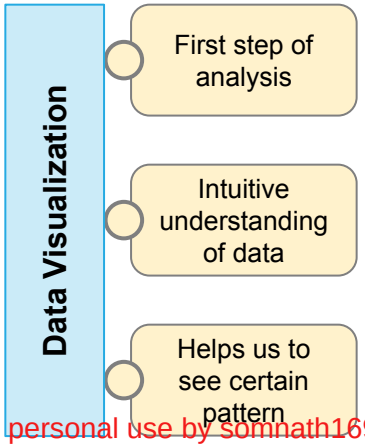
Data Visualization

What is visualization

- Data visualization allows us to quickly interpret the data and adjust different variables to see their effect
- It is a presentation of data in a pictorial or graphical format



Visualization using Matplotlib

Introduction to matplotlib

- Matplotlib is a Python 2D plotting library
- ‘pyplot’ is a subpackage of matplotlib that provides a MATLAB-like way of plotting
- Matplotlib provides a simple way of plotting the various plots like histogram, bar plot, scatter plot and so on

Installation

Open terminal program(for Mac user) or command line(for Windows) and install it using following command:

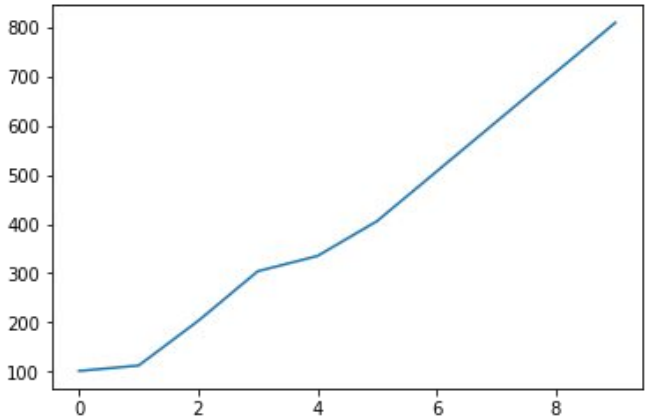
```
conda install
matplotlib
```

Or

```
pip install matplotlib
```

Line plot

It is a simple plot that displays the relationship between one independent and one dependent variable

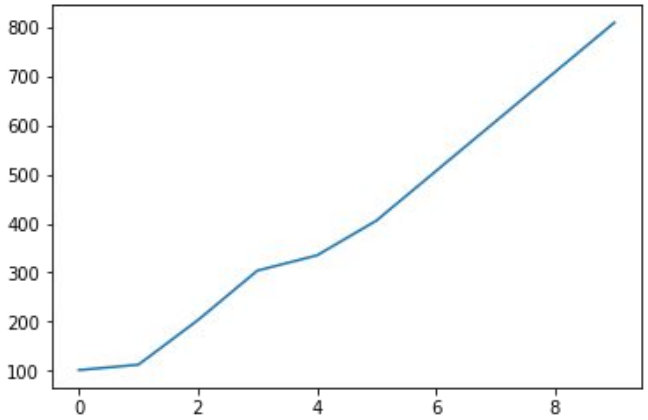


Plot a line plot from list data

```
# import matplotlib
import matplotlib.pyplot as plt
# create a list
data = [101,112,203,304,335,406,507,608,709,810]
plt.plot(data)
plt.show()
```

Plot the data

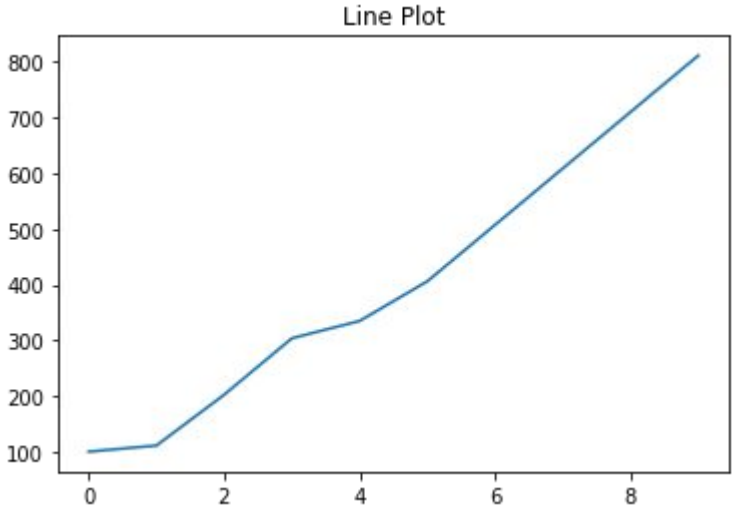
Display the plot



Display the title of the graph

```
# create a list
data = [101,112,203,304,335,406,507,608,709,810]
plt.plot(data)
plt.title('Line Plot')
plt.show()
```

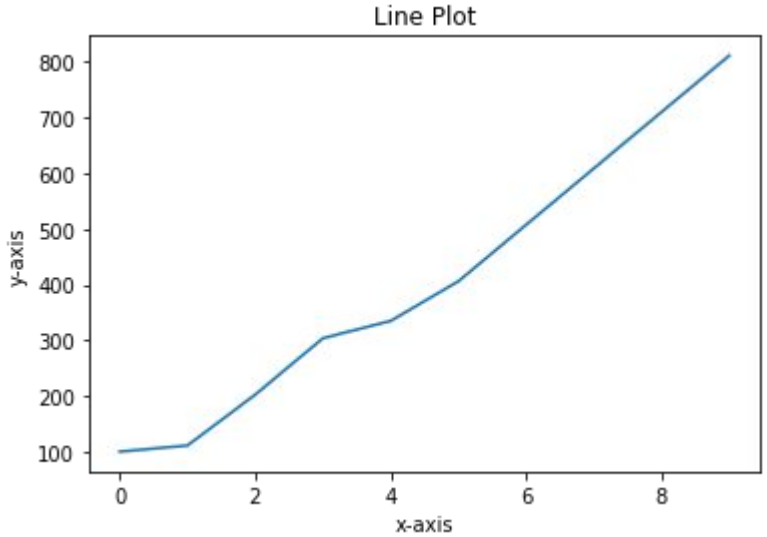
Display the title



Add labels to x and y axis

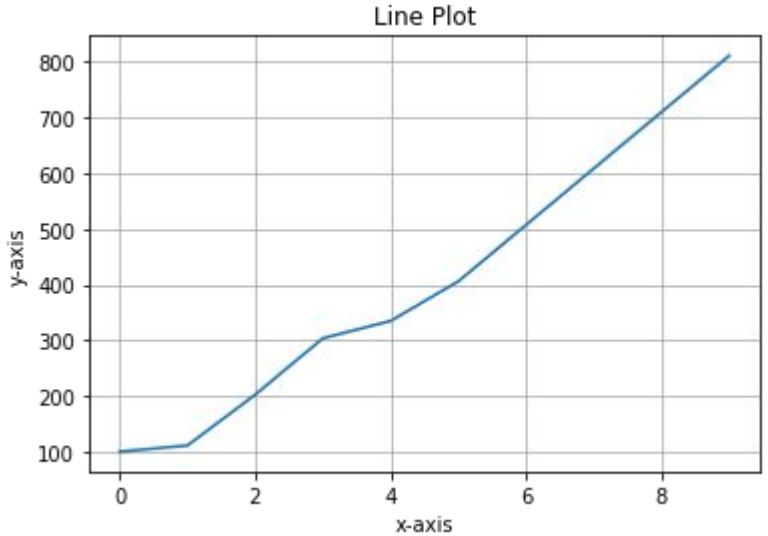
```
data = [101,112,203,304,335,406,507,608,709,810]
plt.plot(data)
plt.title('Line Plot')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```

Add labels to respective x and y axis



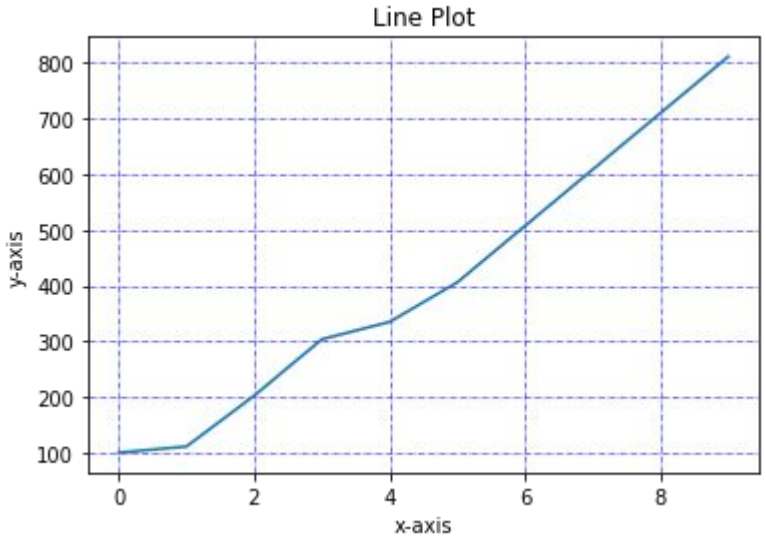
Add grid lines to the chart

```
data = [101,112,203,304,335,406,507,608,709,810]
plt.plot(data)
plt.title('Line Plot')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.grid()
plt.show()
```



Customize the grid

```
data = [101,112,203,304,335,406,507,608,709,810]
plt.plot(data)
plt.title('Line Plot')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.grid()
plt.grid(linestyle='-.', linewidth='0.5', color='blue')
plt.show()
```



Real life example

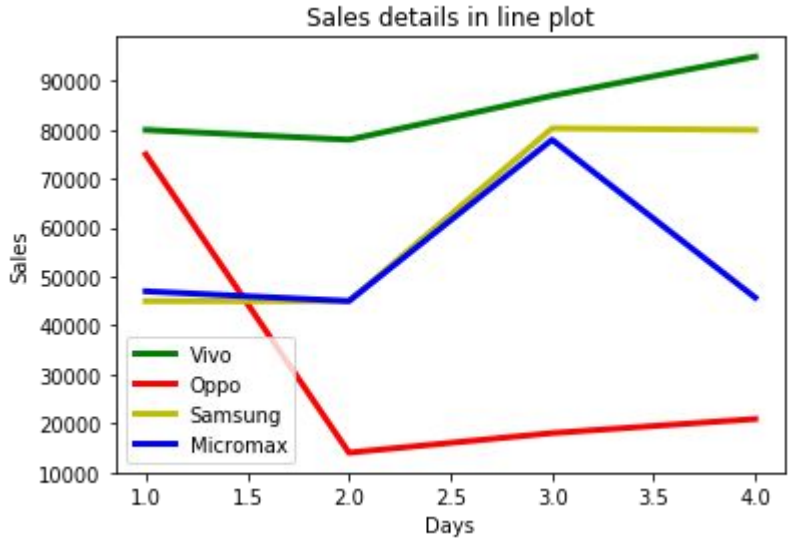
Consider the sales of the company. We represent following data points in the jupyter notebook

Days	Sales			
	Vivo	Oppo	Samsung	Micromax
Day1	80000	75000	45000	47000
Day2	78000	14000	45000	45000
Day3	87000	18000	80333	78000
Day4	95000	20888	80000	45700

Multiple line plots

Various lines present in the graph have unique colors, and each of them represents the sales of a different company

```
x = [1,2,3,4]
y1 = [80000,78000,87000,95000]
y2 = [75000,14000,18000,20888]
y3 = [45000,45000,80333,80000]
y4 = [47000,45000,78000,45700]
plt.plot(x,y1,'g',label='Vivo', linewidth=5)
plt.plot(x,y2,'c',label='Oppo',linewidth=5)
plt.plot(x,y3,'k',label='Samsung',linewidth=5)
plt.plot(x,y4,'y',label='Micromax',linewidth=5)
plt.title('Sales details in line plot')
plt.ylabel('Sales')
plt.xlabel('Days')
plt.legend()
plt.show()
```



Scatter plot

- Dot-based plotting of two or more than two variables along x and y axis represents a scatter plot
- Scatter plot are used to represent the correlation between two or more variables
- We used iris data to create scatter plot

```
import pandas as pd
df_iris = pd.read_csv('Iris.csv')
df_iris.head()
```

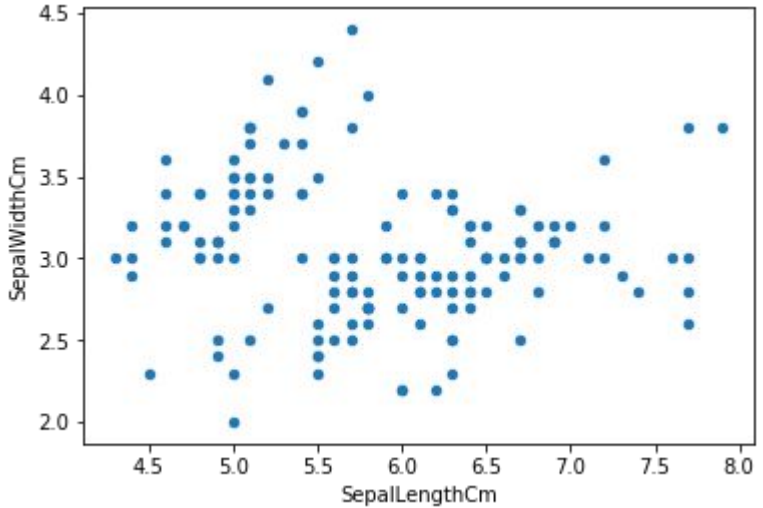
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Scatter plot

Use scatter() method to create scatter plot in matplotlib

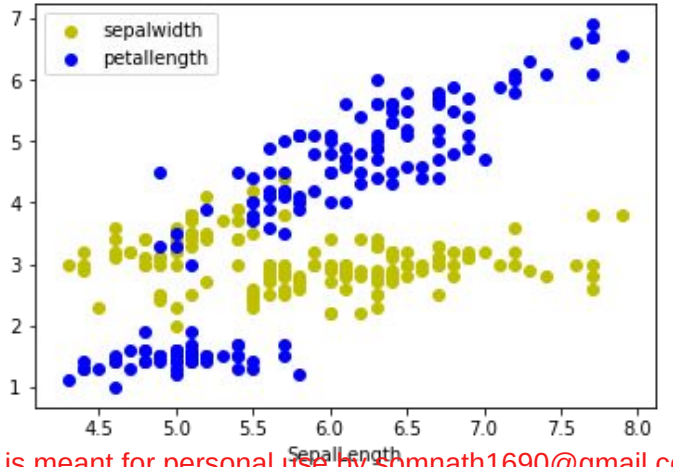
```
df_iris.plot.scatter(x='SepalLengthCm', y='SepalWidthCm')
```

Set the x and y label to the column names



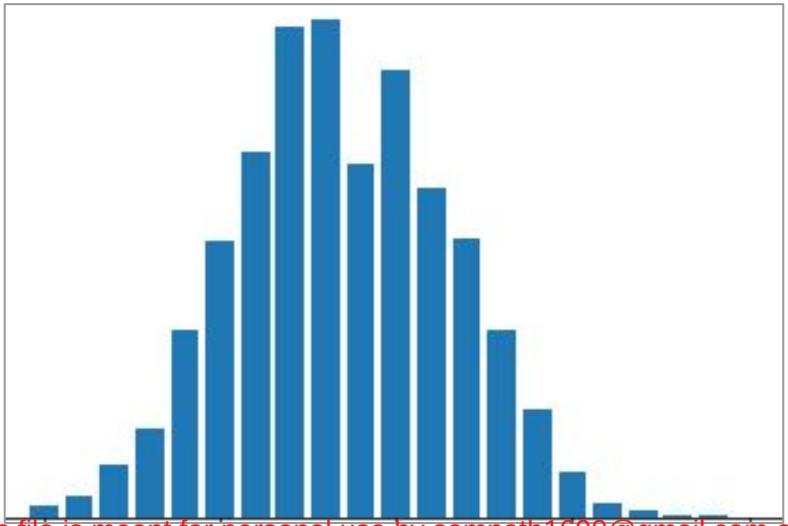
Scatter plot for multiple variables

```
plt.scatter('SepalLengthCm', 'SepalWidthCm', label='sepalwidth',color='y', data=df_iris )
plt.scatter('SepalLengthCm', 'PetalLengthCm', label='petallength',color='b', data=df_iris)
plt.xlabel('SepalLength')
plt.legend()
```



Bar plot

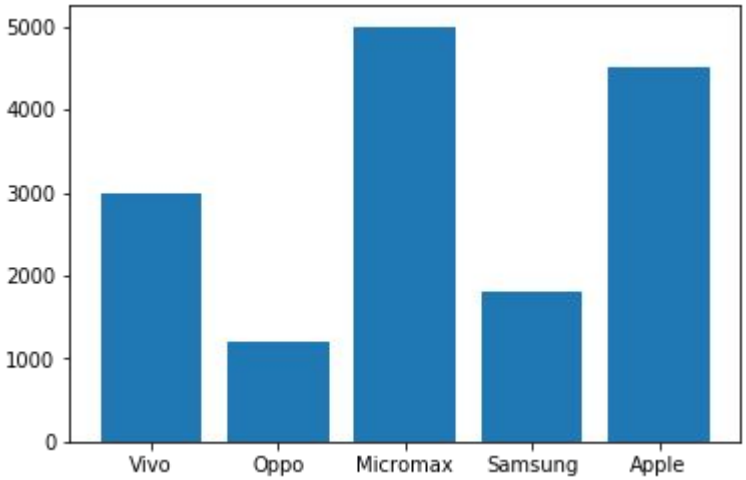
- Categorical data can be plotted in rectangular blocks with different heights proportional to the the values. Such type of plot is called bar plot
- It can be plot in both horizontal and vertical manner



Bar plot

In the graph, each block shows the sales of the individual

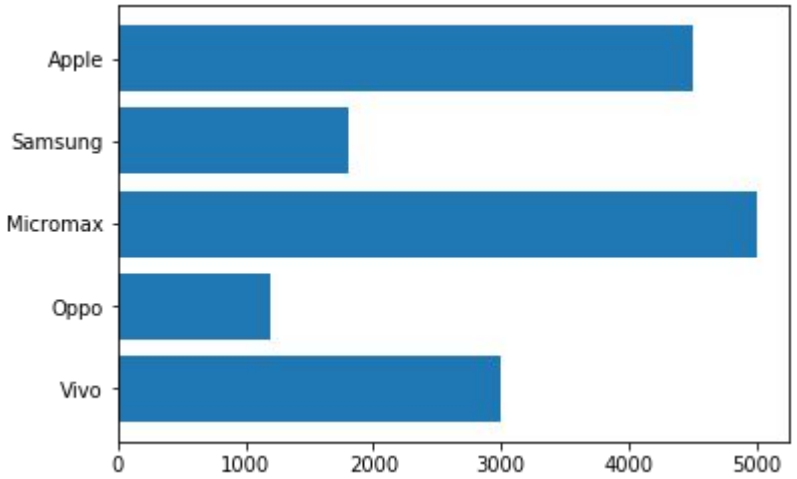
```
import numpy as np
# create a list of sales
sales = [3000, 1200, 5000, 1800, 4500]
bars = ('Vivo', 'Oppo', 'Micromax', 'Samsung', 'Apple')
y_pos = np.arange(len(bars))
# Create bars
plt.bar(y_pos, sales)
# Create names on the x-axis
plt.xticks(y_pos, bars)
# Show graphic
plt.show()
```



Horizontal bar plot

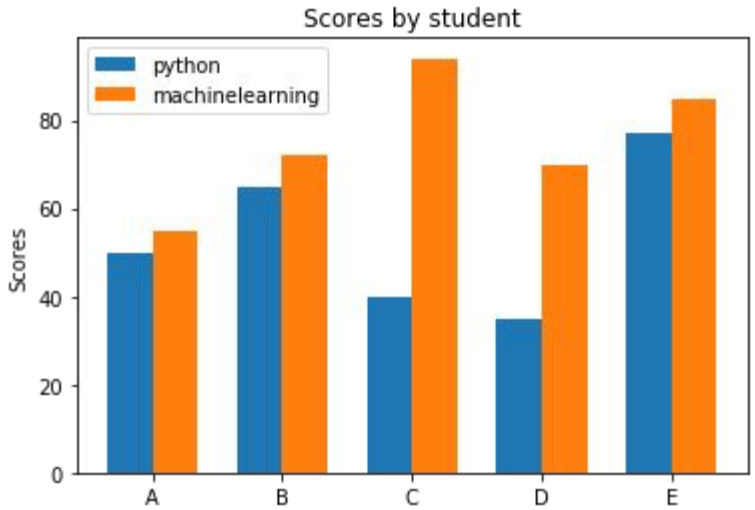
We can plot the exact same chart horizontally using plt.barh() function

```
# create a list of sales
sales = [3000, 1200, 5000, 1800, 4500]
bars = ('Vivo', 'Oppo', 'Micromax', 'Samsung', 'Apple')
y_pos = np.arange(len(bars))
# Create bars
plt.barh(y_pos, sales)
# Create names on the x-axis
plt.yticks(y_pos, bars)
# Show graphic
plt.show()
```



Multiple bar plot

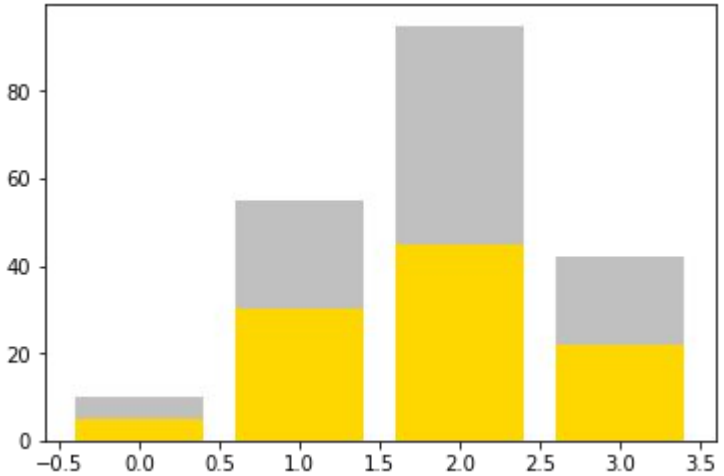
```
N = 5
python = (50, 65, 40, 35, 77)
machinelearning = (55, 72, 94, 70, 85)

ind = np.arange(N)
width = 0.35
plt.bar(ind, python, width, label='python')
plt.bar(ind + width, machinelearning, width,
    label='machinelearning')

plt.ylabel('Scores')
plt.title('Scores by student')

plt.xticks(ind + width / 2, ('A', 'B', 'C', 'D', 'E'))
plt.legend()
plt.show()
```



Set the location and the label of the x-axis

Stacked bar plot

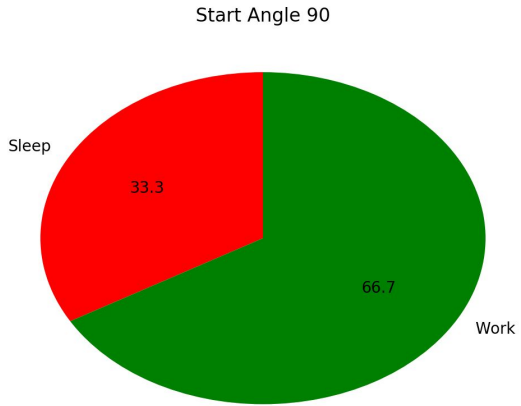
```
A = [5., 30., 45., 22.]
B = [5., 25., 50., 20.]
X = range(4)
plt.bar(X, A, color = 'gold')
plt.bar(X, B, color = 'silver', bottom = A)
plt.show()
```



y coordinate of the bars

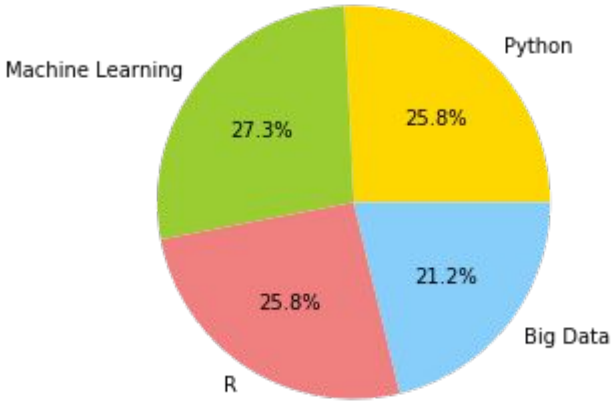
Pie plot

Statistical data can be represented in a circular graph where the circle is divided into portions that denote a particular data. Such type of graph is called pie plot



Pie plot

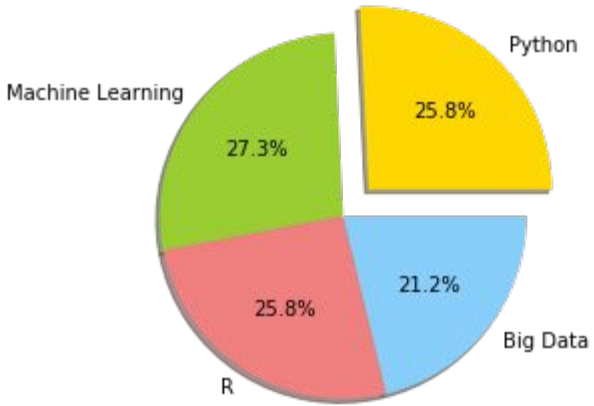
```
# Data to plot
labels = 'Python', 'Machine Learning', 'R', 'Big Data'
sizes = [85, 90, 85, 70]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
# Plot
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors)
plt.axis('equal')
plt.show()
```



It gives numeric labels

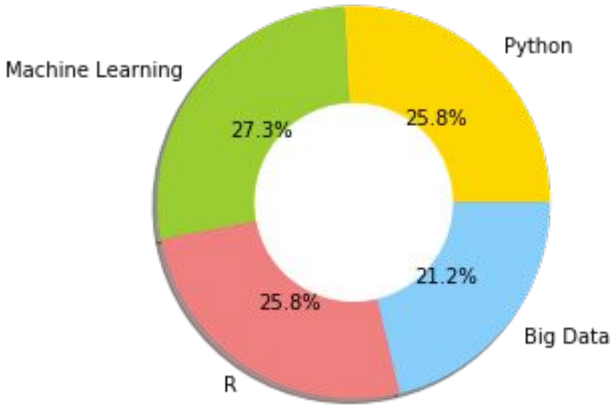
Exploded pie plot

```
# Data to plot
labels = 'Python', 'Machine Learning', 'R', 'Big Data'
sizes = [85, 90, 85, 70]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (0.2, 0, 0, 0)  # explode 1st slice
# Plot
plt.pie(sizes, labels=labels, explode=explode, colors=colors,
autopct='%1.1f%%', shadow=True)
plt.axis('equal')
plt.show()
```



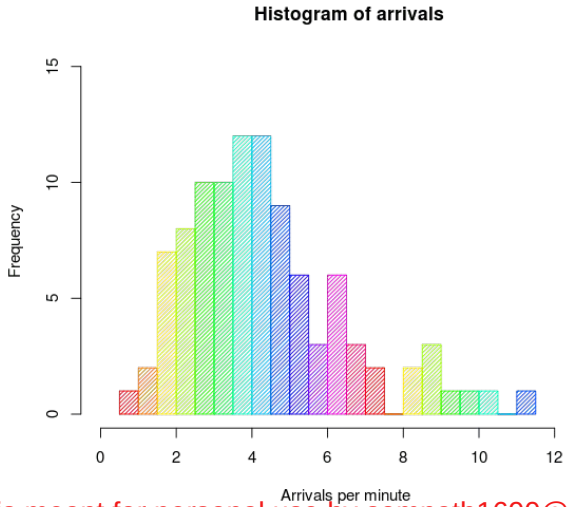
Donut pie plot

```
# Data to plot
labels = 'Python', 'Machine Learning', 'R', 'Big Data'
sizes = [85, 90, 85, 70]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
# Plot
plt.pie(sizes, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True)

# add a circle at the center
circle =  plt.Circle( (0,0), 0.5, color='white')
plt.gcf()
plt.gca().add_artist(circle)

# display the plot
plt.axis('equal')
plt.show()
```



Histogram plot

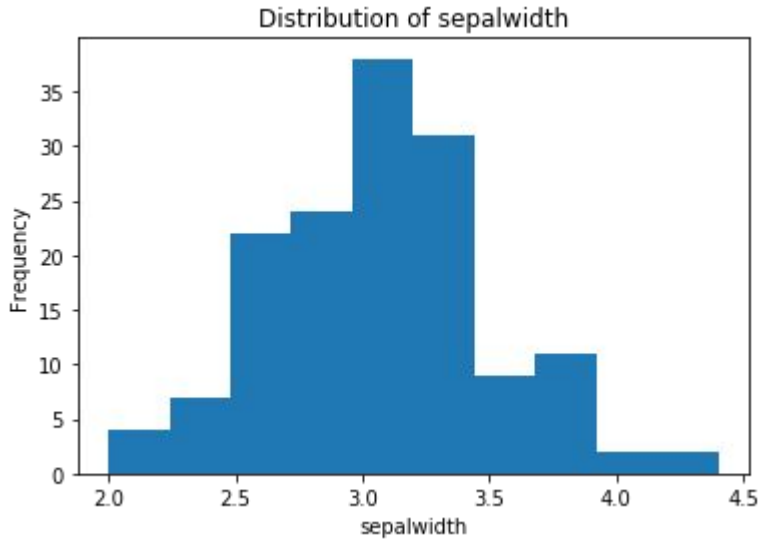
- Histogram is used to represent graphical distribution of numerical data
- It is an estimate of the probability distribution of a continuous data



Histogram plot

Use iris data to plot histogram

```
# plot histogram
df_iris['SepalWidthCm'].plot.hist()
# add the graph title and axes labels
plt.title('Distribution of sepalwidth')
plt.xlabel('sepalwidth')
plt.ylabel('Frequency')
```

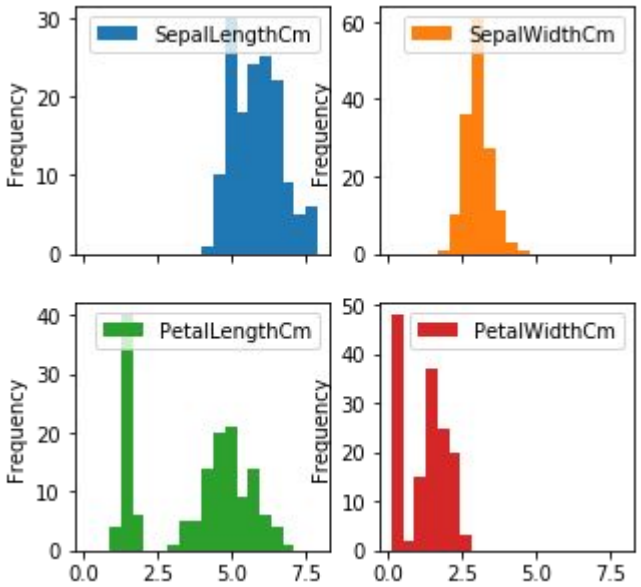


Create multiple histograms

```
df_iris.plot.hist(subplots=True, layout=(2,2), figsize=(5, 5), bins=20)
plt.show()
```

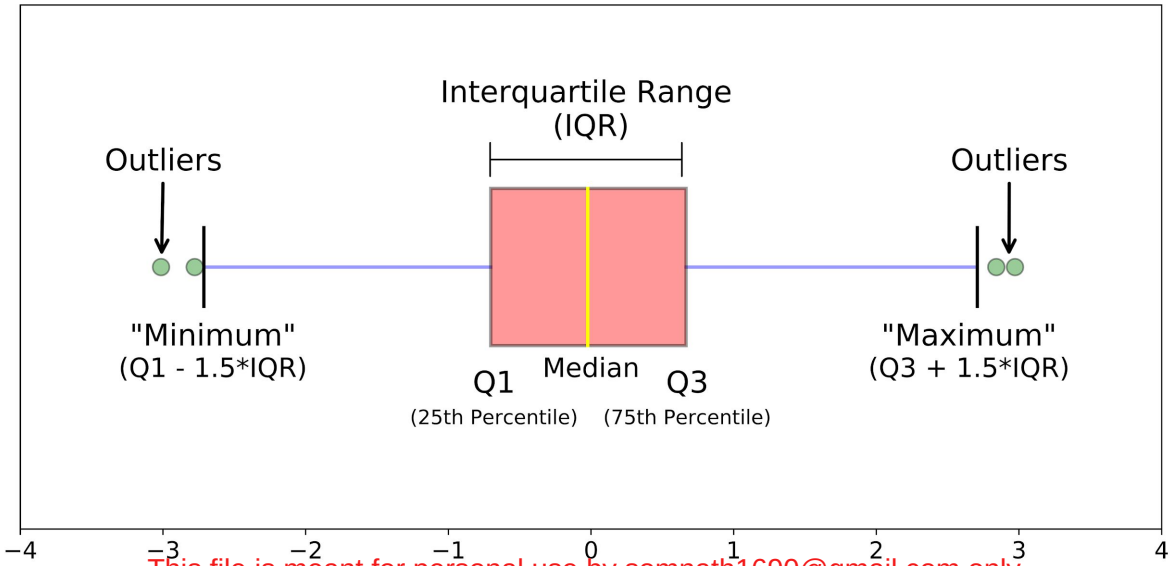
Separate plot for each feature

Number of plots per row and column



Box plot

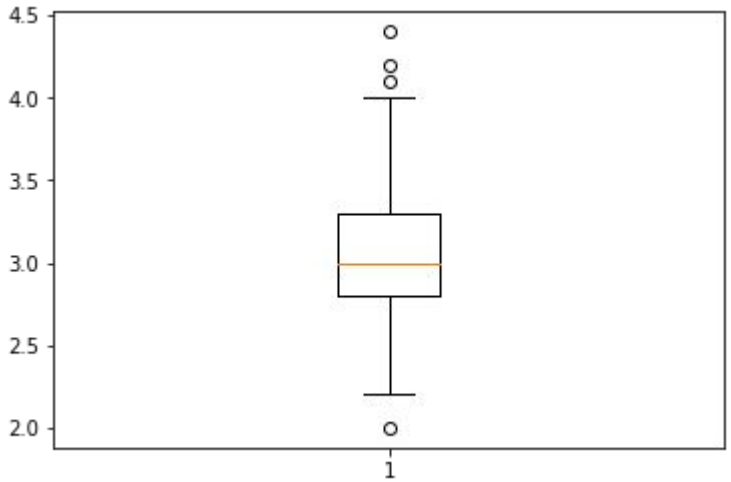
- Box-plot is used to represent group of numerical data
- It gives us summary statistics for one or several numeric variables



Create a single box plot

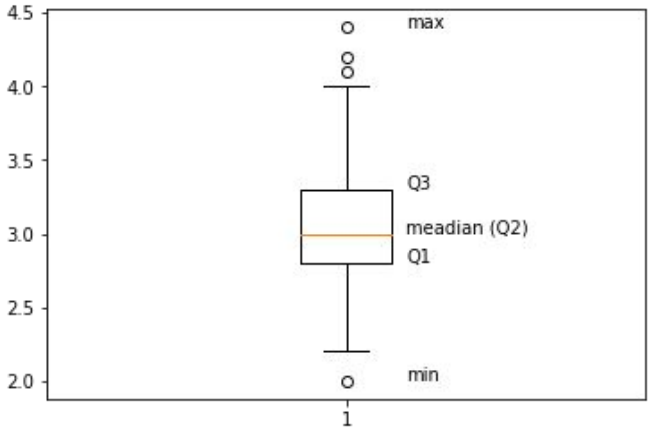
Use iris data to plot boxplot

```
# create a boxplot
plt.boxplot(df_iris['SepalWidthCm'])
plt.show()
```



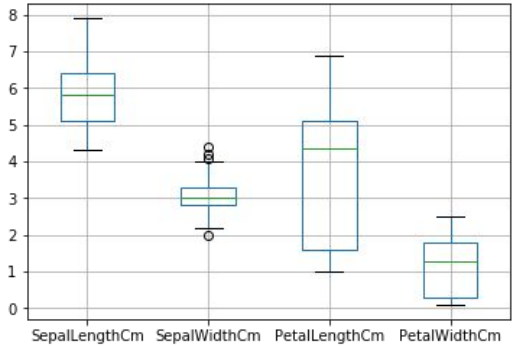
Add labels for box plot - five number summary

```
# create a boxplot
plt.boxplot(df_iris['SepalWidthCm'])
# add labels for five number summary
plt.text(x = 1.1, y = df_iris['SepalWidthCm'].min(), s ='min')
plt.text(x = 1.1, y = df_iris.SepalWidthCm.quantile(0.25), s ='Q1')
plt.text(x = 1.1, y = df_iris['SepalWidthCm'].median(), s ='meadian (Q2)')
plt.text(x = 1.1, y = df_iris.SepalWidthCm.quantile(0.75), s ='Q3')
plt.text(x = 1.1, y = df_iris.SepalWidthCm.max(), s ='max')
plt.show()
```



Create boxplot for all numeric variables

```
# filter all the numeric variables
df_numeric = df_iris.select_dtypes(include=[np.number])
# create a boxplot for all numeric variables
df_numeric.boxplot(['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm'])
```



Visualization using Seaborn

Introduction to seaborn

- Seaborn is used for data visualization and based on matplotlib
- It provides a high-level interface for drawing attractive and informative statistical graphics

Functionalities of seaborn

- Allows comparison between multiple variables
- Supports multi-plot grids
- Available univariate and bivariate visualization
- Availability of different color palettes
- Estimates and plots linear regression automatically

Seaborn Vs. matplotlib

- Matplotlib settings are difficult to figure out. Seaborn comes with numerous customized themes and high-level interfaces
- Matplotlib doesn’t serve well when it comes to dealing with dataframes, while seaborn functions actually work on dataframes

Installation

Open terminal program(for Mac user) or command line(for Windows) and install it using following command:

```
conda install seaborn
```

Or

```
pip install seaborn
```

Strip plot

- Strip plot is one of the kind of scatter plot of categorical data with the help of seaborn
- Categorical data represented in x-axis and values correspond to them represented through y-axis
- Load titanic data to create strip plot

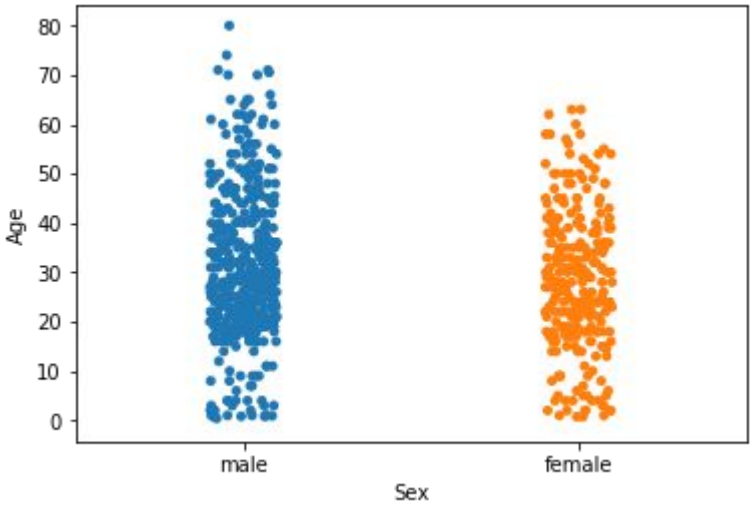
```
df_titanic = pd.read_csv("titanic.csv")
df_titanic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Strip plot

Use stripplot() function to create a plot

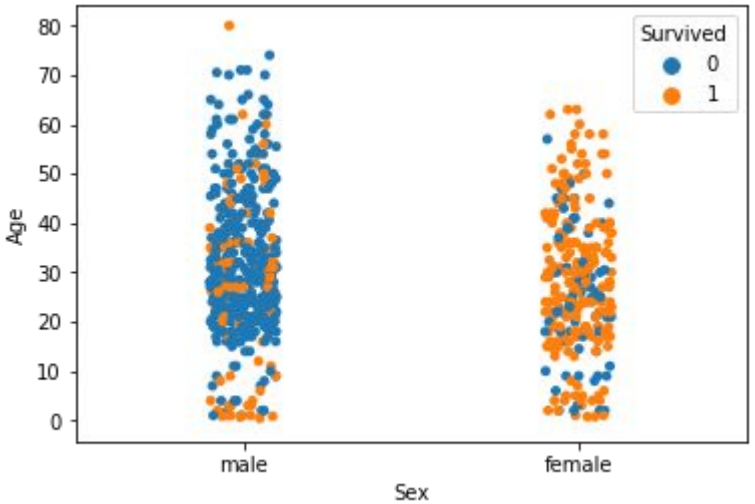
```
sns.stripplot(x='Sex', y='Age', data=df_titanic)
plt.show()
```



Strip plot

You can add categorical column to strip plot using hue parameter

```
sns.stripplot(x='Sex', y='Age',hue= 'Survived', data=df_titanic)
plt.show()
```



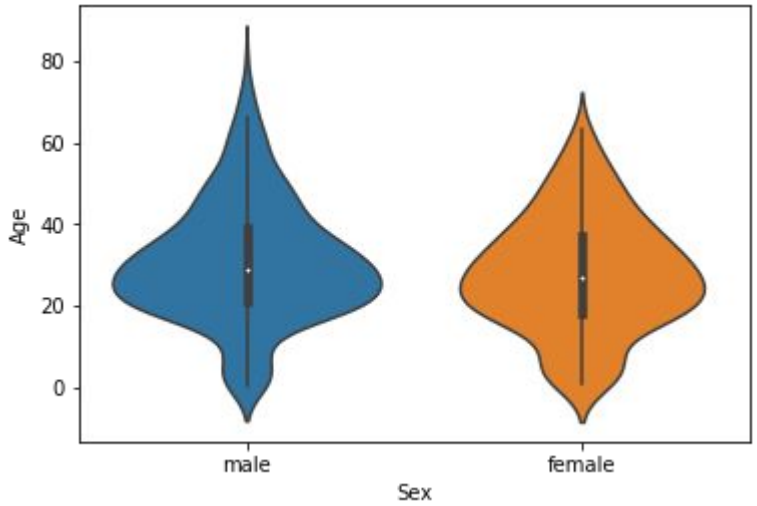
Violin plot

- Violin plot is similar to boxplot, however it allows us to display components that actually correspond to the data point
- It shows the distribution of quantitative data across categorical variables such that those distribution can be compared

Violin plot

violinplot() function is used to plot the violin plot

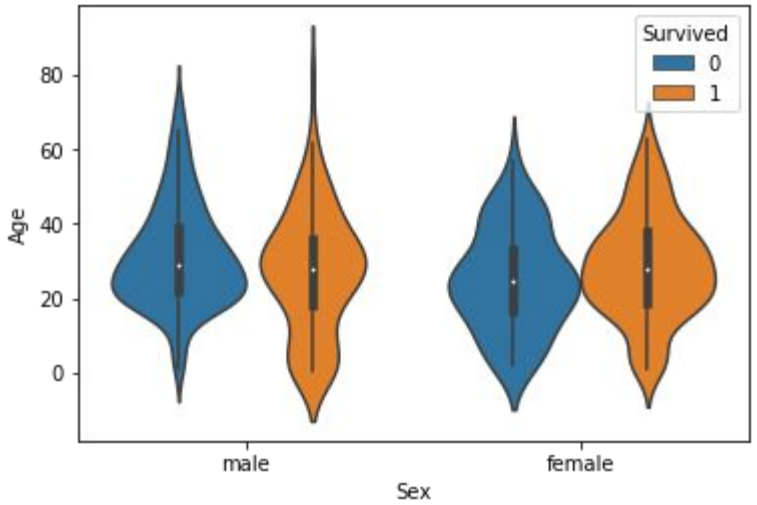
```
sns.violinplot(x='Sex', y='Age', data=df_titanic)
plt.show()
```



Violin plot

You can also add the one more categorical variable to the violin plot using hue parameter

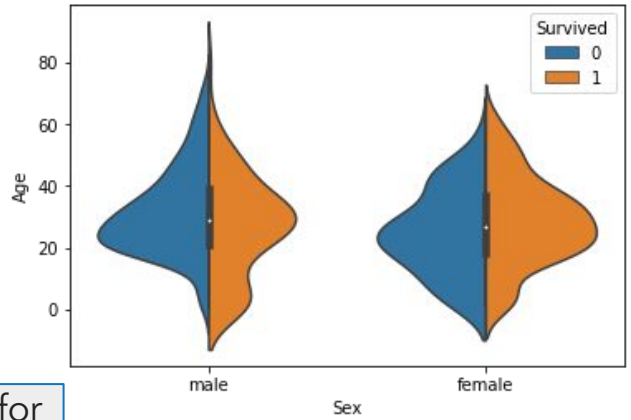
```
sns.violinplot(x='Sex', y='Age', data=df_titanic,hue='Survived')
plt.show()
```



Violin plot

You can have one violin plot divided into two halves, where one halve represents surviving while other half represents the non-surviving passenger

```
sns.violinplot(x='Sex', y='Age', data=df_titanic, hue='Survived', split=True)
plt.show()
```



Pass True as value for the split parameter

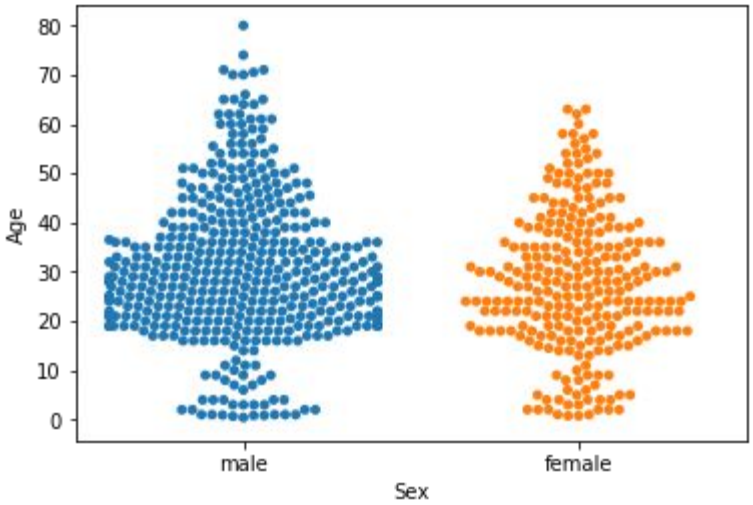
Swarm plot

- Swarm plot is the combination of strip and violin plots
- In the swarm plots, the points are adjusted in such a way that they don't overlap
- Use `swarmplot()` function to create a swarmplot

Swarm plot

Let's create a swarm plot for the distribution of age against gender

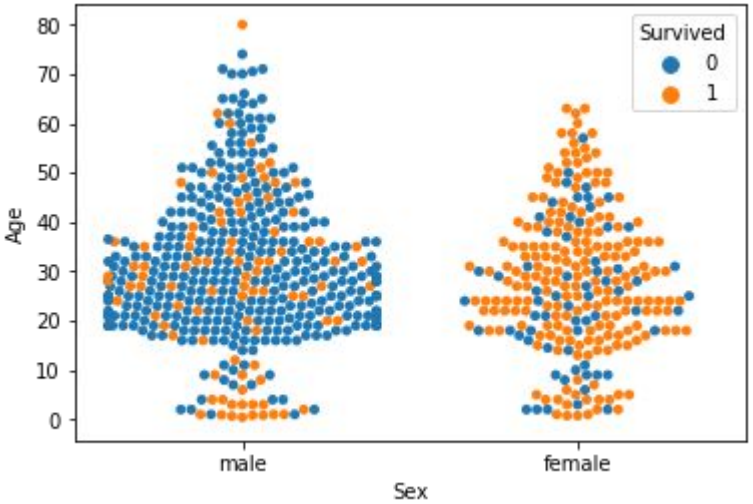
```
sns.swarmplot(x='Sex', y='Age', data=df_titanic)
plt.show()
```



Swarm plot

Add one more categorical variable to the swarm plot using the hue parameter

```
sns.swarmplot(x='Sex', y='Age', data=df_titanic, hue='Survived')
plt.show()
```



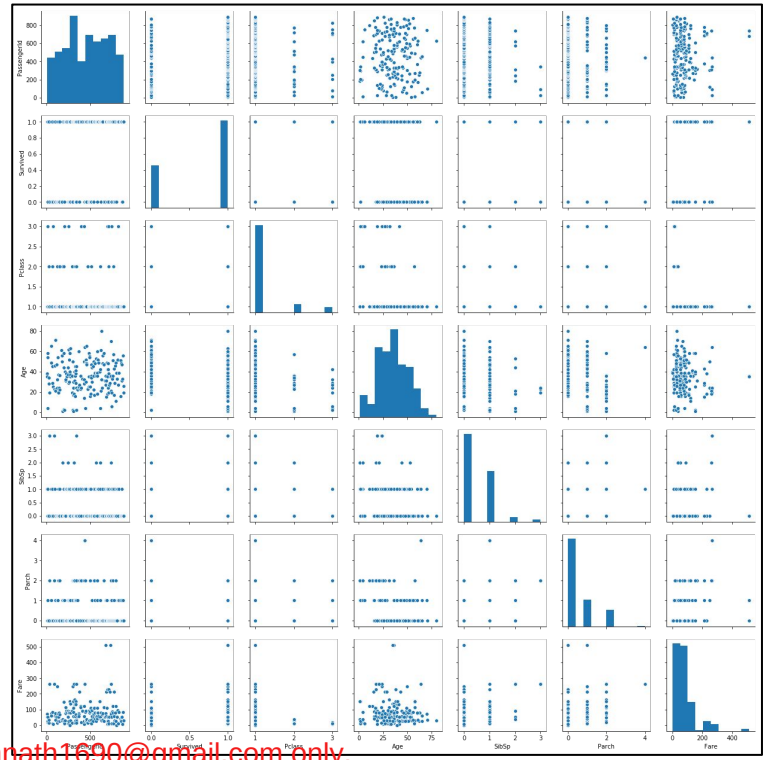
Pair plot

- Pair plot is a type of distribution plot that basically plot for all the possible combination of numeric and boolean variables
- It is used to visualize the relationship between two variables, where the variables can be continuous, categorical or booleans

Pair plot

Use pairplot() function to create pairplot

```
# drop all the null values to create pair plot
df_titanic = df_titanic.dropna()
sns.pairplot(df_titanic)
plt.show()
```



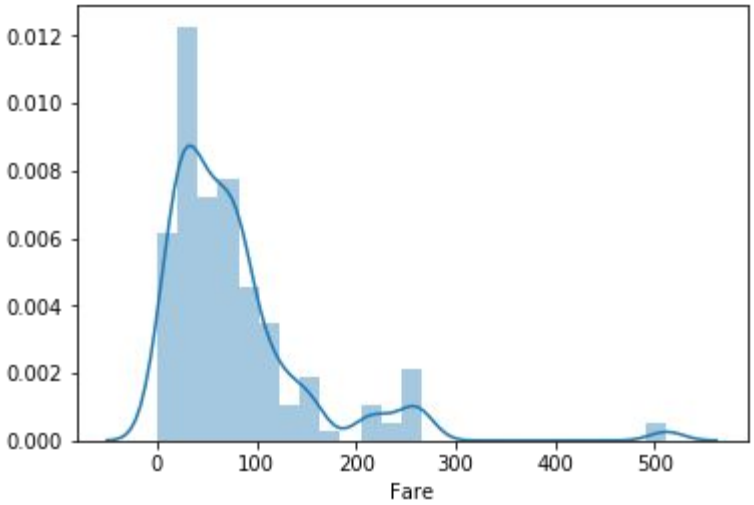
Distribution plot

- As the name suggests, It shows the statistical distribution of data
- Distribution plot is a variation of histogram that uses kernel smoothing to plot values, allowing for smoother distributions by smoothing out the noise

Distribution plot

Use distplot() function to create distribution plot. Let's see the distribution of the price of the ticket

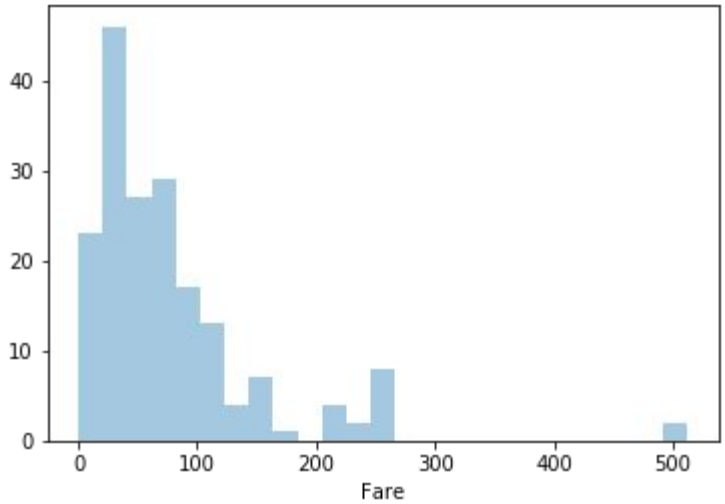
```
sns.distplot(df_titanic['Fare'])
plt.show()
```



Distribution plot

You can remove line from the previous plot by passing False as the parameter for the kde attribute

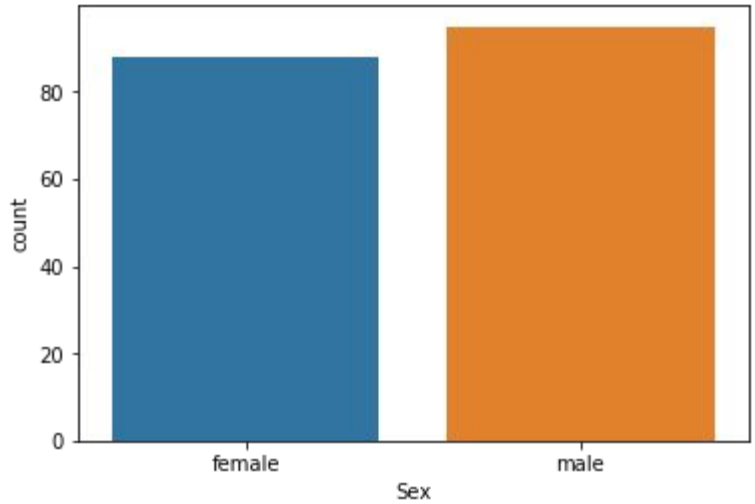
```
sns.distplot(df_titanic['Fare'], kde=False)
plt.show()
```



Count plot

Count plot is similar to the bar plot. However, it shows the count of the categories in a specific variable

```
sns.countplot(x='Sex', data=df_titanic)
plt.show()
```



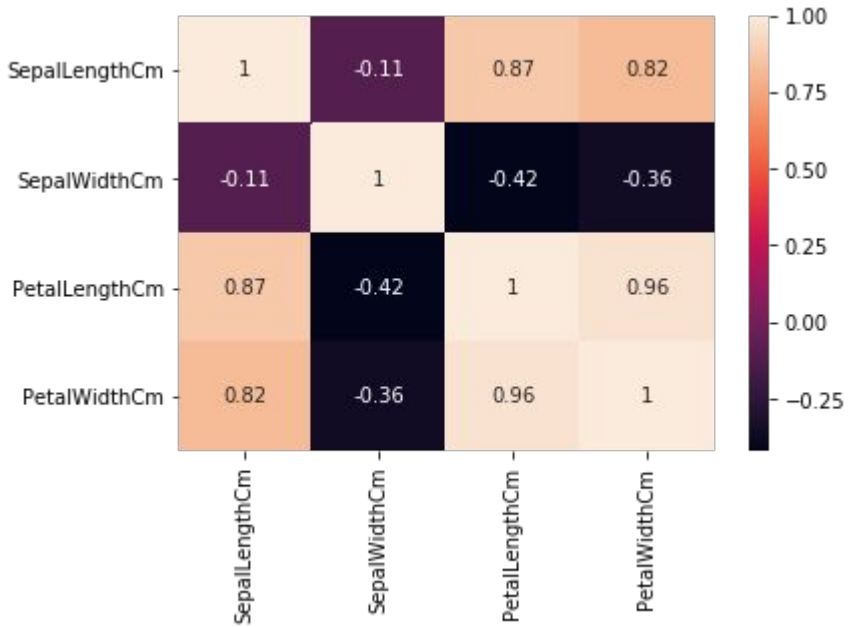
Heatmap

- A heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors
- Each square shows the correlation between the variables on each axis

Heatmap

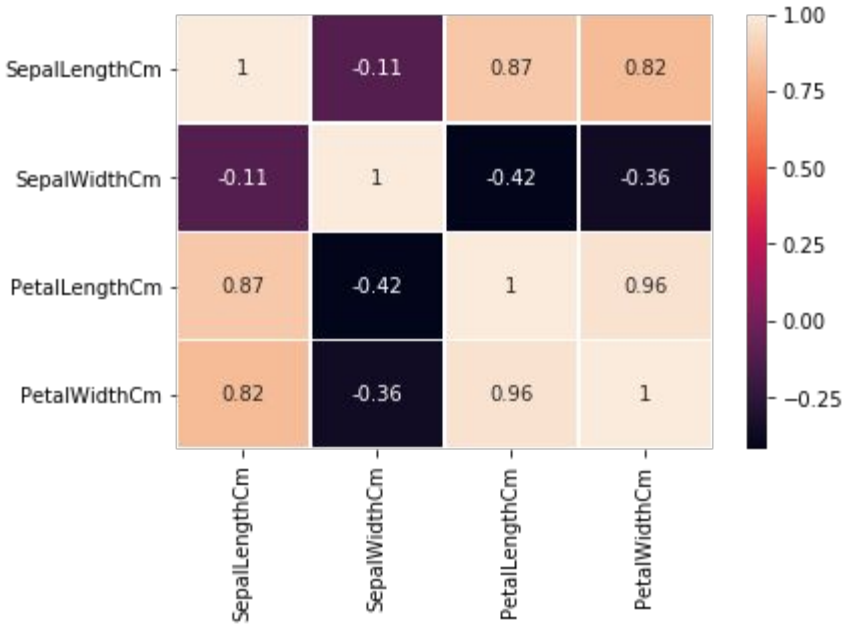
Use iris data to plot heatmap

```
# plot heatmap
# 'annot=True' returns the correlation values
sns.heatmap(df_iris.corr(), annot = True)

# display the plot
plt.show()
```



Add lines between the correlation cells

```
# plot heatmap
# 'linewidth' : add lines between each cells
sns.heatmap(df_iris.corr(), annot = True, linewidth=0.5)

# display the plot
plt.show()
```



Visualization using Plotly

Plotly

- Plotly allows us to make beautiful, interactive, explorable charts
- It is an open-source and browser-based graphing library
- Import the offline version of plotly

```
import plotly
plotly.offline.init_notebook_mode(connected=True)
```

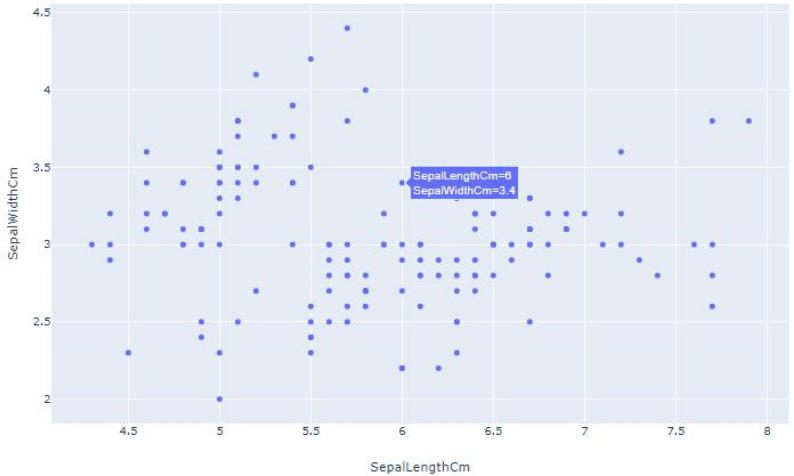
Boxplot using plotly

Use iris data to plot boxplot using plotly

```
# import the library
import plotly.express as px
fig = px.box(df_iris, y ='SepalLengthCm')
fig.show()
layout = Layout(width=50, height=50)
```

Scatter plot using plotly

```
fig = px.scatter(df_iris, x='SepalLengthCm', y='SepalWidthCm')
fig.show()
```



Thank You