

**New
syllabus
2022-23**

Chapter 2
Data
Visualization

Data visualization

"A picture is worth a thousand words". Most of us are familiar with this expression. Data visualization plays an essential role in the representation of both small and large-scale data. It especially applies when trying to explain the analysis of increasingly large datasets.

Data visualization is the discipline of trying to expose the data to understand it by placing it in a visual context. Its main goal is to distill large datasets into visual graphics to allow for easy understanding of complex relationships within the data.

Several data visualization libraries are available in Python, namely Matplotlib, Seaborn, and Folium etc.

Purpose of Data visualization

- Better analysis
- Quick action
- Identifying patterns
- Finding errors
- Understanding the story
- Exploring business insights
- Grasping the Latest Trends

Plotting library

Matplotlib is the whole python package/ library used to create 2D graphs and plots by using python scripts. **pyplot** is a module in matplotlib, which supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment for MatLab.

Pyplot provides the state-machine interface to the plotting library in matplotlib. It means that figures and axes are implicitly and automatically created to achieve the desired plot. For example, calling plot from pyplot will automatically create the necessary figure and axes to achieve the desired plot. Setting a title will then automatically set that title to the current axes object. The pyplot interface is generally preferred for non-interactive plotting.

Matplotlib – pyplot features

Following features are provided in matplotlib library for data visualization.

- **Drawing** – plots can be drawn based on passed data through specific functions.
- **Customization** – plots can be customized as per requirement after specifying it in the arguments of the functions. Like color, style (dashed, dotted), width; adding label, title, and legend in plots can be customized.
- **Saving** – After drawing and customization plots can be saved for future use.

How to plot in matplotlib

Steps to plot in matplotlib

- Install matplotlib by pip command -
`pip install matplotlib` in command prompt
- Create a .py & import matplotlib library in it
using - `import matplotlib.pyplot as plt`
statement
- Set data points in `plot()` method of `plt` object
- Customize plot through changing different parameters
- Call the `show()` method to display plot
- Save the plot/graph if required

Types of plot using matplotlib

- LINE PLOT
- BAR GRAPH
- HISTOGRAM

Matplotlib –line plot

Line Plot

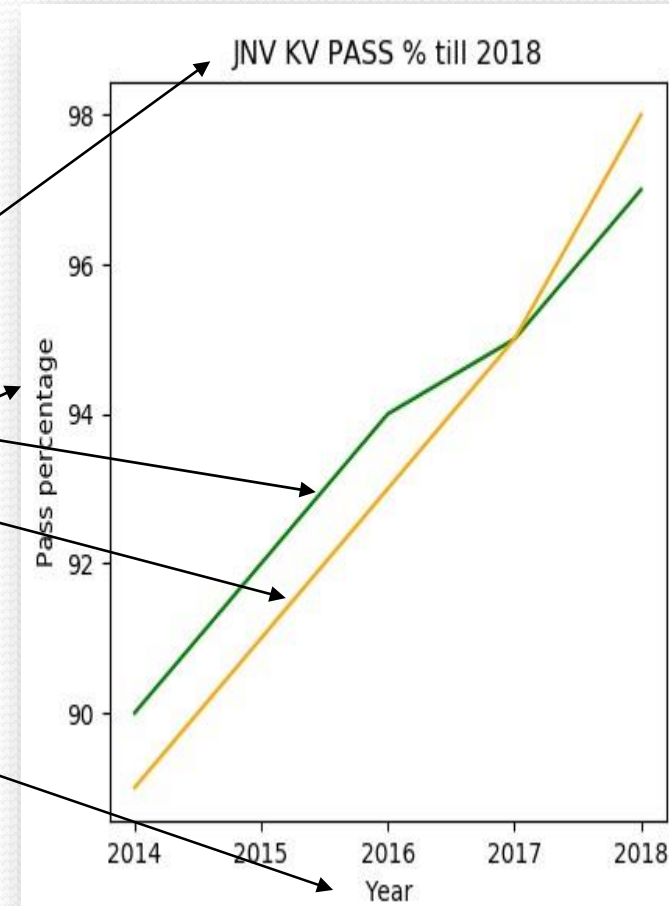
A line plot/chart is a graph that shows the frequency of data occurring along a number line.

The line plot is represented by a series of datapoints connected with a straight line. Generally line plots are used to display trends over time. A line plot or line graph can be created using the `plot()` function available in pyplot library. We can, not only just plot a line but we can explicitly define the grid, the x and y axis scale and labels, title and display options etc.

Matplotlib –line plot

E.G.PROGRAM

```
import numpy as np
import matplotlib.pyplot as plt
year = [2014,2015,2016,2017,2018]
jnvpasspercentage = [90,92,94,95,97]
kvpasspercentage = [89,91,93,95,98]
plt.plot(year, jnvpasspercentage, color='g')
plt.plot(year, kvpasspercentage, color='orange')
plt.xlabel('Year')
plt.ylabel('Pass percentage')
plt.title('JNV KV PASS % till 2018')
plt.show()
```



Note:- As many lines required call plot() function multiple times with suitable arguments.

Matplotlib –line plot

Line Plot customization

- **Custom line color**
`plt.plot(year, kvpasspercentage, color='orange')`
Change the value in color argument.like 'b' for blue,'r','c',.....
- **Custom line style**
`plt.plot([1,1.1,1,1.1,1], linestyle='-' , linewidth=4).`
set linestyle to any of '-' for solid line style, '--' for dashed, '-.' , ':' for dotted line
- **Custom line width**
`plt.plot('x', 'y', data=df, linewidth=22)`
set linewidth as required
- **Title**
`plt.title('JNV KV PASS % till 2018')` – Change it as per requirement
- **Lable** - `plt.xlabel('Year')` - change x or y label as per requirement
- **Legend** - `plt.legend(('jnv','kv'),loc='upper right',frameon=False)`
Change (),loc,frameon property as per requirement

Matplotlib – Bar Graph

Bar Graph

A graph drawn using rectangular bars to show how large each value is. The bars can be horizontal or vertical.

A bar graph makes it easy to compare data between different groups at a glance. Bar graph represents categories on one axis and a discrete value in the other. The goal bar graph is to show the relationship between the two axes. Bar graph can also show big changes in data over time.

Plotting with Pyplot

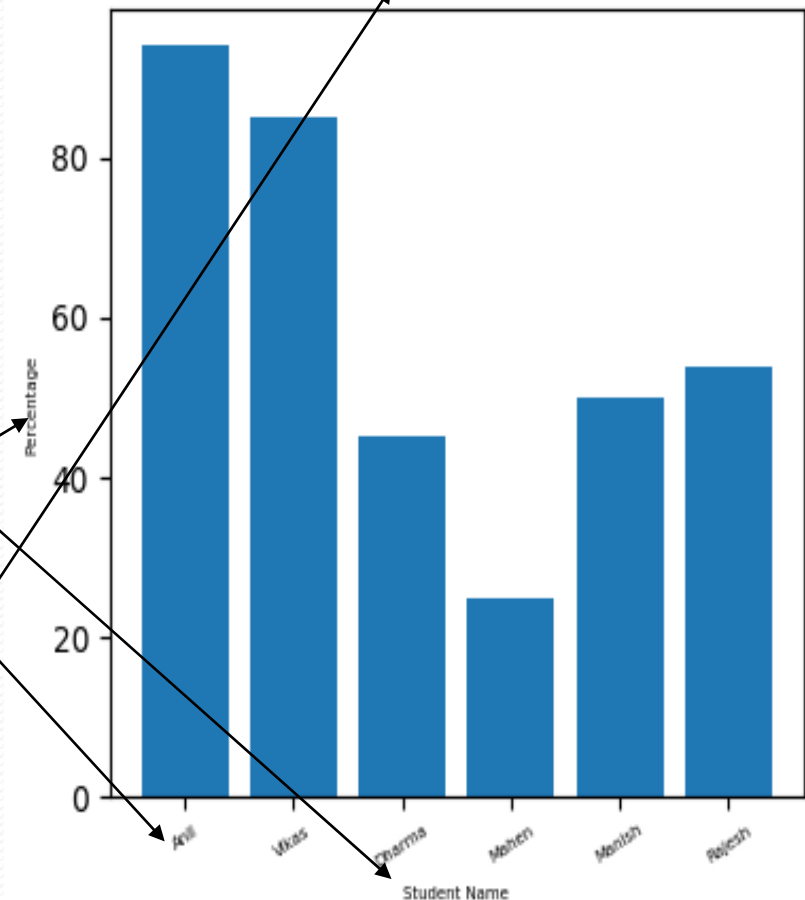
Plot bar graphs

e.g program

```
import matplotlib.pyplot as plt
import numpy as np
label = ['Anil', 'Vikas', 'Dharma', 'Mahen',
'Manish', 'Rajesh']
per = [94,85,45,25,50,54]
index = np.arange(len(label))
plt.bar(index, per)
plt.xlabel('Student Name', fontsize=5)
plt.ylabel('Percentage', fontsize=5)
plt.xticks(index, label, fontsize=5,
rotation=30)
plt.title('Percentage of Marks achieve by
student Class XII')
plt.show()
```

#Note – use barh () for horizontal bars

Percentage of Marks achieve by student Class XII



Matplotlib –Bar graph

Bar graph customization

- **Custom bar color**

`plt.bar(index, per,color="green",edgecolor="blue")`

Change the value in color,edgecolor argument.like 'b' for blue,'r','c',.....

- **Custom line style**

`plt.bar(index, per,color="green",edgecolor="blue",linewidth=4,linestyle='--')`

set linestyle to any of '-' for solid line style, '--' for dashed, '-.' , ':' for dotted

line

- **Custom line width**

`plt.bar(index, per,color="green",edgecolor="blue",linewidth=4)`

set linewidth as required

- **Title**

`plt.title('Percentage of Marks achieve by student Class XII')`

Change it as per requirement

- **Lable** - `plt.xlabel('Student Name', fontsize=5)`- change x or y label as per requirement
- **Legend** - `plt.legend(('jnv', 'kv'), loc='upper right', frameon=False)`

Matplotlib –Histogram

A histogram is a graphical representation which organizes a group of data points into user-specified ranges.

Histogram provides a visual interpretation of numerical data by showing the number of data points that fall within a specified range of values (“bins”). It is similar to a vertical bar graph but without gaps between the bars.

Matplotlib –Histogram

Histogram in Python –

```
import numpy as np
import matplotlib.pyplot as plt
data = [1,11,21,31,41]
plt.hist([5,15,25,35,45, 55], bins=[0,10,20,30,40,50, 60], weights=[20,10,45,33,6,8],
edgecolor="red")
plt.show()
```

#first argument of hist() method is position (x,y Coordinate) of weight, where weight is to be displayed.

No of coordinates must match with No of weight otherwise error will generate

#Second argument is interval

#Third argument is weight for bars

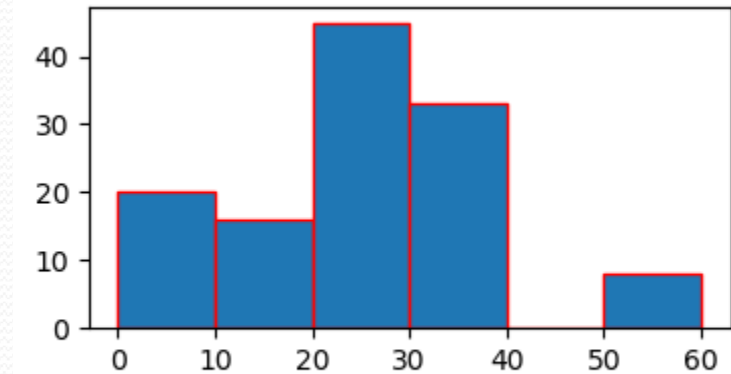
Matplotlib –Histogram

Histogram in Python –

For better understading we develop the same program with minor change .

```
import numpy as np
import matplotlib.pyplot as plt
data = [1,11,21,31,41]
plt.hist([5,15,25,35,15, 55], bins=[0,10,20,30,40,50,60],
edgecolor="red")
plt.show()
```

at interval(bin)40 to 50 no bar because we have not mentioned position from 40 to 50 in first argument(list) of hist method. Where as in interval 10 to 20 width is being Displayed as 16 (10+6 both weights are added) because 15 is twice In first argument.



Matplotlib –Histogram

Customization of Histogram –

By default bars of histogram is displayed in blue color but we can change it to other color with following code .

```
plt.hist([1,11,21,31,41, 51], bins=[0,10,20,30,40,50, 60], weights=[10,1,0,33,6,8], facecolor='y', edgecolor="red")
```

In above code we are passing 'y' as facecolor means yellow color to be displayed in bars.

To give a name to the histogram write below code before calling show()

```
plt.title("Histogram Heading")
```

Edge color and bar color can be set using following parameter in hist() method

```
edgecolor='#E6E6E6',color='#EE6666
```

 .color value can be rgb in hexadecimal form

For x and y label below code can be written

Matplotlib – How to save plot

For future use we have to save the plot. To save any plot `savefig()` method is used. Plots can be saved like `pdf`, `svg`, `png`, `jpg` file formats.

```
plt.savefig('line_plot.pdf')
```

```
plt.savefig('line_plot.svg')
```

```
plt.savefig('line_plot.png')
```

Parameter for saving plots .e.g.

```
plt.savefig('line_plot.jpg', dpi=300, quality=80, optimize=True,  
progressive=True)
```

Which Export Format to Use?

The export as vector-based SVG or PDF files is generally preferred over bitmap-based PNG or JPG files as they are richer formats, usually providing higher quality plots along