Line Plot & Important Matplotlib Functions

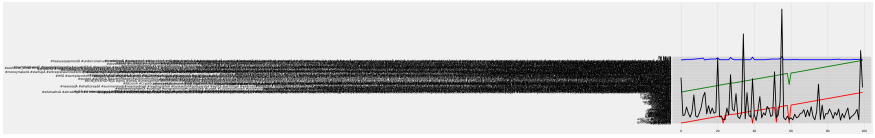
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("/content/instagram_reach.csv", encoding = "latin-1")
print(df.head())
```

```
   Unnamed: 0  S.No             USERNAME  \
0           0     1         mikequindazzi
1           1     2       drgorillapaints
2           2     3    aitrading_official
3           3     4  opensourcedworkplace
4           4     5           crea.vision

                                             Caption  Followers  \
0  Who are #DataScientist and what do they do? >>...       1600
1  We all know where itâ€™s going. We just have t...        880
2  Alexander Barinov: 4 years as CFO in multinati...        255
3                                               sfad        340
4  Ever missed a call while your phone was chargi...        304

                                          Hashtags Time since posted  Likes
0  #MachineLearning #AI #DataAnalytics #DataScien...          11 hours    139
1  Â #deckÂ .#macÂ #macintosh#sayhelloÂ #appleÂ #...           2 hours     23
2  #whoiswhoÂ #aitradingÂ #aiÂ #aitradingteam#ins...           2 hours     25
3  #iotÂ #cre#workplaceÂ #CDOÂ #bigdataÂ #technol...           3 hours     49
4  #instamachinelearningÂ #instabigdata#instamark...           3 hours     30
```

```
plt.plot(df["USERNAME"], "-r", label="USERNAME")
plt.plot(df["Hashtags"], "-g", label="Hashtags")
plt.plot(df["Time since posted"], "-b", label="Time since posted")
plt.plot(df["Likes"], "-k", label="Likes")
plt.show()
```

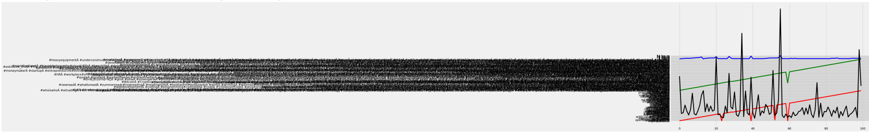
```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
```

```
plt.figure(figsize=(15, 10)) # Customizing Figure Size
plt.plot(df["USERNAME"], "-r", label="Username")
plt.plot(df["Hashtags"], "-g", label="Hashtags")
plt.plot(df["Time since posted"], "-b", label="Time since posted")
plt.plot(df["Likes"], "-k", label="Likes")
plt.show()
```



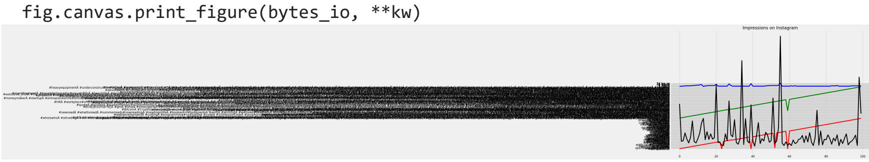
```
plt.style.use('fivethirtyeight')          # for customizing theme
plt.figure(figsize=(15, 10))
plt.plot(df["USERNAME"], "-r", label="USERNAME")
plt.plot(df["Hashtags"], "-g", label="Hashtags")
plt.plot(df["Time since posted"], "-b", label="Time since posted")
plt.plot(df["Likes"], "-k", label="Likes")
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
```



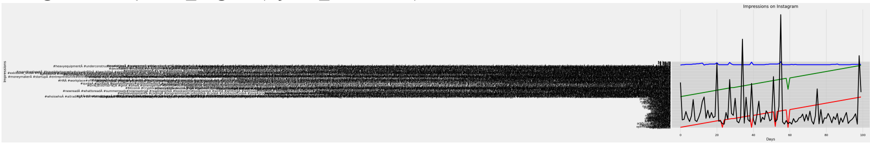
```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(df["USERNAME"], "-r", label="USERNAME")
plt.plot(df["Hashtags"], "-g", label="Hashtags")
plt.plot(df["Time since posted"], "-b", label="Time since posted")
plt.plot(df["Likes"], "-k", label="Likes")
plt.title("Impressions on Instagram")                # for adding a title
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
```



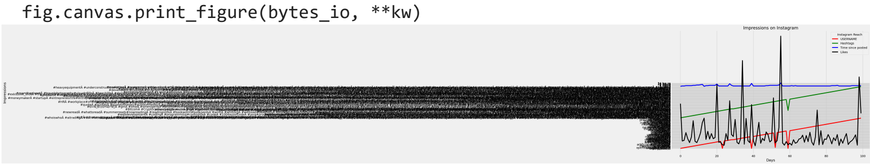
```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(df["USERNAME"], "-r", label="USERNAME")
plt.plot(df["Hashtags"], "-g", label="Hashtags")
plt.plot(df["Time since posted"], "-b", label="Time since posted")
plt.plot(df["Likes"], "-k", label="Likes")
plt.title("Impressions on Instagram")
plt.xlabel("Days") # adding label on xaxis
plt.ylabel("Impressions") # adding label on yaxis
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
```



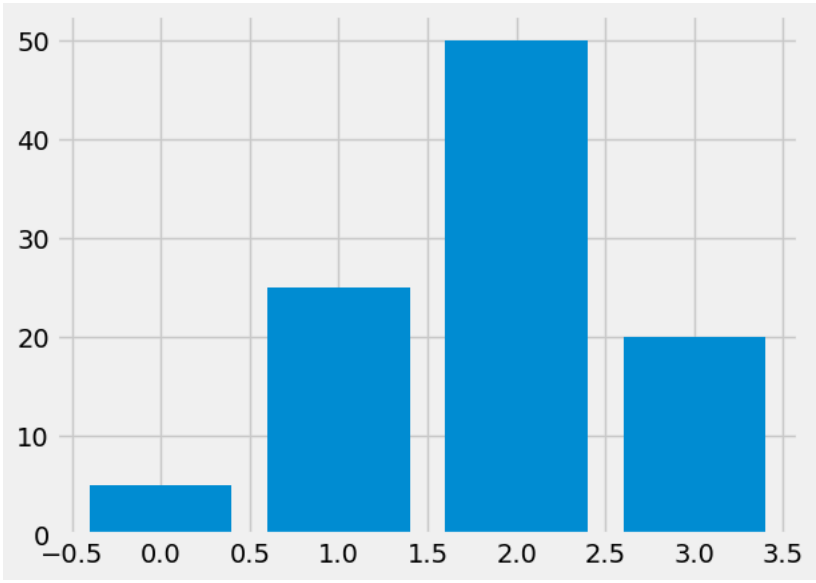
```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(df["USERNAME"], "-r", label="USERNAME")
plt.plot(df["Hashtags"], "-g", label="Hashtags")
plt.plot(df["Time since posted"], "-b", label="Time since posted")
plt.plot(df["Likes"], "-k", label="Likes")
plt.title("Impressions on Instagram")
plt.xlabel("Days")
plt.ylabel("Impressions")
plt.legend(title="Instagram Reach") # for adding legend with a title
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
```

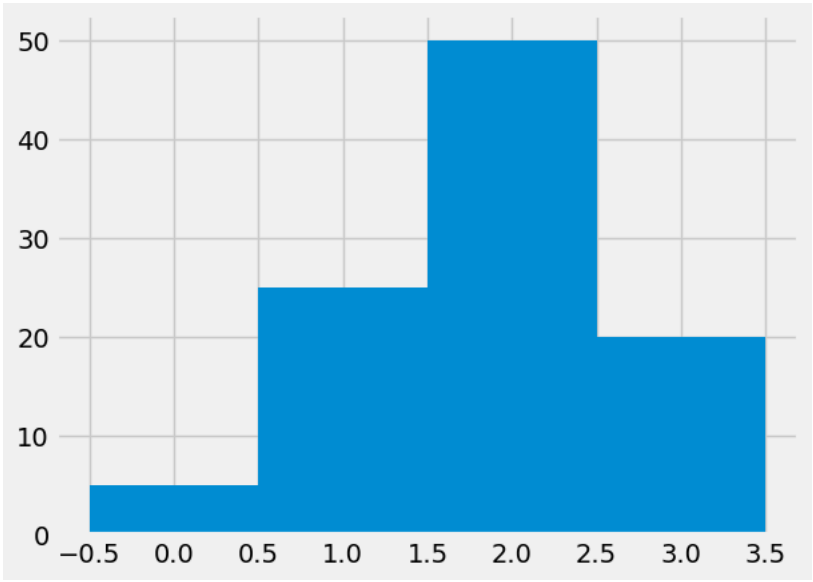


BAR PLOT

```
import pandas as pd
import matplotlib.pyplot as plt
data = [5., 25., 50., 20.]
plt.bar(range(len(data)), data)
plt.show()
```

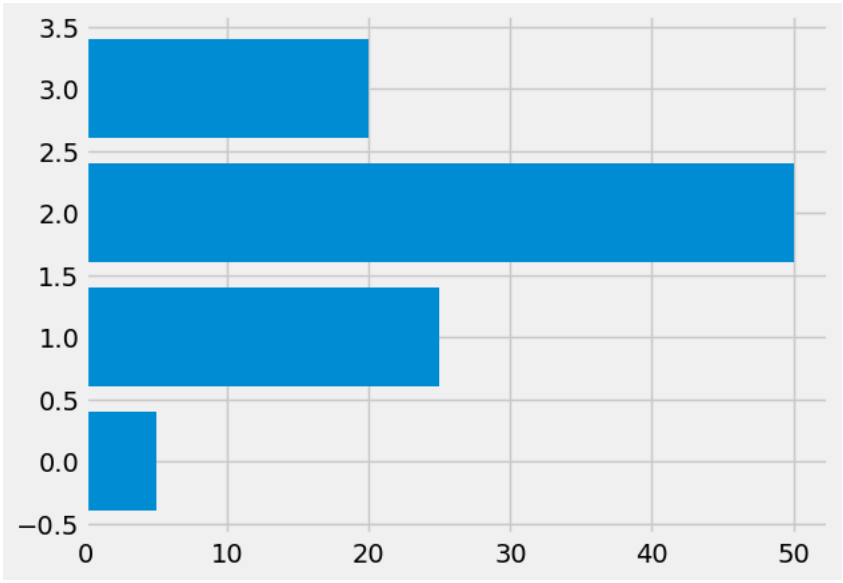



```
data = [5., 25., 50., 20.]
plt.bar(range(len(data)), data, width=1.)
plt.show()
```

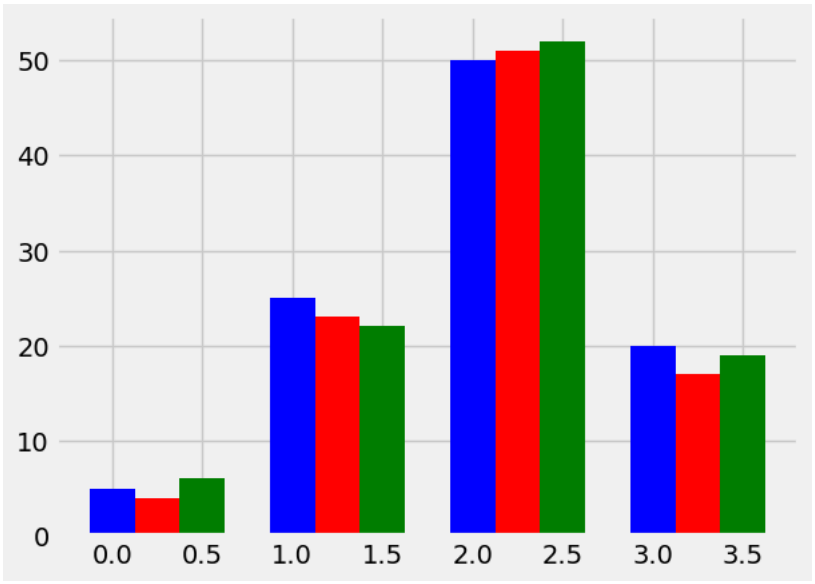


Horizontal Bar plots

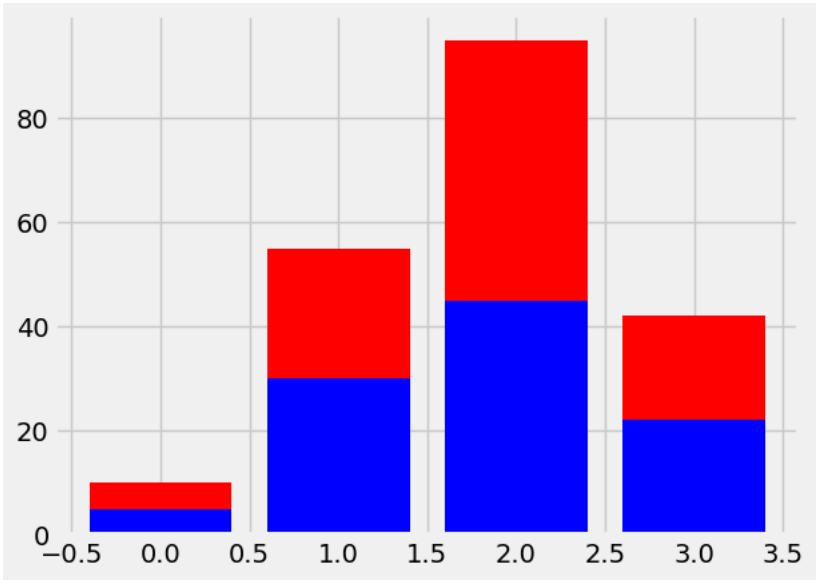
```
import pandas as pd
import matplotlib.pyplot as plt
data = [5., 25., 50., 20.]
plt.barh(range(len(data)), data)
plt.show()
```




```
import numpy as np
data = [[5., 25., 50., 20.],
        [4., 23., 51., 17.],
        [6., 22., 52., 19.]]
x = np.arange(4)
plt.bar(x + 0.00, data[0], color='b', width=0.25)
plt.bar(x + 0.25, data[1], color='r', width=0.25)
plt.bar(x + 0.50, data[2], color='g', width=0.25)
plt.show()
```

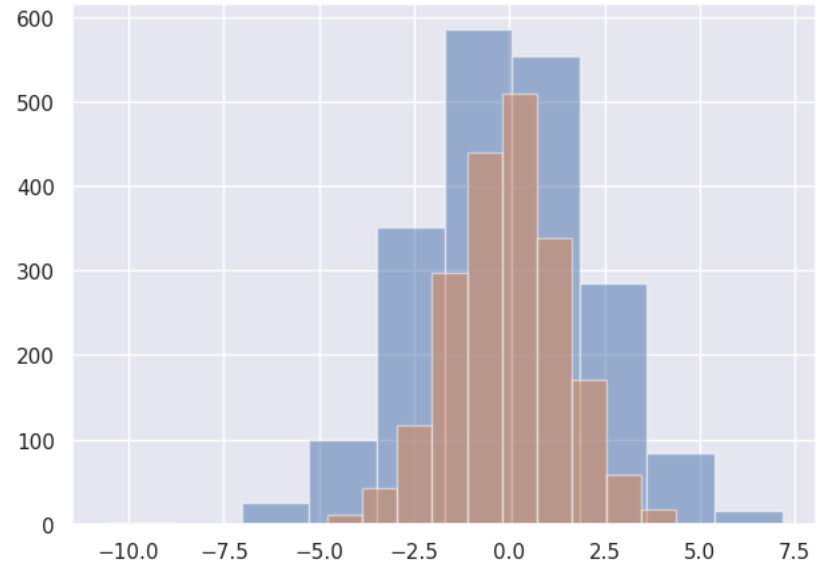



```
a = [5., 30., 45., 22.]
b = [5., 25., 50., 20.]
x = range(4)
plt.bar(x, a, color='b')
plt.bar(x, b, color='r', bottom=a)
plt.show()
```



Histogram and density plots

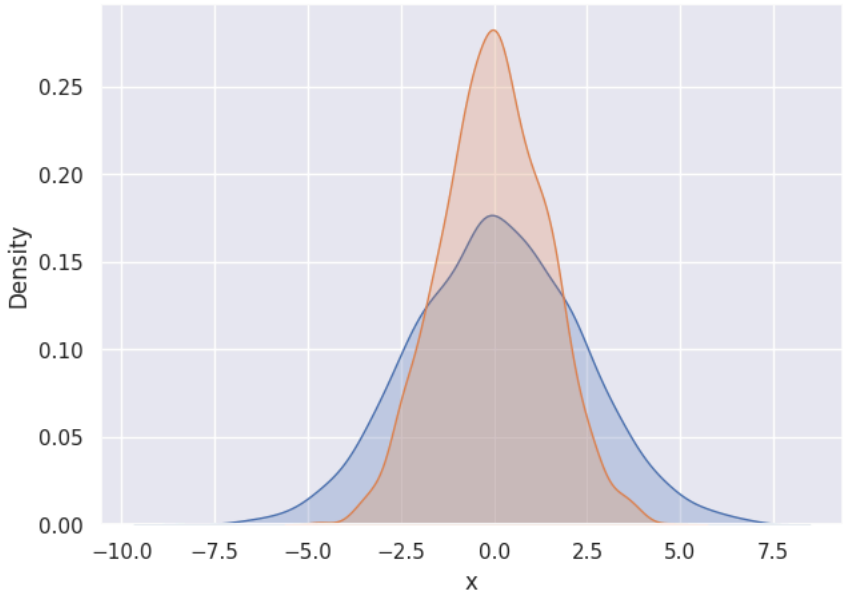
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])
plt.hist(data["x"], alpha=0.5)
plt.hist(data["y"], alpha=0.5)
plt.show()
```



Density plot

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])
sns.kdeplot(data["x"], shade=True)
sns.kdeplot(data["y"], shade=True)
plt.show()
```

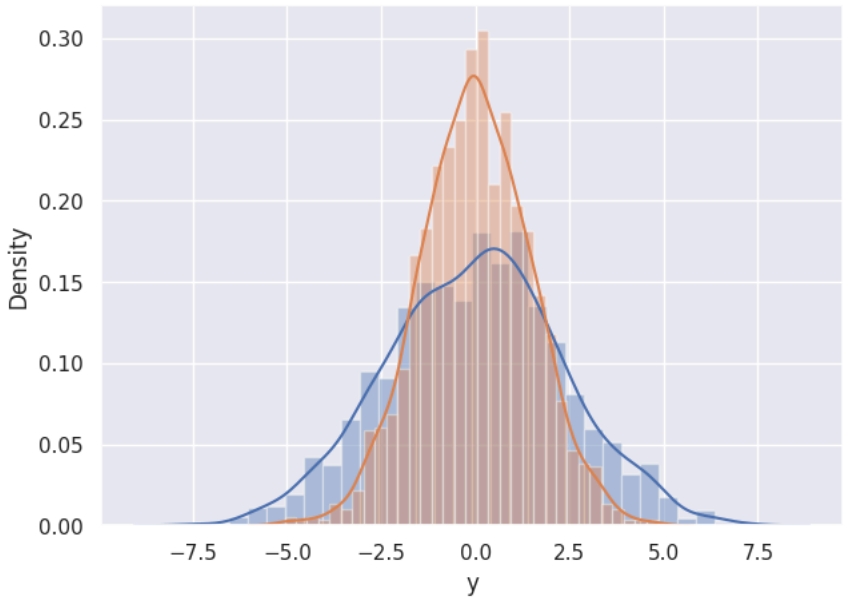
```
<ipython-input-30-cbd2e714dd7e>:9: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data["x"], shade=True)
<ipython-input-30-cbd2e714dd7e>:10: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data["y"], shade=True)
```



Histogram and density plot

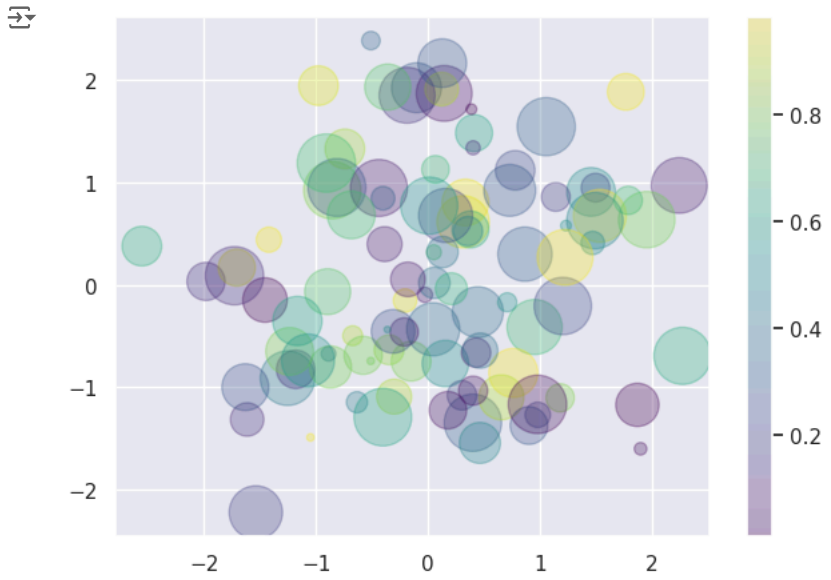
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])
sns.distplot(data['x'])
sns.distplot(data['y'])
plt.show()
```

```
<ipython-input-31-dcc4ca8fcfb4>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['x'])
<ipython-input-31-dcc4ca8fcfb4>:10: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['y'])
```

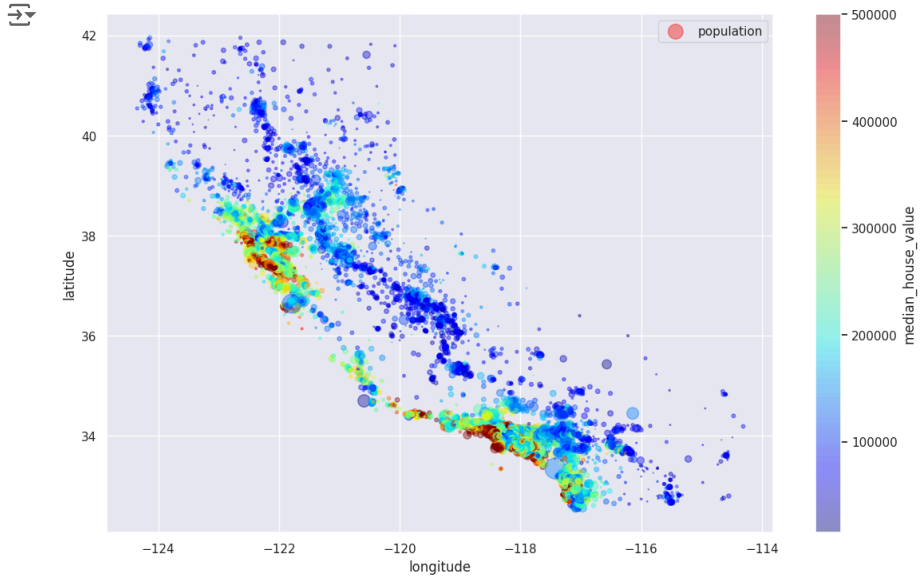


Scatter plot

```
import numpy as np
import matplotlib.pyplot as plt
rng = np.random.RandomState(0)
x = rng.randn(100)
y = rng.randn(100)
colors = rng.rand(100)
sizes = 1000 * rng.rand(100)

plt.scatter(x, y, c=colors, s=sizes, alpha=0.3,
            cmap='viridis')
plt.colorbar()
plt.show()
```




```
import pandas as pd
housing = pd.read_csv("https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.csv")
housing.plot(kind='scatter', x='longitude', y='latitude', alpha=0.4, s=housing['population']/100, label='population',
figsize=(12, 8), c='median_house_value', cmap=plt.get_cmap('jet'), colorbar=True)
plt.legend()
plt.show()
```



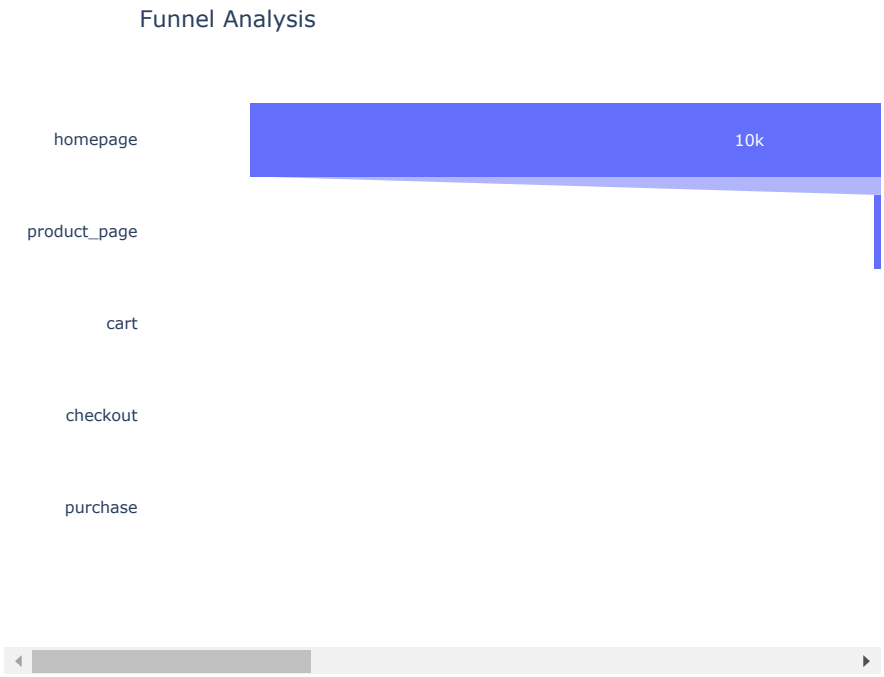
User funnels

```
#importing csv file
file_path = '/content/user_data.csv'

import pandas as pd

data = pd.read_csv(file_path)
```

```
print(data["stage"].value_counts())
```

```
stage
homepage        10000
product_page     5000
cart             1500
checkout          450
purchase          225
Name: count, dtype: int64
```

```
import plotly.graph_objects as go
import plotly.io as pio
pio.templates.default = "plotly_white"

#define the funnel stages
funnel_stages = ['homepage', 'product_page', 'cart', 'checkout', 'purchase']

#calculate the number of users and conversions for each stage
num_users = []
num_conversions = []

for stage in funnel_stages:
    stage_users = data[data['stage'] == stage]
    num_users.append(len(stage_users))
    num_conversions.append(stage_users['conversion'].sum())

#create a funnel chart
fig = go.Figure(go.Funnel(
    y=funnel_stages,
    x=num_users,
    textposition='inside',
    textinfo='value',
    name='Users'
))

fig.add_trace(go.Funnel(
    y=funnel_stages,
    x=num_conversions,
    textposition='inside',
    textinfo='value',
    name='Conversions'
))

fig.update_layout(
    title='Funnel Analysis',
    funnelmode='stack'
)

fig.show()
```



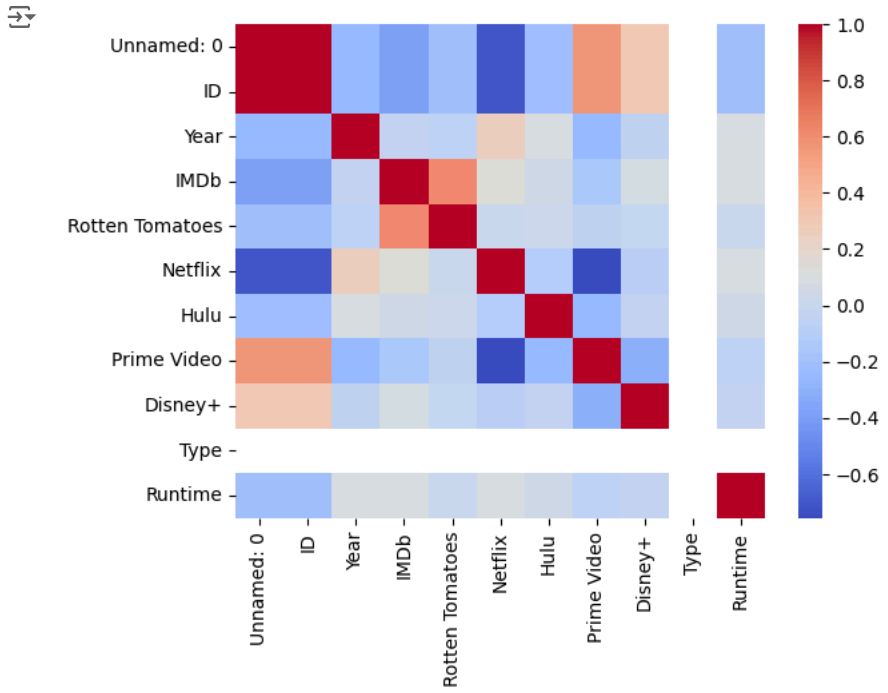
Heatmap using Python

In Data Science, a heatmap is used to understand the relationship between different features in a dataset. It represents numbers in the form of a coloured pallet such that darker shades represent a high degree of relationship between the features and the lighter shades represent a low degree of relationship between the features.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("/content/MoviesOnStreamingPlatforms_updated.csv", encoding = "latin-1")
print(df.head())
```

```
   Unnamed: 0  ID                         Title  Year  Age  IMDb  \
0           0   1                     Inception  2010  13+   8.8
1           1   2                    The Matrix  1999  18+   8.7
2           2   3  Avengers: Infinity War  2018  13+   8.5
3           3   4      Back to the Future  1985   7+   8.5
4           4   5  The Good, the Bad and the Ugly  1966  18+   8.8

  Rotten Tomatoes  Netflix  Hulu  Prime Video  Disney+  Type  \
0             87%        1     0            0        0     0
1             87%        1     0            0        0     0
2             84%        1     0            0        0     0
3             96%        1     0            0        0     0
4             97%        1     0            1        0     0

                      Directors                            Genres  \
0               Christopher Nolan  Action,Adventure,Sci-Fi,Thriller
1  Lana Wachowski,Lilly Wachowski                     Action,Sci-Fi
2        Anthony Russo,Joe Russo           Action,Adventure,Sci-Fi
3                 Robert Zemeckis           Adventure,Comedy,Sci-Fi
4                    Sergio Leone                           Western

                      Country                Language  Runtime
0  United States,United Kingdom  English,Japanese,French    148.0
1                United States                 English    136.0
2                United States                 English    149.0
3                United States                 English    116.0
4     Italy,Spain,West Germany                 Italian    161.0
```

```
import pandas as pd
movies = pd.read_csv("MoviesOnStreamingPlatforms_updated.csv")
movies['Rotten Tomatoes'] = movies["Rotten Tomatoes"].str.replace("%", "").astype(float)
movies_numeric = movies.select_dtypes(include=['float64', 'int64'])
correlations = movies_numeric.corr(method='pearson')

import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(correlations, cmap="coolwarm")
plt.show()
```

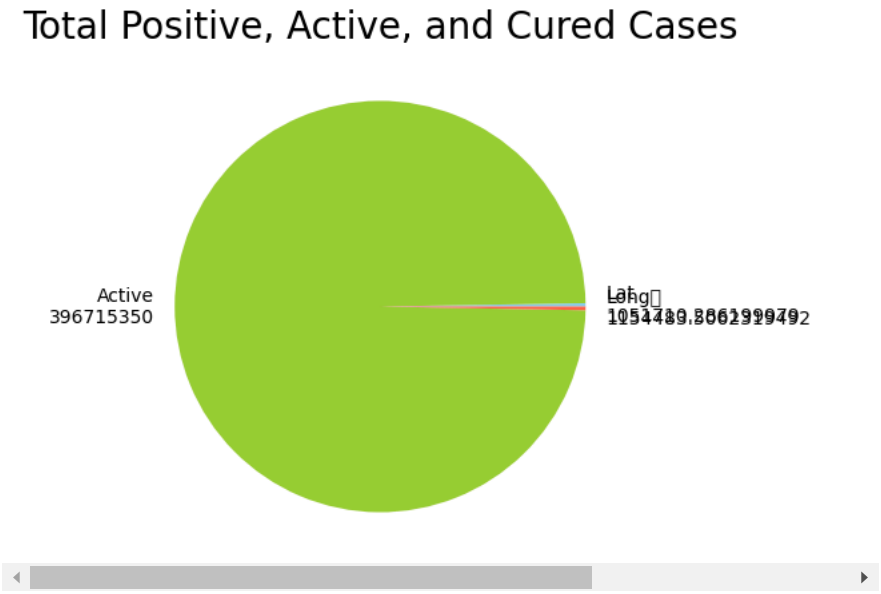


Pie Charts using Python

A pie chart is created by dividing a circle into two or more sections depending on the number of entities we want to visualize. It is used to analyze the proportion of each entity among all the entities.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("/content/covid_19_clean_complete.csv")
group_size = [sum(df.Lat), sum(df.Active), sum(df.Long)]
group_labels = ["Lat\n"+str(sum(df.Lat)),
                "Active\n"+str(sum(df.Active)),
                "Long   \n"+str(sum(df.Long ))]
custom_colors = ["skyblue", "yellowgreen", 'tomato']
plt.figure(figsize=(5, 5))
plt.pie(group_size, labels=group_labels, colors=custom_colors)
plt.rc('font', size=12)
plt.title("Total Positive, Active, and Cured Cases", fontsize=20)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
```

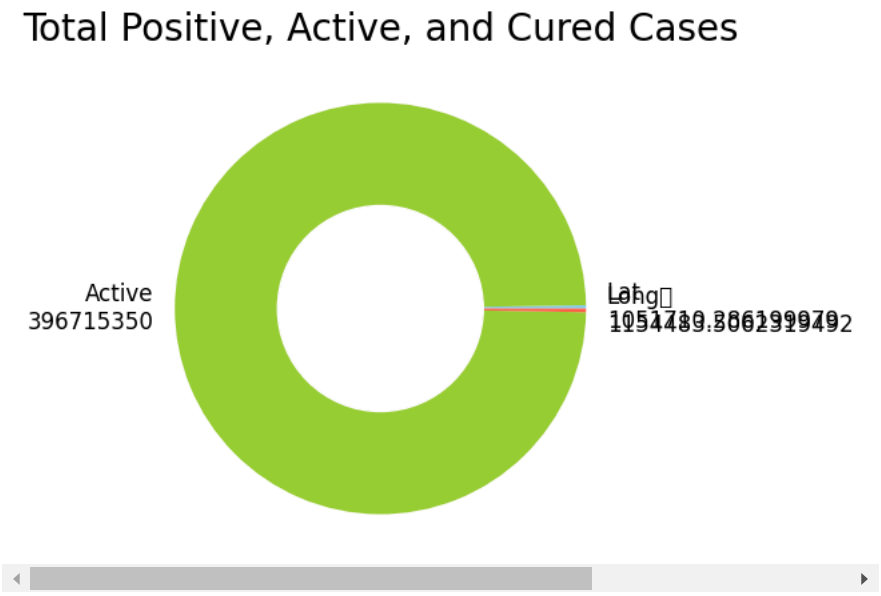


Donut Plot with Python

The Donut plot is similar to a pie chart, except it has a hole in the middle similar to a donut.

```
import seaborn as sns
import matplotlib.pyplot as plt
group_size = [sum(df.Lat), sum(df.Active), sum(df.Long)]
group_labels = ["Lat\n"+str(sum(df.Lat)),
                "Active\n"+str(sum(df.Active)),
                "Long   \n"+str(sum(df.Long ))]
custom_colors = ["skyblue", "yellowgreen", 'tomato']
plt.figure(figsize=(5, 5))
plt.pie(group_size, labels=group_labels, colors=custom_colors)
central_circle = plt.Circle((0, 0), 0.5, color='white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size=12)
plt.title("Total Positive, Active, and Cured Cases", fontsize=20)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning:
  fig.canvas.print_figure(bytes_io, **kw)
```



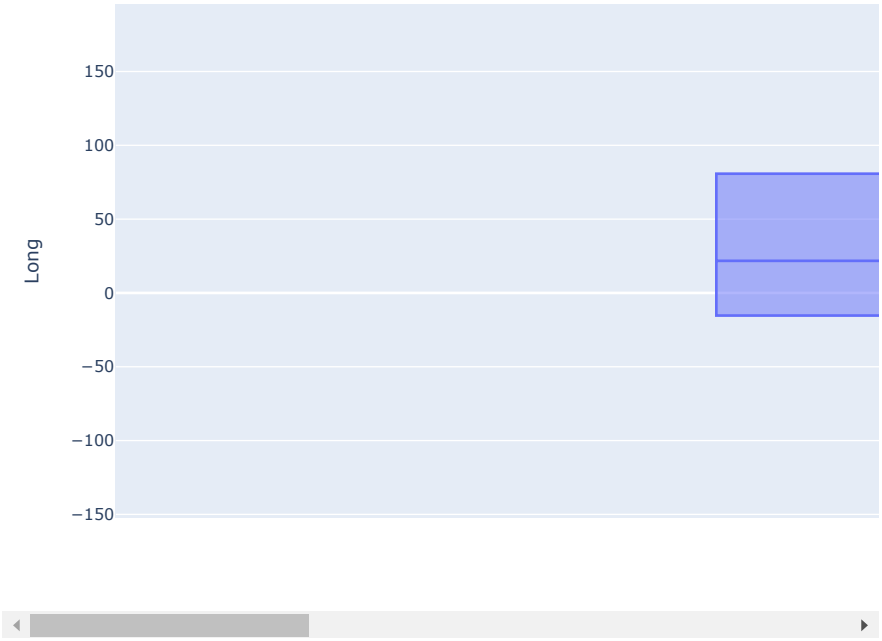
Box Plot using Python

A box plot is a statistical data visualization technique for analyzing the distribution and patterns of numerical data points of a dataset. It represents quartile 1, quartile 3, median, maximum and minimum data points of a feature which helps to understand the distribution of the numerical values of a dataset.

```
import pandas as pd
data = pd.read_csv("/content/covid_19_clean_complete.csv")
print(data.head())
```

```
  Province/State Country/Region       Lat       Long        Date  Confirmed  \
0            NaN    Afghanistan  33.93911  67.709953  2020-01-22          0
1            NaN        Albania  41.15330  20.168300  2020-01-22          0
2            NaN        Algeria  28.03390   1.659600  2020-01-22          0
3            NaN        Andorra  42.50630   1.521800  2020-01-22          0
4            NaN         Angola -11.20270  17.873900  2020-01-22          0

   Deaths  Recovered  Active             WHO Region
0       0          0       0  Eastern Mediterranean
1       0          0       0                 Europe
2       0          0       0                 Africa
3       0          0       0                 Europe
4       0          0       0                 Africa
```

```
import plotly.express as px
fig = px.box(data, y="Long")
fig.show()
```



Time Series Graph using Python

A time-series graph is a line plot that displays trends or patterns over a dataset collected over an interval of time. For example, when you visualize a line plot of the daily sales made by a business, you are visualizing a time-series graph. It is one of the most important data visualizations for every data scientist.

```
import pandas as pd
import yfinance as yf
import datetime
from datetime import date, timedelta
today = date.today()

d1 = today.strftime("%Y-%m-%d")
end_date = d1
d2 = date.today() - timedelta(days=360)
d2 = d2.strftime("%Y-%m-%d")
start_date = d2

data = yf.download('AAPL',
                   start=start_date,
                   end=end_date,
                   progress=False)
print(data.head())
```

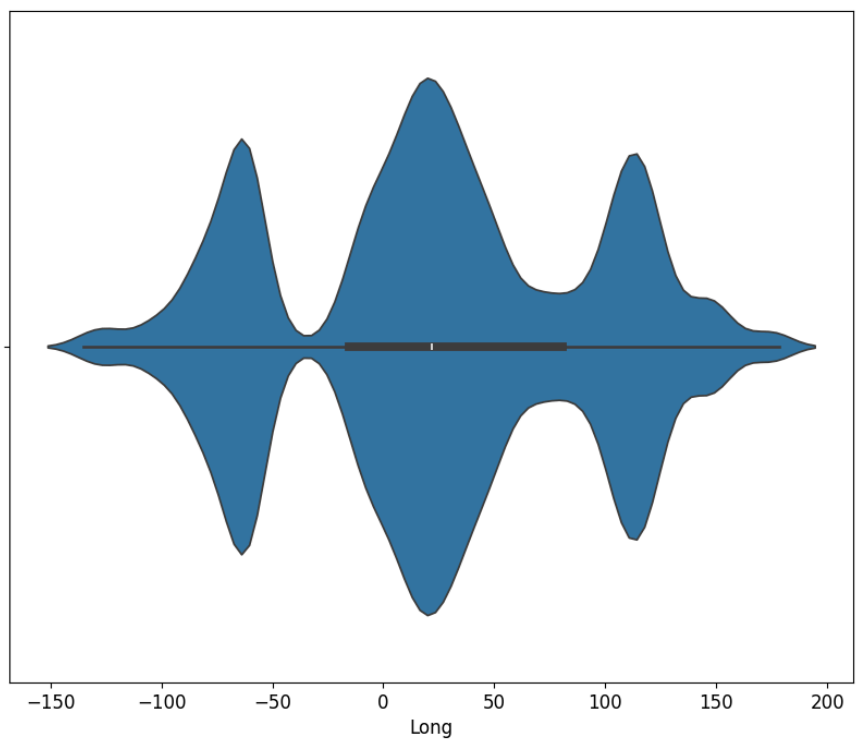
```
                  Open        High         Low       Close   Adj Close  \
Date
2023-05-10  173.020004  174.029999  171.899994  173.559998  172.638519
2023-05-11  173.850006  174.589996  172.169998  173.750000  172.827484
2023-05-12  173.619995  174.059998  171.000000  172.570007  171.891205
2023-05-15  173.160004  173.210007  171.470001  172.070007  171.393173
2023-05-16  171.990005  173.139999  171.800003  172.070007  171.393173

               Volume
Date
2023-05-10  53724500
2023-05-11  49514700
2023-05-12  45497800
2023-05-15  37266700
2023-05-16  42110300
```

```
import plotly.express as px
figure = px.line(data, x = data.index, y = "Close")
figure.show()
```

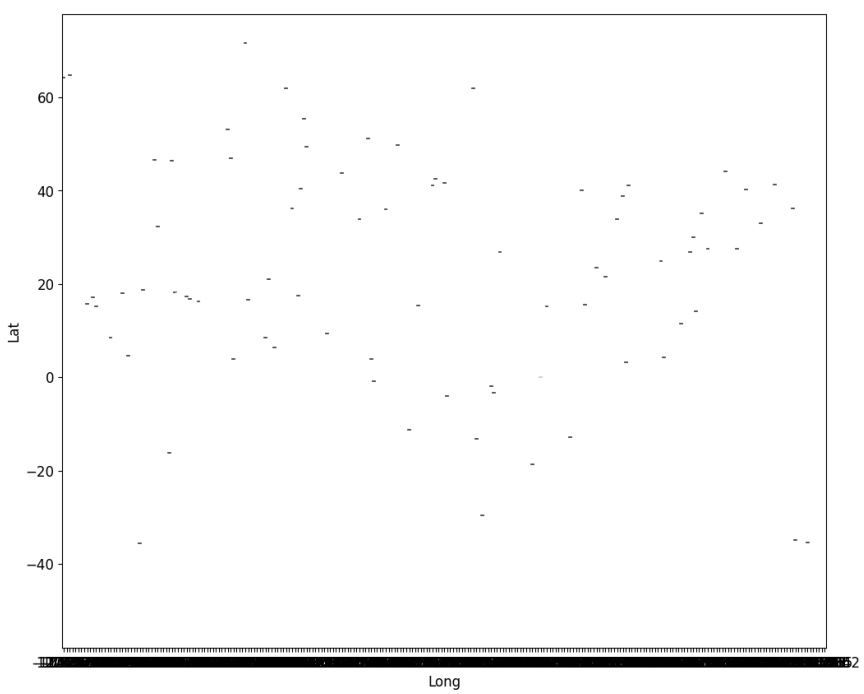
Violin Plot using Python

A violin plot is used to visualize and compare the distribution of quantitative data over several levels of categorical features. It is very useful to visualize several distributions in a dataset at a time.

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("/content/covid_19_clean_complete.csv")
plt.figure(figsize=(10, 8))
sns.violinplot(x=df["Long"])
plt.show()
```



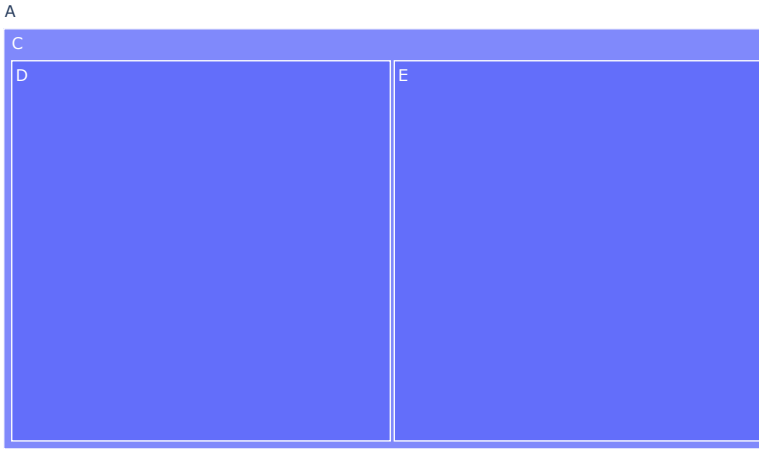
```
plt.figure(figsize=(12, 10))
sns.violinplot(x="Long", y="Lat", data=df)
plt.show()
```



Treemap using Python

A treemap is used to visualize hierarchical data as a set of nested rectangles. It is a data visualization tool for displaying data structured in a tree structure using nested rectangles.

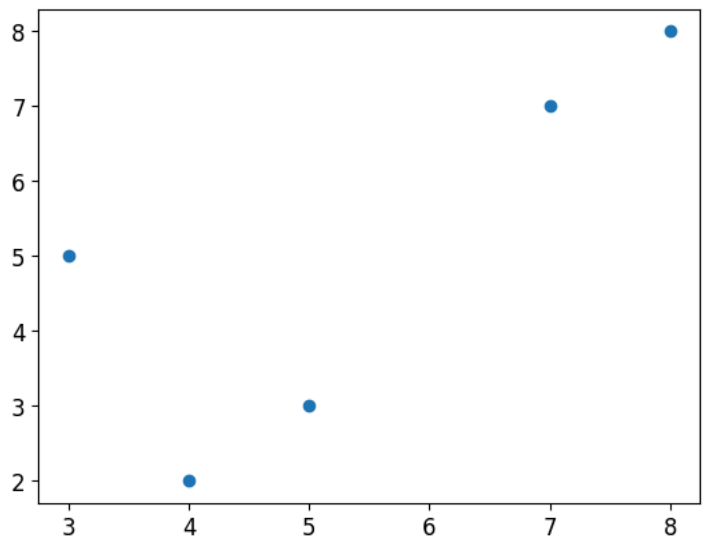
```
import plotly.graph_objects as go

fig = go.Figure(go.Treemap(
    labels = ["A","B", "C", "D", "E", "F", "G", "H", "I"],
    parents = ["", "A", "A", "C", "C", "A", "A", "G", "A"]
))

fig.show()
```



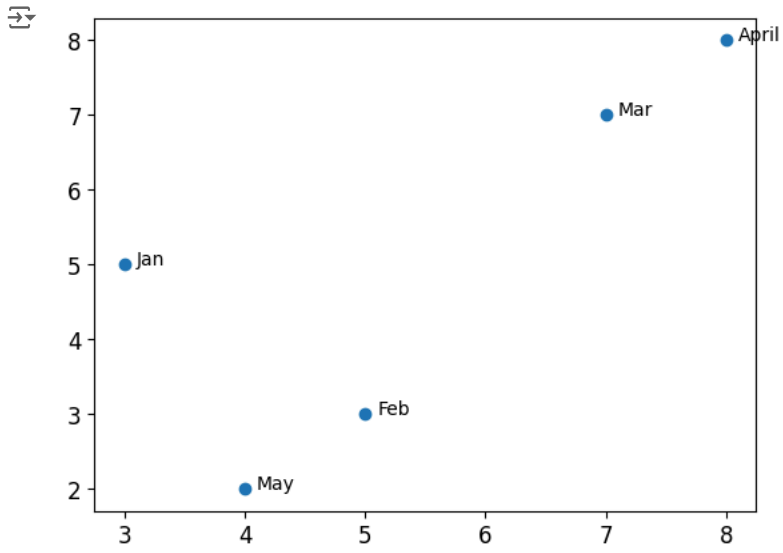
Plotting Annotations using Python

Plotting annotations while viewing graphs is considered a good practice because it makes your graphs self-explanatory. Sometimes it can be difficult to understand which data points indicate which characteristic especially in a scatter plot.

```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("/content/covid_19_clean_complete.csv")
plt.scatter(x, y)
plt.show()
```




```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("/content/covid_19_clean_complete.csv")
labels = ["Jan", "Feb", "Mar", "April", "May"]
plt.scatter(x, y)
for i, j in enumerate(labels):
    plt.annotate(j, (x[i]+0.10, y[i]), fontsize=10)
plt.show()
```



Word Cloud from a Pandas DataFrame in Python

A word cloud is a data visualization technique that shows the most used words in large font and the least used words in small font. It helps to get an idea about your text data, especially when working on problems based on natural language processing.

```
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
from wordcloud import STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv("/content/covid_19_clean_complete.csv")
print(data.head())
```

```
  Province/State Country/Region       Lat       Long        Date  Confirmed  \
0            NaN    Afghanistan  33.93911  67.709953  2020-01-22          0
1            NaN        Albania  41.15330  20.168300  2020-01-22          0
2            NaN        Algeria  28.03390   1.659600  2020-01-22          0
3            NaN        Andorra  42.50630   1.521800  2020-01-22          0
4            NaN         Angola -11.20270  17.873900  2020-01-22          0

   Deaths  Recovered  Active             WHO Region
0       0          0       0  Eastern Mediterranean
1       0          0       0                 Europe
2       0          0       0                 Africa
3       0          0       0                 Europe
4       0          0       0                 Africa
```

```
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
from wordcloud import STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/spam.csv")
print(data.head())
```

```
  label                                               text
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

```
text = " ".join(i for i in data.text)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Visualize a Neural Network using Python

Before visualizing the architecture of a neural network, we must first design a neural network. If you have ever worked on a problem using a neural network, you can skip this part and learn how to visualize the architecture of your neural network as shown in the next section below.

```
import keras
import tensorflow.keras as keras
from matplotlib.pyplot import title
from tensorflow.keras.models import Sequential, Input, Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.advanced_activations import LeakyReLU
import tensorflow as tf
from tensorflow import keras
```

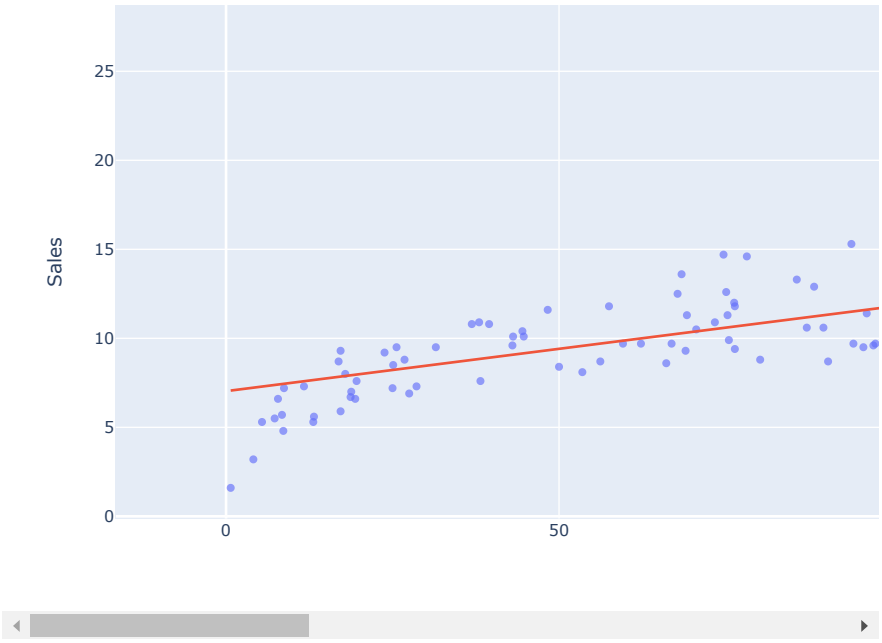
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='linear',input_shape=(28,28,1),padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D((2, 2),padding='same'))
model.add(Conv2D(64, (3, 3), activation='linear',padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
model.add(Conv2D(128, (3, 3), activation='linear',padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
model.add(Flatten())
model.add(Dense(128, activation='linear'))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(500, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
```

Visualize a Machine Learning Algorithm using Python

A machine learning algorithm is used to find relationships between features and labels. Features are the independent variables that we feed into an algorithm to train a machine learning model, and labels are the dependent variables that we aim to predict using the machine learning algorithm.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/Advertising.csv")
print(data.head())
x = data["TV"].values.reshape(-1, 1)
y = data["Sales"]

model = LinearRegression()
model.fit(x, y)
x_range = np.linspace(x.min(), x.max(), 100)
y_range = model.predict(x_range.reshape(-1, 1))

import plotly.express as px
import plotly.graph_objects as go
fig = px.scatter(data, x='TV', y='Sales', opacity=0.65)
fig.add_traces(go.Scatter(x=x_range, y=y_range,
                          name='Linear Regression'))
fig.show()
```

```
   Unnamed: 0     TV  Radio  Newspaper  Sales
0           1  230.1   37.8       69.2   22.1
1           2   44.5   39.3       45.1   10.4
2           3   17.2   45.9       69.3    9.3
3           4  151.5   41.3       58.5   18.5
4           5  180.8   10.8       58.4   12.9
```



Candlestick Chart using Python

A candlestick chart is a data visualization tool used to analyze the price movements of stocks, cryptocurrencies, currencies, and other financial instruments. If you work as a data scientist/analyst in the finance domain, a candlestick chart is one of the most important data visualizations for you.

```
import pandas as pd
data = pd.read_csv("/content/covid_19_clean_complete.csv")
import plotly.graph_objects as go
figure = go.Figure(data=[go.Candlestick(x=data["Date"],
                                        open=data["Long"],
                                        high=data["Lat"],
                                        low=data["Deaths"],
                                        close=data["Recovered"])])
figure.update_layout(title = "Apple Stock Price Analysis",
                     xaxis_rangeslider_visible=False)
figure.show()
```

Animated Scatter Plot using Python

A scatter plot is one of the most useful ways to analyze the relationship between two features.

```
import plotly.express as px
data = px.data.gapminder()
print(data.head())
```

```
       country continent  year  lifeExp       pop   gdpPercap iso_alpha  \
0  Afghanistan      Asia  1952   28.801   8425333  779.445314       AFG
1  Afghanistan      Asia  1957   30.332   9240934  820.853030       AFG
2  Afghanistan      Asia  1962   31.997  10267083  853.100710       AFG
3  Afghanistan      Asia  1967   34.020  11537966  836.197138       AFG
4  Afghanistan      Asia  1972   36.088  13079460  739.981106       AFG

   iso_num
0        4
1        4
2        4
3        4
4        4
```

```
px.scatter(data, x="gdpPercap", y="lifeExp", animation_frame="year", animation_group="country",
           size="pop", color="country", hover_name="country",
           log_x=True, size_max=55, range_x=[100,100000], range_y=[25,90])
```