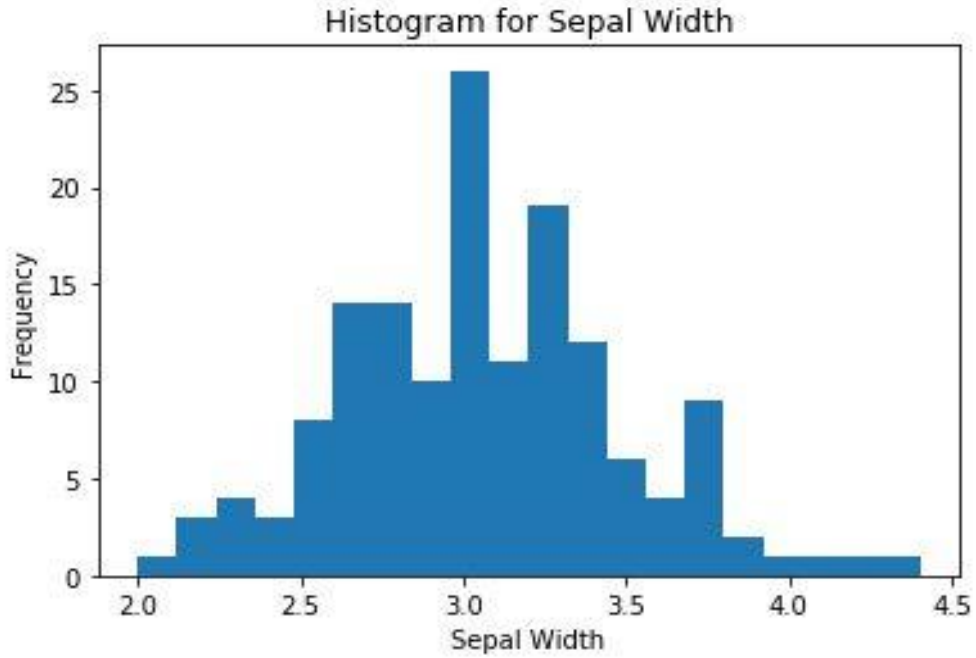
sns.swarmplot(x="day", y="total_bill", hue="sex", data=tips);



Histogram

- The histogram shows the distribution of a continuous variable. It can discover the frequency distribution for a single variable in a univariate analysis.

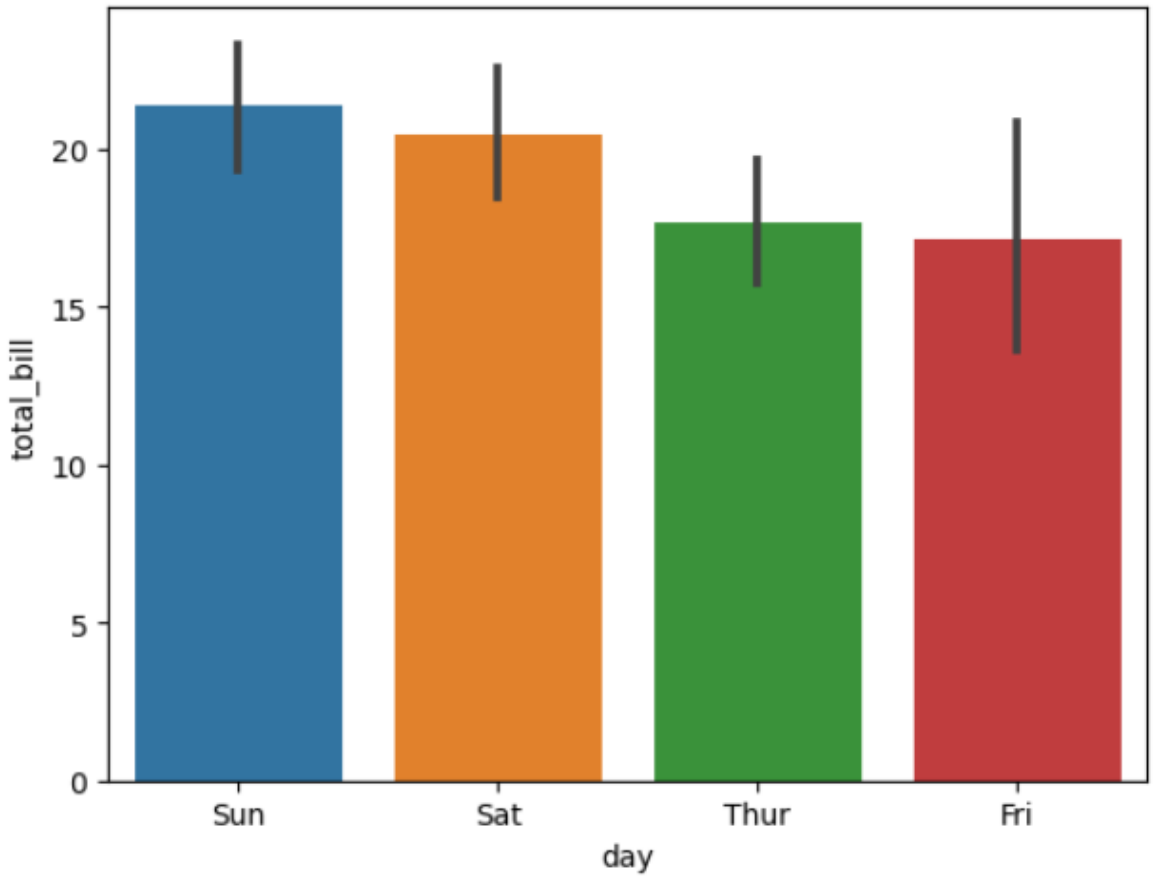
```
plt.hist(iris["sepal_width"],bins = 20)
plt.title("Histogram for Sepal Width")
plt.xlabel("Sepal Width")
plt.ylabel("Frequency")
plt.show()  # This is used to only display the graph in the o/p
```



BarPlot

- Bar Chart or Bar Plot is used to represent categorical data with vertical or horizontal bars. It is a general plot that allows you to aggregate the categorical data based on some function, by default the mean.

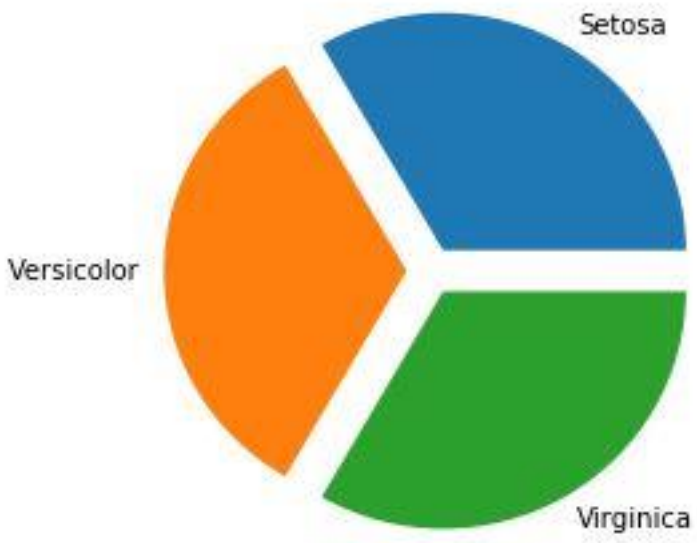
```
sns.barplot(x='day', y='total_bill', data=tips, palette='tab10');
```



Pie Chart

- Pie Chart is a type of plot which is used to represent the proportion of each category in categorical data. The whole pie is divided into slices which are equal to the number of categories.

```
labels = ['Setosa', 'Versicolor','Virginica']
sizes = [50,50,50]
plt.pie(sizes,labels=labels,explode= (0.1,0.1,0.1))
plt.axis('equal') # To set a proper axis
plt.show()
```

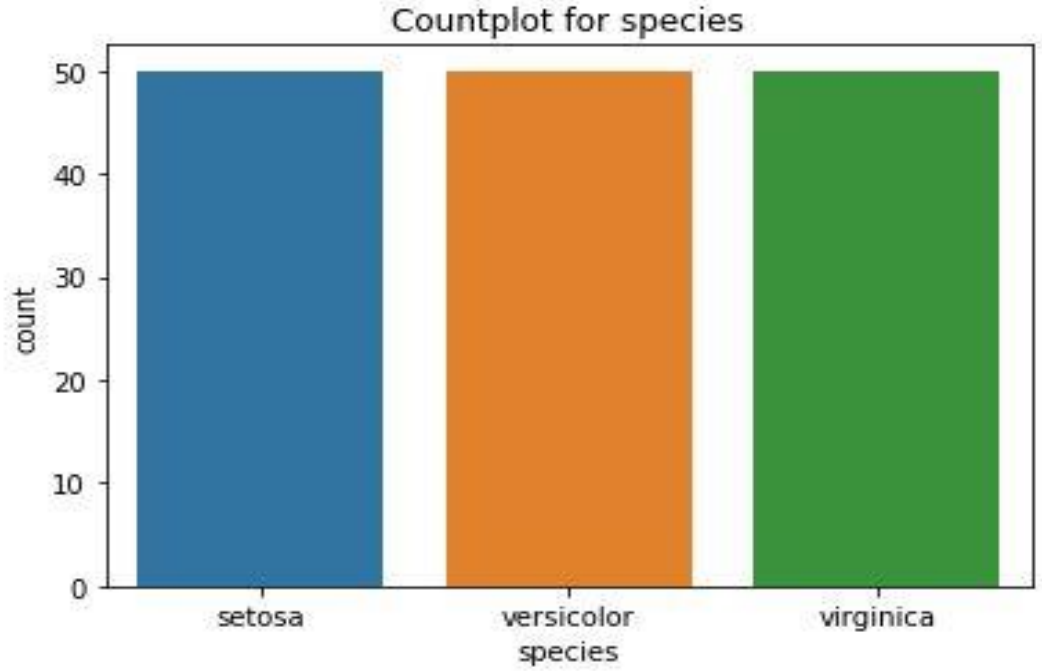


Countplot

- Countplot is similar to a bar plot except that we only pass the X-axis and Y-axis represents explicitly counting the number of occurrences. Each bar represents count for each category of species.

```
sns.countplot(x='species',data=iris)
plt.title("Countplot for species")
```

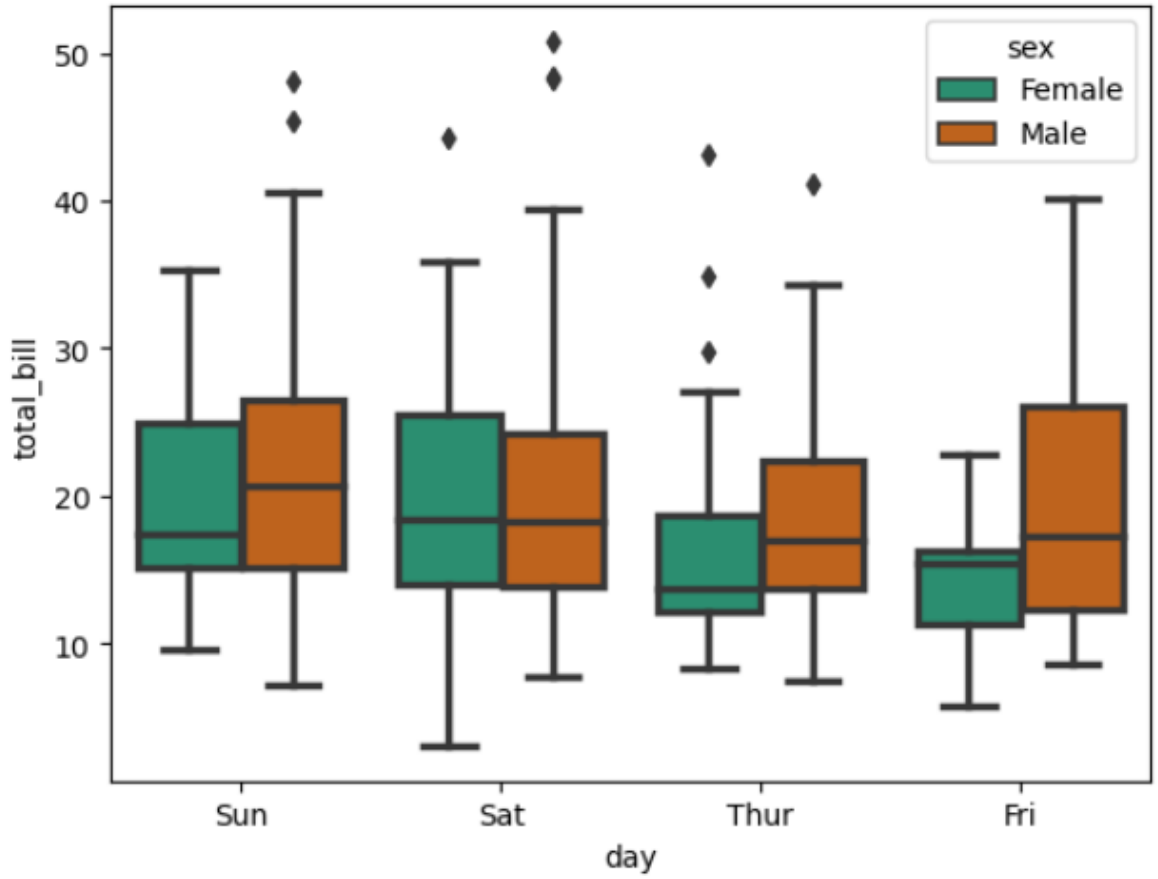
```
Text(0.5,1,'Countplot for species')
```



Boxplot

- Boxplot is used to show the distribution of a variable. The box plot is a standardized way of displaying the distribution of data based on the five-number summary: minimum, first quartile, median, third quartile, and maximum.

sns.boxplot(x='day', y='total_bill', hue='sex', data=tips, linewidth =2.5, palette='Dark2');

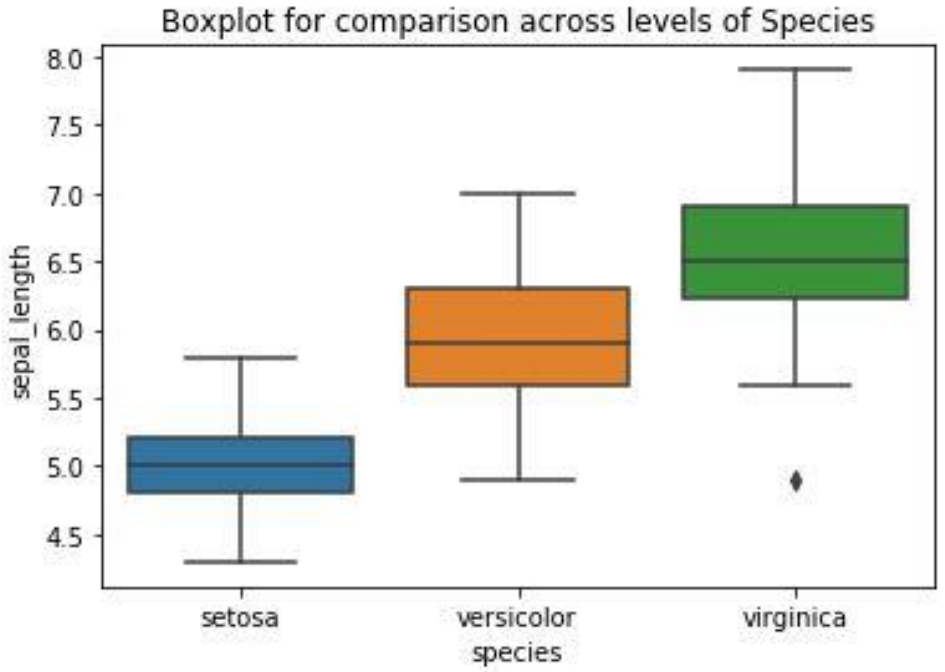


Boxplot

- Boxplot is used to show the distribution of a variable. The box plot is a standardized way of displaying the distribution of data based on the five-number summary: minimum, first quartile, median, third quartile, and maximum.

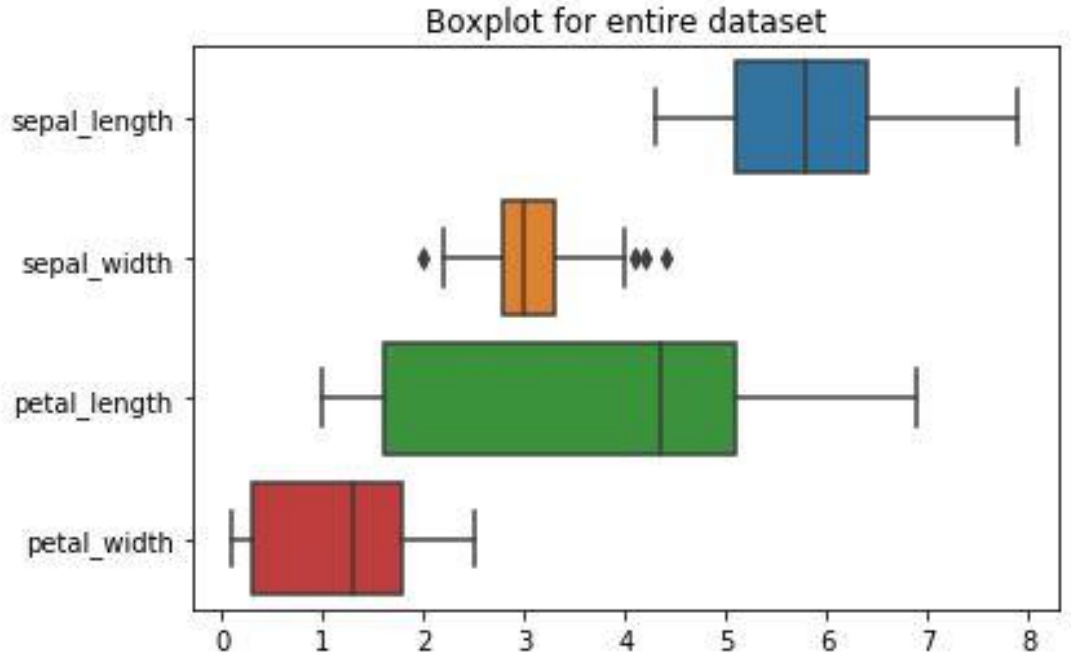
```
sns.boxplot(x = "species", y="sepal_length", data=iris)
plt.title("Boxplot for comparison across levels of Species")
```

```
Text(0.5,1,'Boxplot for comparison across levels of Species')
```




```
sns.boxplot(data= iris, orient='h')
plt.title("Boxplot for entire dataset")
```

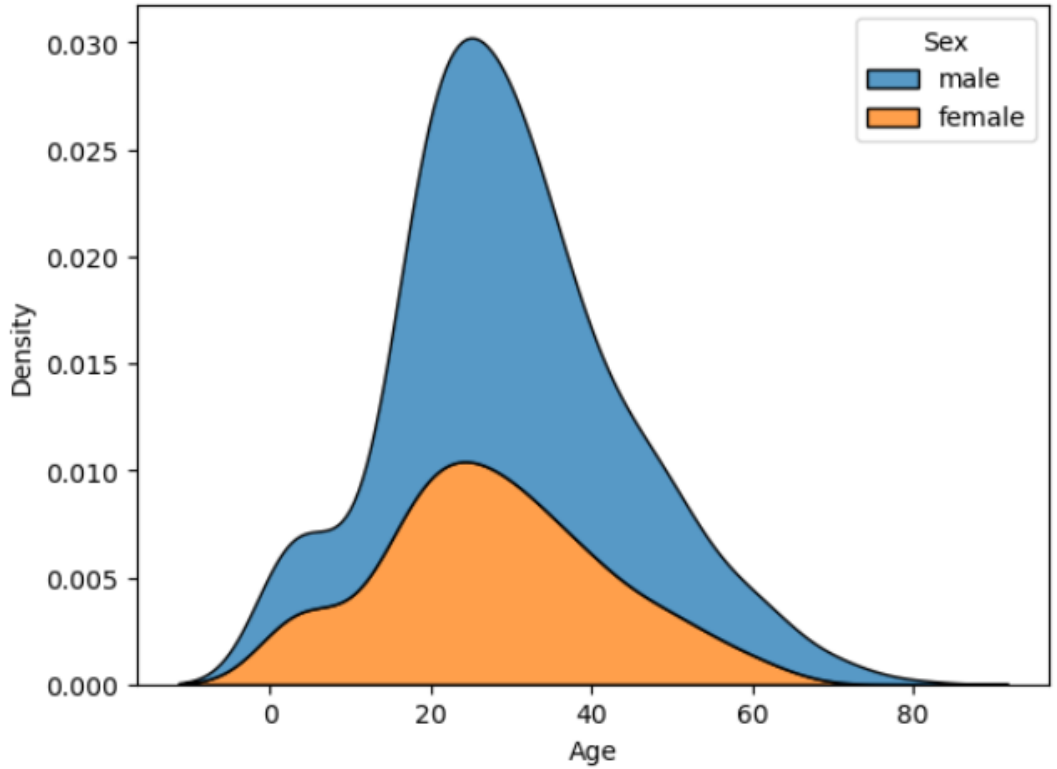
```
Text(0.5,1,'Boxplot for entire dataset')
```



Kdeplot

- A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

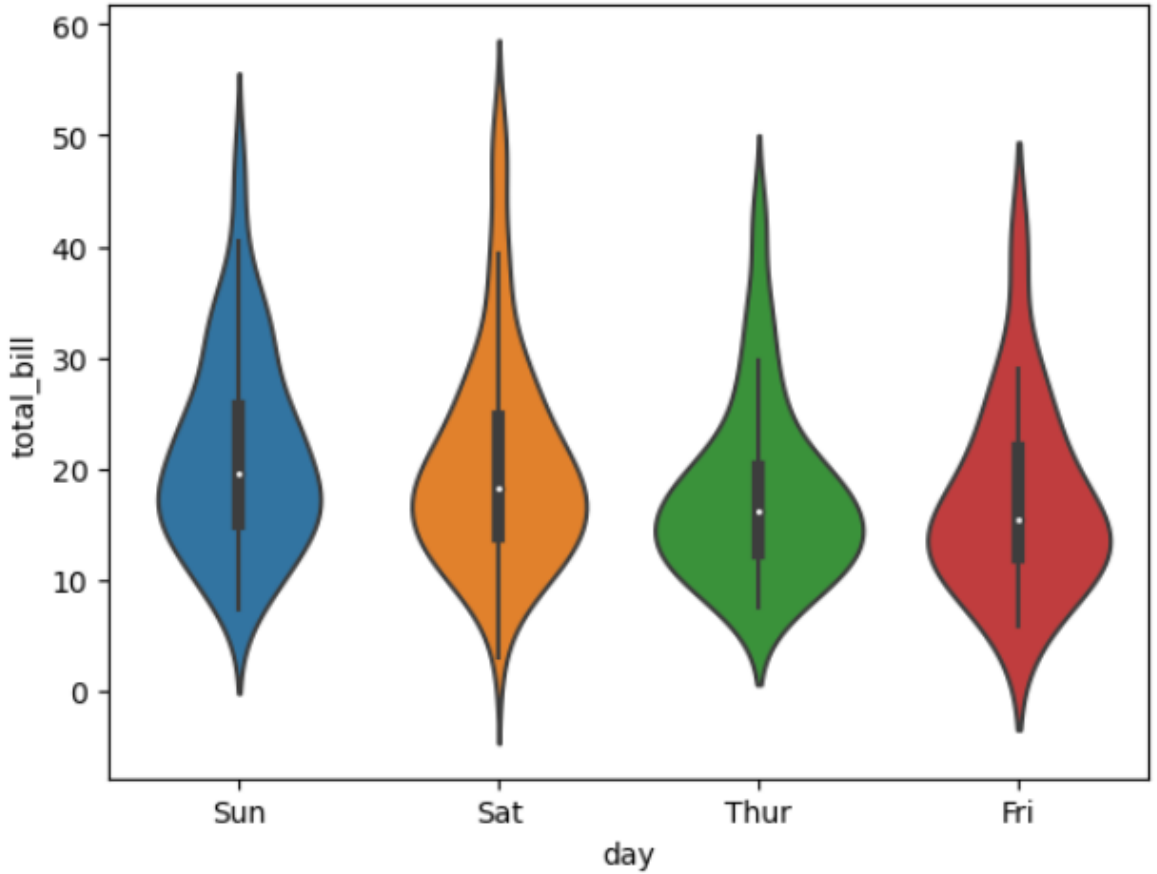
sns.kdeplot(data=df , x='Age', hue='Sex', multiple='stack', palette='tab10');



Violinplot

- Violin plots are used when you want to observe the distribution of numeric data, and are especially useful when you want to make a comparison of distributions between multiple groups. The peaks, valleys, and tails of each group's density curve can be compared to see where groups are similar or different.

sns.violinplot(x="day", y="total_bill", data=tips);



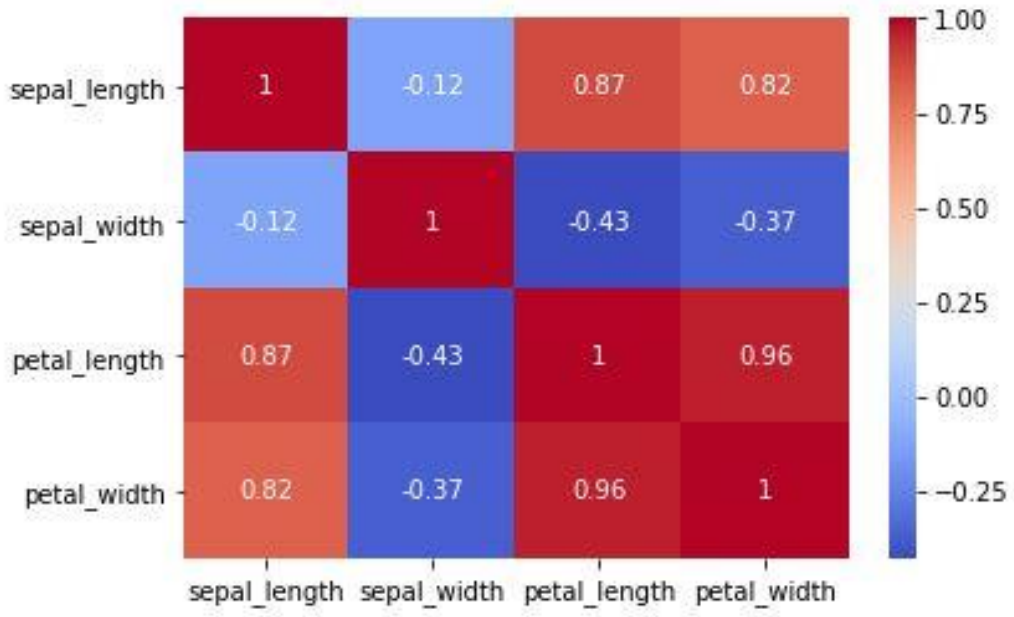
Heatmap

- Heatmap is a type of Matrix plot that allows you to plot data as color-encoded matrices. It is mostly used to find multi-collinearity in a dataset.

- To plot a heatmap, your data should already be in a matrix form, the heatmap basically just colors it in for you.

```
sns.heatmap(iris.corr(),cmap='coolwarm',annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xdbf92e8>
```

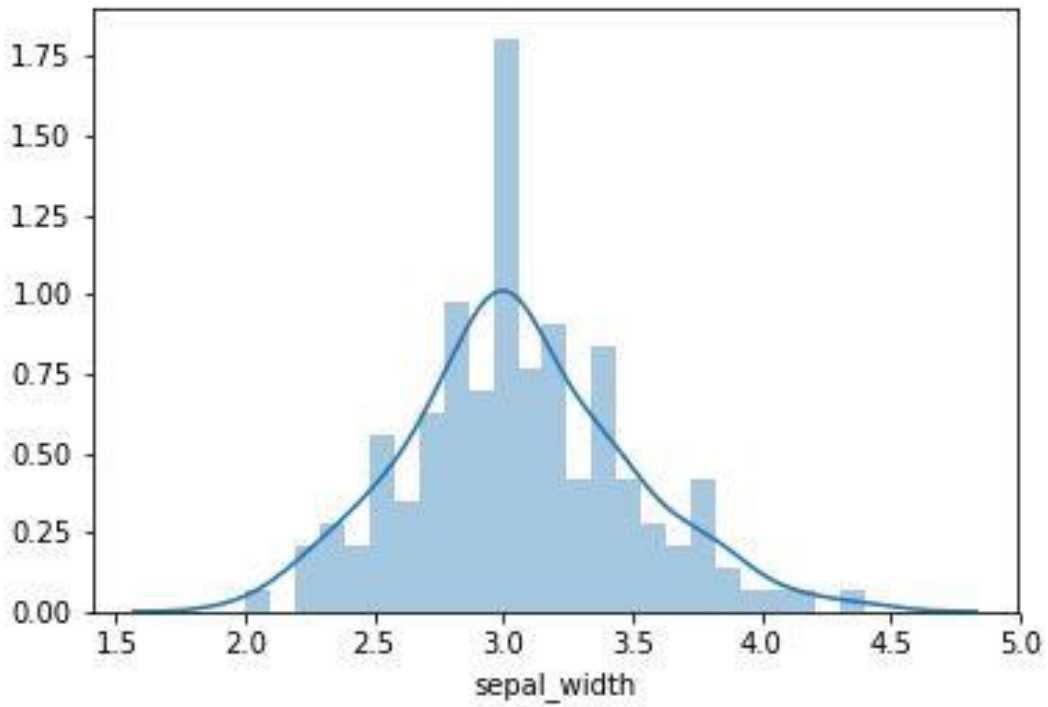


Distplot

- The Distplot shows the distribution of a univariate data

```
sns.distplot(iris['sepal_width'],bins = 25,kde= True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xe3080b8>
```

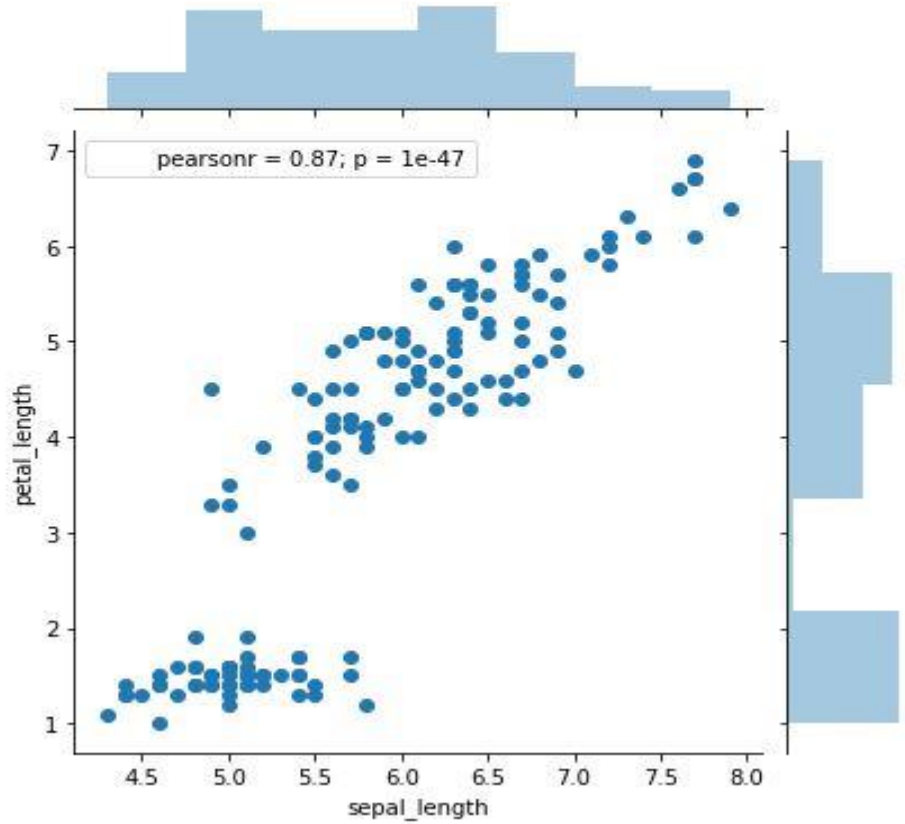


Jointplot

- Jointplot is used to represent the distribution of one variable to match up with the distribution of another variable. To be more specific, Jointplot allows you to basically match up two Distplots for bivariate data.

```
sns.jointplot(x='sepal_length',y='petal_length',data=iris,kind='scatter')
```

```
<seaborn.axisgrid.JointGrid at 0xd34a128>
```



Conclusions

- Hence, we have covered most of the basics of Python Visualization using seaborn and matplotlib.
- I hope this article will give you a head start for diving into Python Visualization.
- Also, You can refer to the official documentation for Matplotlib and Seaborn for further reference and deep understandings