Data visualisation

- Introduction to Matplotlib
- Installing Matplotlib
- How to use Matplotlib
- The Relation between – Matplotlib, Pyplot and Python
- Create a simple plot
- Adding elements to a plot
- Making Multiple Plots in One Figure
- Create Subplots
- Figure Object
- Axes Object
- Different Types of Plots
- Saving Plot

Data Visualization Introduction

- Data visualization is the process of converting raw data into easily understandable pictorial representation, that enables fast and effective decisions.
- It is both an Art and a Science.
- Data visualization is a strategy where we represent the quantitative information in a graphical form.

Why Data Visualization?

- Data visualization is the first step of analysis work.
- It gives intuitive understanding of data.
- Helps you to see data in certain meaningful patterns.
- Visual representations enhances the human cognitive process.

Benefits of Data Visualization

- Data visualization allow users to see several different perspectives of data.
- Data visualization makes it possible to interpret vast amounts of data.
- It offers ability to note expectations in data.
- Exploring trends within a database through visualization by letting analysts navigate through data and visually orient themselves to the patterns in the data.

Data Visualization Techniques

Some of the popular techniques are:

- Pie chart
- Bar graph
- Histogram
- Wordle or tag cloud
- Tree map
- Scatter plot
- Line chart
- Bubble chart etc.

- There are multiple tools and technologies available in the industry for data visualisation, python being the most used. Python offers multiple libraries for data visualisation, few of the popular graphic libraries are:
- **Matplotlib**
- **Seaborn**
- **Pandas visualisation**
- **Plotly**

Installing Matplotlib

- There are multiple ways to install the matplotlib library. The easiest way to install matplotlib is to download the Anaconda package. Matplotlib is default installed with Anaconda package and does not require any additional steps.
- Download anaconda package from the official site of Anaconda
- To install matplotlib, go to anaconda prompt and run the following command
- pip install matplotlib
- or
- conda install matplotlib
-
- Verify whether the matplotlib is properly installed using the following command in Jupyter notebook
- import matplotlib
- matplotlib.__version__

How to use Matplotlib

- Before using matplotlib, we need to import the package. This can be done using the ‘import’ method in Jupyter notebook. PyPlot is the graphical module in matplotlib which is mostly used for data visualisation, importing PyPlot is sufficient to work around data visualisation.
- # import matplotlib library as mpl

- import matplotlib as mpl #import the pyplot module from matplotlib as plt (short name used for referring the object)
- import matplotlib.pyplot as plt

The relation between – Matplotlib, Pyplot and Python

- Python Is a very popular programming language, used for web development, mathematics and statistical analysis. Python works on most of the platforms and is also simple to use.
- Python has multiple libraries used for specific purposes, below libraries are mostly used for visualisation and data analysis.
  - NumPy
  - Pandas
  - Matplotlib
  - Seaborn
  - Plotly
  - SciKit-Learn
- As you observe one of the packages is matplotlib which is developed using python. This library is very widely used for data visualisations.

Create a Simple Plot

- Here we will be depicting a basic plot using some random numbers generated using NumPy. The simplest way to create a graph is using the ‘plot()’ method. To generate a basic plot,
- we need two axes (X) and (Y), and we will generate two random numbers using the ‘linspace()’ method from Numpy.
- # import the NumPy package
- import numpy as np

- need two axes (X) and (Y), and we will generate two random numbers using the ‘linspace()’ method from Numpy.
- # import the NumPy packageimport numpy as np
- # Create a basic plotplt.plot(x,y)

Adding Elements to Plot

- set different elements to the plot generated above
- Add title using ‘plt.title’
- Add x-label using ‘plt.xlabel’
- Add y-label using ‘plt.ylabel’
- set x-axis limits using ‘plt.xlim’
- set y-axis limits using ‘plt.ylim’
- Add legend using ‘plt.legend’

- Let’s, add few more elements to the plot like colour, markers, line customisation.
- add color, style, width to line elementplt.plot(x, y, c = 'r', linestyle = '--', linewidth=2)
- add markers to the plot, marker has different elements i.e., style, color, size etc.,plt.plot (x, y, marker='*', markersize=3, c=’g’)
- add grid using grid() methodPlt.grid(True)
- add legend and labelplt.legend()

- Plots could be customised at three levels:
- Colours
  - b – blue
  - c – cyan
  - g – green
  - k – black
  - m – magenta
  - r – red
  - w – white
  - y – yellow
  - Can use Hexadecimal, RGB formats
- Line Styles
  - ‘-‘ : solid line
  - ‘- -‘: dotted line
  - ‘- .’: dash-dot line
  - ‘:’ – dotted line

- Marker Styles
  - . – point marker
  - , – Pixel marker
  - v – Triangle down marker
  - ^ – Triangle up marker
  - < – Triangle left marker
  - > – Triangle right marker
  - 1 – Tripod down marker
  - 2 – Tripod up marker
  - 3 – Tripod left marker
  - 4 – Tripod right marker
  - s – Square marker
  - p – Pentagon marker
  - * – Star marker

- Other configurations
  - color or c
  - linestyle
  - linewidth
  - marker
  - markeredgewidth
  - markeredgecolor
  - markerfacecolor
  - markersize

Making Multiple Plots in One Figure

- Let’s plot two lines sin(x) and cos(x) in a single figure and add legend to understand which line is what.
- lets plot two lines Sin(x) and Cos(x)
- loc is used to set the location of the legend on the plot
- label is used to represent the label for the line in the legend
- generate the random number
- x= np.arange(0,1500,100)
- plt.plot(np.sin(x),label='sin function x')
- plt.plot(np.cos(x),label='cos functon X’)
  plt.legend(loc='upper right')

Create Subplots

- subplots are used to create multiple plots in a single figure
- let's create a single subplot first following by adding more subplots
- x = np.random.rand(50)
- y = np.sin(x*2)
- need to create an empty figure with an axis as below, figure and axis are two separate objects in matplotlib

- fig, ax = plt.subplots() #add the charts to the plot ax.plot(y)
- Let’s add multiple plots using subplots() function# Give the required number of plots as an argument in subplots(), below function creates 2 subplotsfig, axs = plt.subplots(2)
- create datax=np.linspace(0,100,10)
- assign the data to the plot using axsaxs[0].plot(x, np.sin(x**2))axs[1].plot(x, np.cos(x**2))

- add a title to the subplot figurefig.suptitle('Vertically stacked subplots')

- Create horizontal subplots# Give two arguments rows and columns in the subplot() function
- subplot() gives two dimensional array with 2*2 matrix
- need to provide ax also similar 2*2 matrix as below fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
- add the data to the plotsax1.plot(x, x**2)ax2.plot(x, x**3)ax3.plot(x, np.sin(x**2))ax4.plot(x, np.cos(x**2))

- add titlefig.suptitle('Horizontal plots')

another simple way of creating multiple subplots as below, using axsfig, axs = plt.subplots(2, 2)

add the data referring to row and columnaxs[0,0].plot(x, x**2,'g')axs[0,1].plot(x, x**3,'r')axs[1,0].plot(x, np.sin(x**2),'b')axs[1,1].plot(x, np.cos(x**2),'k')

add titlefig.suptitle('matrix sub plots')

Figure Object

- Matplotlib is an object-oriented library and has objects, calluses and methods
- The object figure is a container for showing the plots and is instantiated by calling figure() function.
- ‘plt.figure()’ is used to create the empty figure object in matplotlib. Figure has the following additional parameters.

- Figsize – (width, height) in inches
- Dpi – used for dots per inch (this can be adjusted for print quality)
- facecolor
- edgecolor
- linewidth

- let’s create a figure object
- change the size of the figure is ‘figsize = (a,b)’ a is width and ‘b’ is height in inches
- create a figure object and name it as fig

- fig = plt.figure(figsize=(4,3))
- create a sample dataX = np.array([1,2,3,4,5,6,8,9,10])Y = X**2
- plot the figureplt.plot(X,Y)
- let’s change the figure size and also add additional parameters like facecolor, edgecolor, linewidth
- fig = plt.figure(figsize=(10,3),facecolor='y',edgecolor='r',linewidth=5)

Axes Object

- Left – position of axes from left of figure
- bottom – position of axes from the bottom of figure
- width – width of the chart
- height – height of the chart

- Other parameters that can be used for the axes object are:
- Set title using ‘ax.set_title()’
- Set x-label using ‘ax.set_xlabel()’
- Set y-label using ‘ax.set_ylabel()’

- create a sample datay = [1, 5, 10, 15, 20,30]x1 = [1, 10, 20, 30, 45, 55]x2 = [1, 32, 45, 80, 90, 122]
- create the figurefig = plt.figure()
- add the axesax = fig.add_axes([0,0,2,1])l1 = ax.plot(x1,y,'ys-') l2 = ax.plot(x2,y,'go--')
- add additional parametersax.legend(labels = ('line 1', 'line 2'), loc = 'lower right') ax.set_title("usage of add axes function")ax.set_xlabel('x-axix')ax.set_ylabel('y-axis')plt.show()

Different Types of Matplotlib Plots

- bar chart
- line chart
- pie chart
- scatter chart
- bubble chart
- waterfall chart
- circular area chart
- stacked bar chart etc.,

Bar Graph

- ***Function:***
- The function used to show bar graph is ‘plt.bar()’
- The bar() function expects two lists of values one on x-coordinate and another on y-coordinate
- ***Customisations:***
- plt.bar() function has the following specific arguments that can be used for configuring the plot.
- Width, Color, edge colour, line width, tick_label, align, bottom,
- Error Bars – xerr, yerr

- *Example:*
- # lets create a simple bar chart# x-axis is shows the subject and y -axis shows the markers in each subject subject = ['maths','english','science','social','computer']marks =[70,80,50,30,78]plt.bar(subject,marks)plt.show()
- #let's do some customizations#width – shows the bar width and default value is 0.8#color – shows the bar color#bottom – value from where the y – axis starts in the chart i.e., the lowest value on y-axis shown#align – to move the position of x-label, has two options 'edge' or 'center'#edgecolor – used to color the borders of the bar#linewidth – used to adjust the width of the line around the bar#tick_label – to set the customized labels for the x-axis plt.bar(subject,marks,color ='g',width = 0.5,bottom=10,align ='center',edgecolor='r',linewidth=2,tick_label=subject)
- # errors bars could be added to represent the error values referring to an array value# here in this example we used standard deviation to show as error barsplt.bar(subject,marks,color ='g',yerr=np.std(marks))
- # to plot horizontal bar plot use plt.barh() functionplt.barh(subject,marks,color ='g',xerr=np.std(marks))

Scatter Plot

- Any relationship exists between the two columns
- + ve Relationship
- Or -Ve relationship
- ***Function:***
- The function used for the scatter plot is ‘plt.scatter()’
- ***Customizations:***
- plt.scatter() function has the following specific arguments that can be used for configuring the plot.
- size – to manage the size of the points
- color – to set the color of the points
- marker – type of marker
- alpha – transparency of point
- norm – to normalize the data (

Histogram

- Center of the distribution
- Spread of the distribution
- Shape of the distribution
- Peak of the distribution

- ***Function:***
- The function used for scatter plot is ‘plt.hist()’
- ***Customisations:***
- plt.hist() function has the following specific arguments that can be used for configuring the plot.
- bins – number of bins
- color
- edgecolor
- alpha – transparency of the color
- normed
- xlim – to set the x-limits
- ylim – to set the y-limits
- xticks, yticks
- facecolor, edgecolor, density

Saving Plot

- Saving plot as an image using ‘savefig()’ function in matplotlib. The plot can be saved in multiple formats like .png, .jpeg, .pdf and many other supporting formats.