MATPLOTLIB

1. **import matplotlib.pyplot as plt** # To import matplotlib library.
2. **%matplotlib inline** # To show the plots automatically without need to enter plt.show( ).
3. **from matplotlib import style , style.use("ggplot")** **#** For style purpose.
4. **plt.legend(['name'],loc= 'upper left' /1/2/3/4, title='legend title)** # To show the labels of the lines.
5. **plt.grid(True, color = ' ')** **#** To show the grid.
6. **plt.xticks([ ] ,rotation='vertical'/90,size=10, color='red') , plt.yticks(np.arange(0,20,2), [ ])**
   #To shows the ticks on x-axis and y-axis.
7. **plt.xlabel('Year') , plt.ylabel('Sales')** # To show the labels on x-axis and y-axis.
8. **plt.title('Year Sales Diagram', fontsize=24)** # To show the title on the graph.
9. **plt.axes().get_xaxis().set_visible(False)** **#** To remove the x-axis.
10. **plt.axes().get_yaxis().set_visible(False)** **#** To remove the y-axis.
11. **plt.xlim(100) , plt.ylim(20)** # To set the starting point of graph, x=100 & y=20.
12. **plt.figure(figsize=(10, 20)) ; plt.rcParams['figure.figsize]=(18,20)** # To adjust the figure size.
13. **plt.savefig('plot_name.png', bbox_inches='tight', format= 'pdf')** # To save the graph plot.
14. **plt.rcParams['lines.linestyle'] = ':'** # To change the line style of graph.
15. **df.plot(kind= 'optional')** # To draw a plot of all columns at once.
16. **Line Plot** - plt.plot(x-elements, y-elements , 'bo-')
    # Line graphs are used to show value of some items over time.
    plt.plot(df['Year'] , df['Sales']) , plt.plot([1,2,3,4], [21,34,56,39]) ,
    plt.plot(a, b, color='g', linewidth=2 , makersize = 10)
    'bo-' → b = blue color , o = marker (*,.+sd) , - = Draws line

17. **Bar Plot** - plt.bar(x-elements, y-elements)
    # For Categorical Data….Bar graphs are used to make comparison between different categories.
    plt.bar(x , y , color = 'bkgrc' , label= 'New' , width = 0.7), df.plot.bar(stacked=True)

18. **Horizontal Bar Chart** - plt.barh(x-elements, y-elements)
    # Bars will raise from x-value and goes up-to y values. df.plot.barh(stacked=True)

19. **Scatter Plot** - plt.scatter(x-elements, y-elements , color = 'r', s = 20 , edgecolor= 'red' . style='*-')
    # It shows data as a collection of points. Predictor(Indep.) on x-axis & Target(Dep.) on y-axis.

20. **Bubble Plot** - plt.scatter(x-elements, y-elements , color = np.random.random(length of column) , s = 20 , edgecolor= 'red' . style='*-')
    # Same as scatter plot.

21. **Histogram** - plt.hist(data, bins= , color = ' ' , rwidth =)
    # Show frequency of data divided into intervals. It tends to show the distribution by grouping segments together.

22. Stack Plot - plt.stackplot(list1, list2, list3, list4 , color = 'mcbr')
It is generated by plotting different datasets vertically on top of one another.

23. Pie Chart - plt.pie(slices, labels= activities, colors = 'bryg', startangle= , shadow=True, explode=(0,0,0.1,0.2), autopct= '%1.1f%%', pctdistance=0.75) .
Slices = [12,15,20,10] , activities = ['eating', ' sleeping' , 'working', 'playing'].
# Explode – To cut the slices out. Autopct – To show the % on the chart using string format. pctdistance – Distance of % from center
# Compare parts of data to the whole. It shows the size of items(wedges) in one data series proportional to the sum of the items.

24. Box Plot - df.boxplot( ) , sns.boxplot(x='Cat_col' , y='Num_col', data=df)
This graph represents the min, max, median, first quartile & third quartile in the dataset. It shows how well distributed the data is in a dataset.

25. Heat Map - plt.pcolor(df, cmap='RdBu') , plt.colorbar( )
The darker shades of the chart represent higher values than the lighter shade.

26. 3D Charts

```
from mpl_toolkits.mplot3d import axes3d          # To add a subplot to an existing 2d plot.
chart = plt.figure()
chart3d = chart.add_subplot(111, projection='3d')
# Create some test data
x,y,z = axes3d.get_test_data(0.08)
# Plot a wireframe
chart3d.plot_wireframe(x,y,z, color='r', rstride=15, cstride=10)
plt.show()
```

27. Graph from Pandas directly :
df.plot(x = 'Year', y = 'Sales' , kind = " line/scatter/box/area/stack/pie/bar", figsize = (25,4), color=['red', 'black', 'green', 'yellow', 'orange']).
df.Col_name.plot(style='*-' , figsize = (25,4)
Pandas can make graphs by calling plot directly from the DF (using df.plot( )). Plots can be called by defining plot kinds.

28. Time Series Plot - df.plot( ) , where x = df.datetime_index , y = df.column

29. Plotting two sets of data : plt.scatter(x-elements1 , y-elements1) , plt.scatter(x-elements1, y-elements2)

30. plt.fill_between(x-elements1, y-elements1, y-elements2, facecolor='green', alpha=1.5)
Filling the space between datasets.

31. To draw the month/year wise sales on graph –
months = range(1,13) , plt.bar(months , df.groupby('month/year_col').sum( ))

32. To check the relationship between two columns :

sns.relplot(x = ‘Col_1’ , y = ‘Col_2’ , data = df_name)
sns.relplot(x = ‘Col_1’ , y = ‘Col_2’ , hue = ‘Col_3’ , data = df_name , kind = ‘line’ , height = 5 , aspect = 3)
sns.catplot(x = ‘Col_1’ , y = 'Col_2’ , data = df_name)

33. sns.pairplot(df_name) - It shows the relationship between all the columns with each other (correlation).

34. df.Col_name.plot(kind = ‘ ’) – To draw a plot between the indexes and a column.
df.condition.plot( )

35. sns.countplot(df.Col_name) # To show the value-counts in the form of bar graph.

36. To draw a Sine Wave Plot
x = np.arange(0, 3 * np.pi , 0.1)
y = np.sin(x)
plt.plot(x,y)

37. Adding Annotations (Naming on plot sheet wrt to points)
plt.annotate(xy=[2,1] , s = ‘first annotation at x=2 & y=1’)
plt.annotate(xy=[4,6] , s = ‘second annotation at x=4 & y=6’)

38. To draw multiple lines on one graph
plt.plot(x,y, marker='o', color='g'), plt.plot(x,z , marker='*', color='red')
plt.plot(x,t , marker='.', color='black')

39. To draw Linear Regression Graph
import seaborn as sns
sns.regplot(x = df.Col_x , y = df.Col_y) ; # To see the correlation between two variables.

40. To draw Residual Plot # It represents the error between the actual values.
sns.residplot(df.Col_x , df.Col_y)

41. To draw Distribution Plot # It counts the predicted value versus the actual value.
ax1 = sns.distplot(df.Col_y , hist=False , color= ‘r’, label= ‘Actual Value’)
sns.distplot(yhat , hist=False , color= ‘b’ , label= ‘Fitted Values’ , ax = ax1)

42. Pearson Correlation Heatmap : sns.heatmap(df.corr( ) , vmin=-1, vmax=1, center=0)

43. To draw the Normal Distribution curve –
mu = 0.5 , sigma = 0.1
s = np.random.normal(mu , sigma , 1000)
# Create the bins & histogram
count , bins , ignored = plt.hist(s , 20 , density = True)
Plot the distribution curve

```
plt.plot(bins , 1/(sigma*np.sqrt(2*np.pi)) *
        np.exp( - (bins - mu)**2 / (2*sigma**2)) , linewidth = 3, color = 'y')
plt.show()
```

44. To draw Binomial Distribution curve –

```
from scipy.stats import binom , import seaborn as sns
binom.rvs(size=10, n=20 , p=0.8)
data_binom = binom.rvs(n=20 , p=0.8, loc=0, size=1000)
ax = sns.distplot(data_binom,
                kde=True,
                color='blue',
                hist_kws={"linewidth":25 , 'alpha':1})
ax.set(xlabel = 'Binomial' , ylabel = 'Frequency')
plt.show( )
```

45. To draw Poisson Distribution

```
from scipy.stats import poisson , import seaborn as sns
data_binom = poisson.rvs(mu=4, size=10000)
ax = sns.distplot(data_binom,
                kde=True,
                color='r',
                hist_kws = {'linewidth':25, 'alpha':1})
ax.set(xlabel = 'Position' , ylabel='Frequency')
plt.show( )
```

46. To draw Bernoulli Distribution

```
from scipy.stats import Bernoulli , import seaborn as sns
data_bern = bernoulli.rvs(size=100, p=0.6)
ax = sns.distplot(data_bern,
                kde = True,
                color='c',
                hist_kws={'linewidth':25, 'alpha':1})
ax.set(xlabel='Bernoulli', ylabel='Frequency')
plt.show( )
```

47. To draw a Chi-Square Distribution

```
from scipy import stats , import numpy as np
x = np.linspace(0, 10, 100)
fix, ax = plt.subplots(1,1)
linestyles = [':','--','-.','-']
deg_of_freedom = [1,4,7,6]
for df, ls in zip(deg_of_freedom, linestyles):
   ax.plot(x, stats.chi2.pdf(x, df), linestyle=ls)
plt.xlim(0, 10) , plt.ylim(0, 0.4)
plt.xlabel('Value') , plt.ylabel('Frequency') , plt.title('Chi-Square Distribution')
```

```
plt.show()
```

48. **Insert Image in Jupyter Notebook** -- Convert the Cell to MarkDown > Edit Tab > Insert Image > Run