matplotlib-functions-pract-10

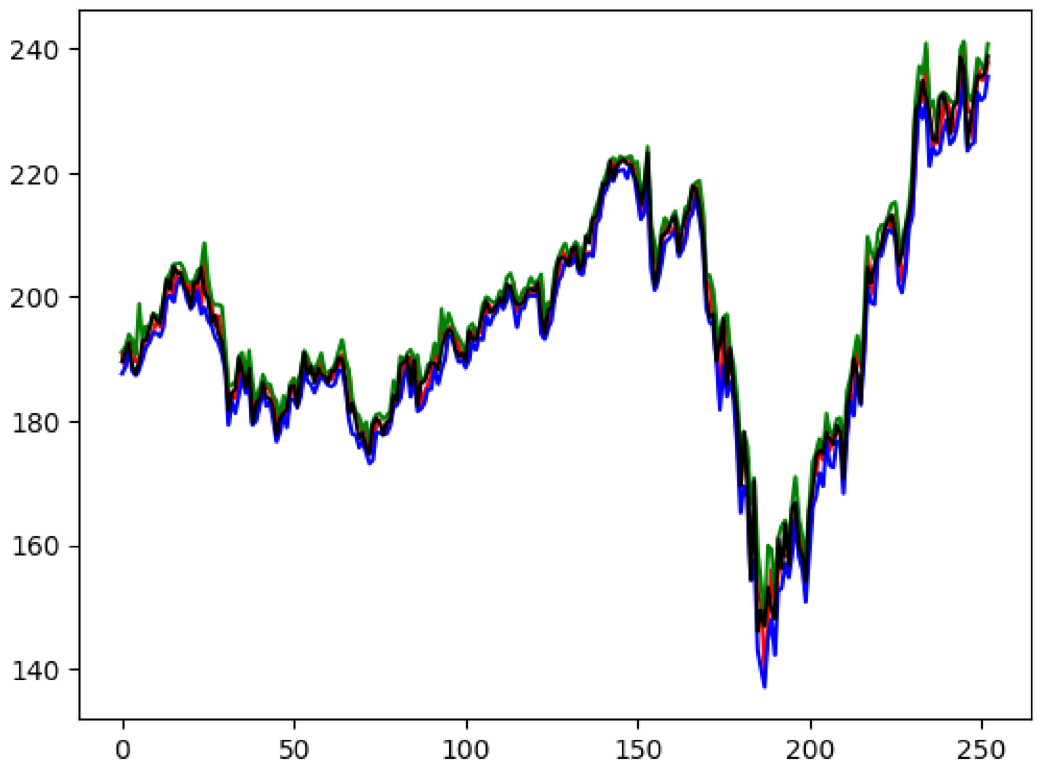
February 7, 2024

Sheth L.U.J. College Of Arts & Sir M.V. College of Science & Commerce Department Of Science Jayesh mali T084 Pract_ 10 Data Science

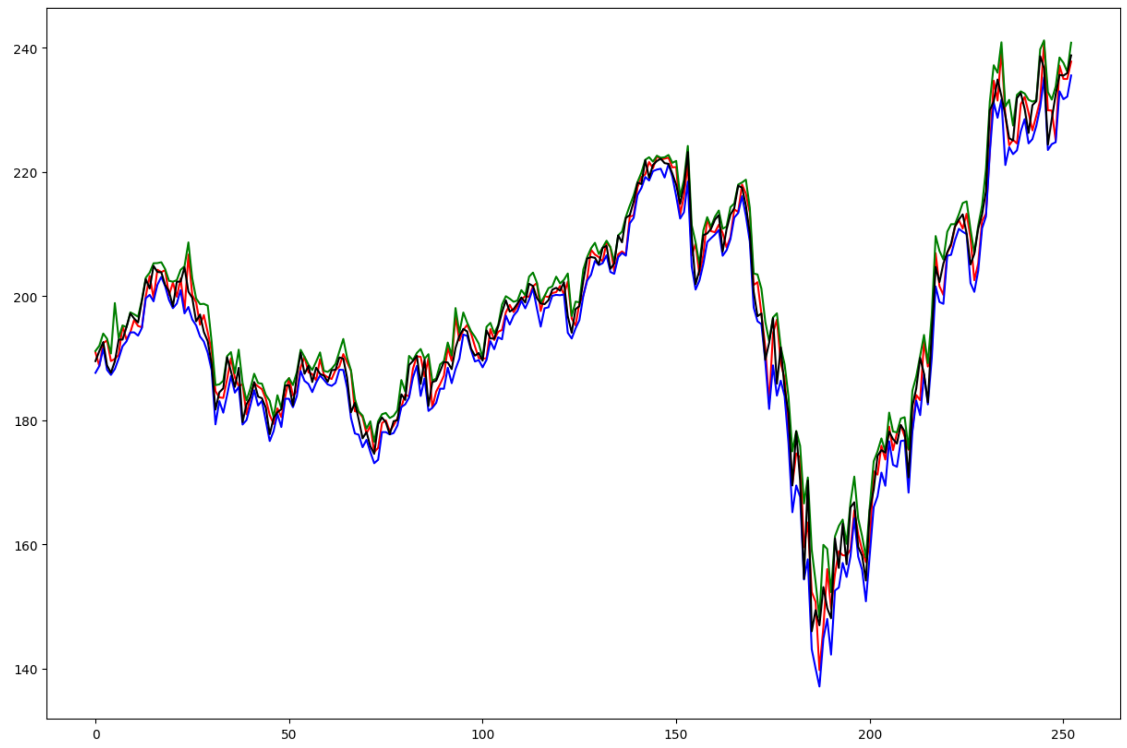
[1]:
```
#Jayesh Mali T084
#Now let's import a dataset and the necessary Python libraries that we need to␣
 ↪create a data visualization:
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("fb_data.csv", encoding="latin-1")
print(data.head())
```

```
        Date        Open        High         Low       Close   Adj Close  \
0  6/20/2019  190.949997  191.160004  187.639999  189.529999  189.529999
1  6/21/2019  188.750000  192.000000  188.750000  191.139999  191.139999
2  6/24/2019  192.419998  193.979996  191.570007  192.600006  192.600006
3  6/25/2019  192.880005  193.139999  188.130005  188.839996  188.839996
4  6/26/2019  189.539993  190.759995  187.309998  187.660004  187.660004

     Volume
0  14635700
1  22751200
2  15509000
3  16750300
4  12808600
```

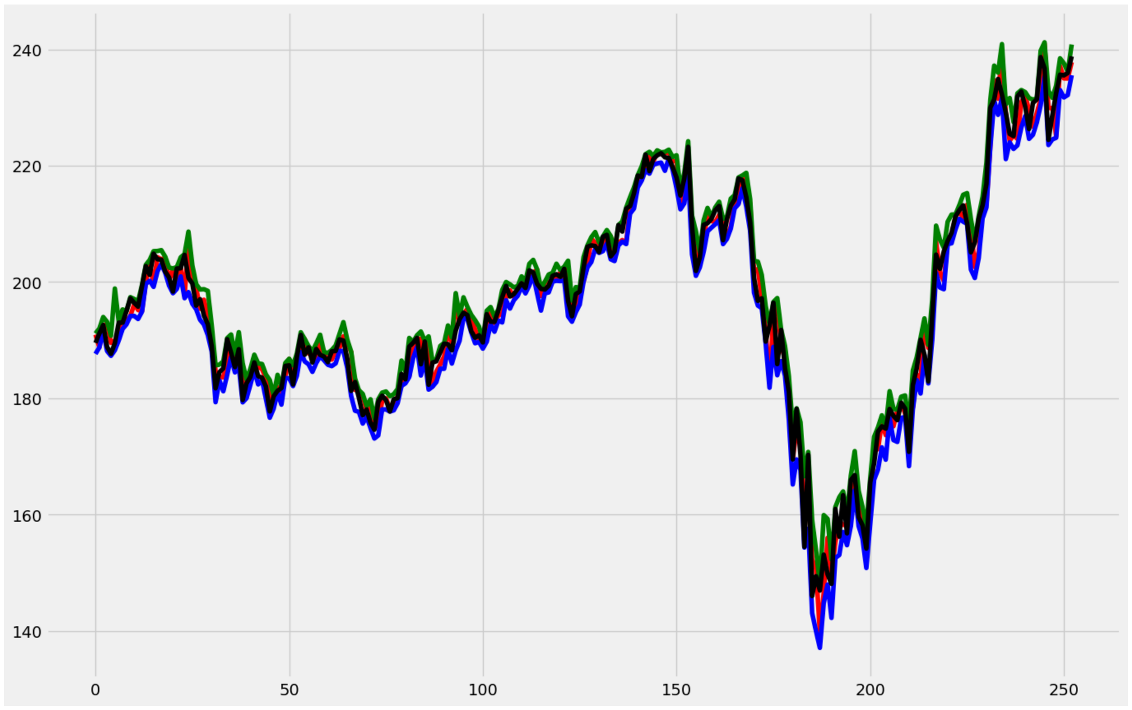
[3]:
```
#Jayesh Mali T084
#So let's create a line plot of my facebook reach from various sources as␣
 ↪mentioned in the dataset:
# Creating a Line Plot
plt.plot(data["Open"], "-r", label="Open")
plt.plot(data["High"], "-g", label="High")
plt.plot(data["Low"], "-b", label="low")
plt.plot(data["Close"], "-k", label="Close")
plt.show() # for visualizing your graph
```



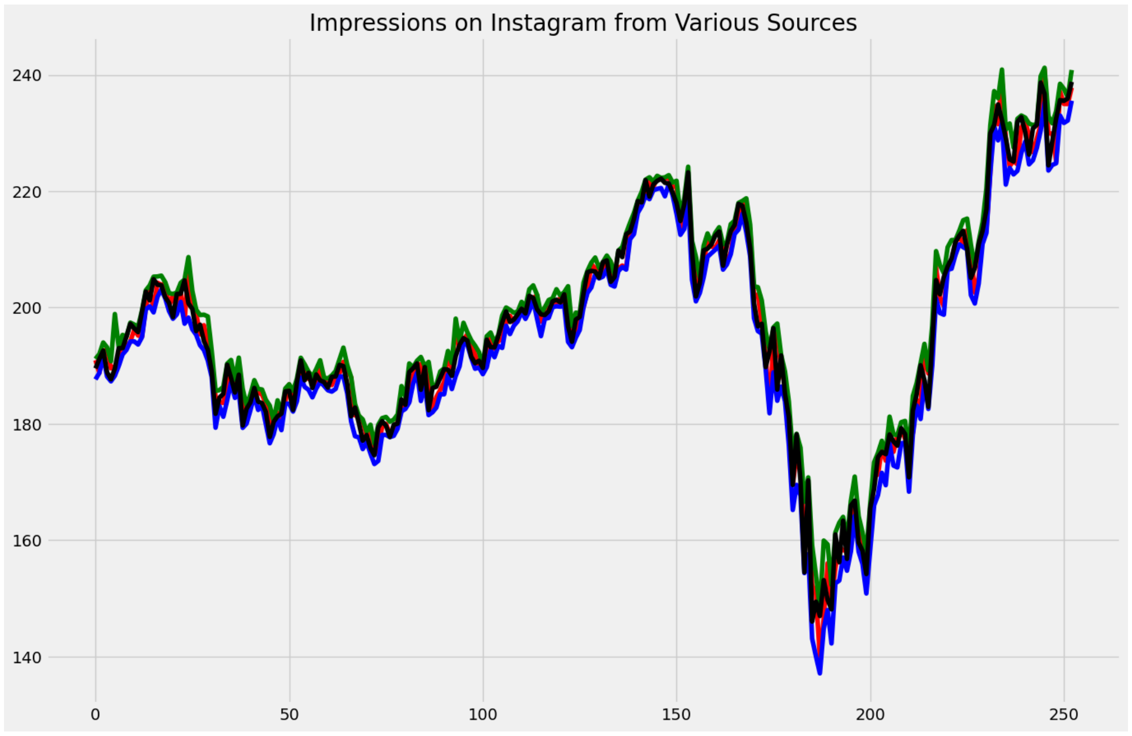
```
[4]: #Jayesh Mali T084
     #Customizing Figure Size:
     #The plt.figure(figsize=(float, float)) helps us to customize the size of our␣
      ↪graphs. Here’s how to use this function
     # Customizing Figure Size
     plt.figure(figsize=(15, 10)) # Customizing Figure Size
     plt.plot(data["Open"], "-r", label="Open")
     plt.plot(data["High"], "-g", label="High")
     plt.plot(data["Low"], "-b", label="low")
     plt.plot(data["Close"], "-k", label="Close")
     plt.show()
```



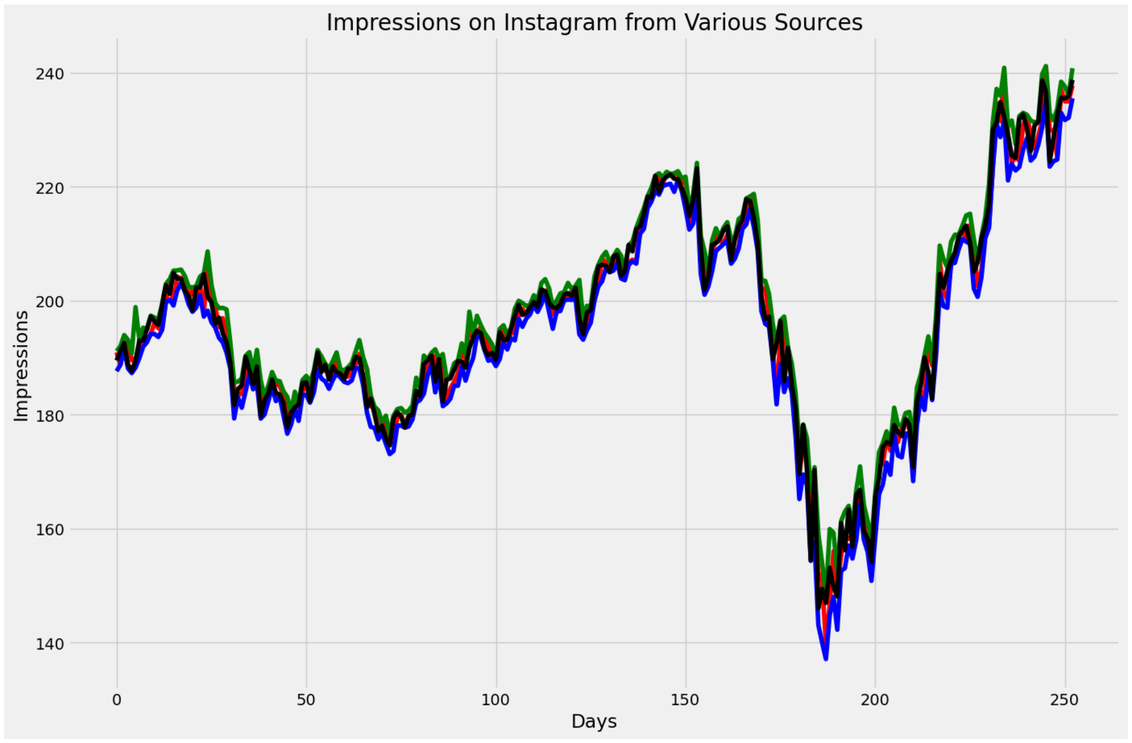
[5]:
```
#Jayesh Mali T084
#Customizing Theme:
#The plt.style.use("style name") helps us customize the theme of our graphs.␣
 ↪For now, I will be using the "fivethirtyeight" theme style of matplotlib.␣
 ↪Here’s how to use this function:
# Customizing Themes
plt.style.use('fivethirtyeight') # for customizing theme
plt.figure(figsize=(15, 10))
plt.plot(data["Open"], "-r", label="Open")
plt.plot(data["High"], "-g", label="High")
plt.plot(data["Low"], "-b", label="low")
plt.plot(data["Close"], "-k", label="Close")
plt.show()
```



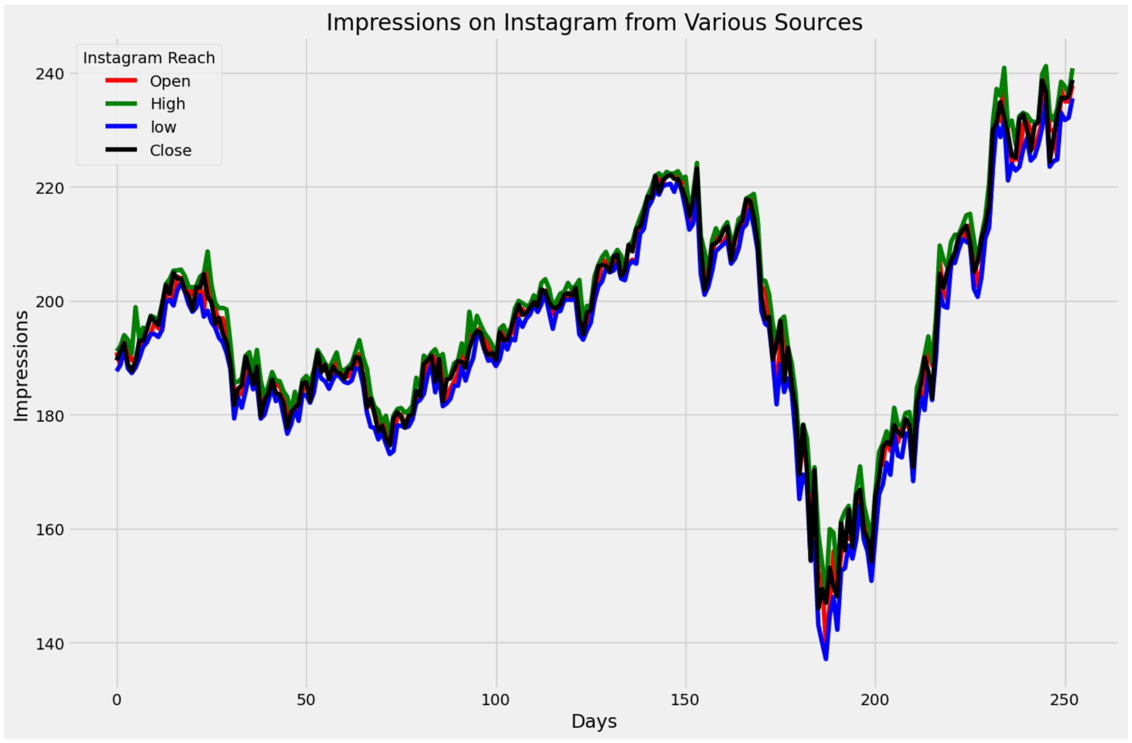
```
[6]: #Jayesh Mali T084
     #Adding Title:
     #Adding a title to your graphs is always a good habit. It helps in describing␣
      ↪the purpose of the data visualization. Here is how to add the title to our␣
      ↪graphs:
     # Adding a Title to Your Graph
     plt.style.use('fivethirtyeight')
     plt.figure(figsize=(15, 10))
     plt.plot(data["Open"], "-r", label="Open")
     plt.plot(data["High"], "-g", label="High")
     plt.plot(data["Low"], "-b", label="low")
     plt.plot(data["Close"], "-k", label="Close")
     plt.title("Impressions on facebook from Various Sources") # for adding a title
     plt.show()
```



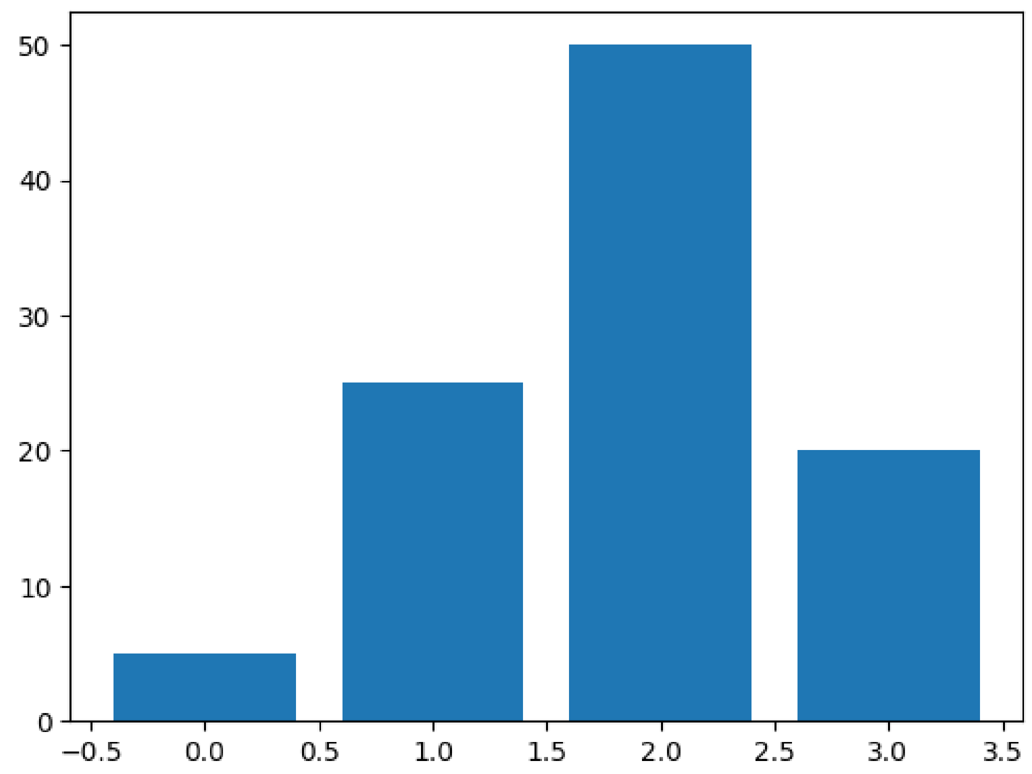
[7]:
```
#Jayesh Mali T084
#Adding Labels on the Axes
#The xlabel() and ylabel() functions will help us add labels to the x and y␣
↪axes of our charts. Here’s how to use these functions:
# Adding Labels on xaxis and yaxis
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(data["Open"], "-r", label="Open")
plt.plot(data["High"], "-g", label="High")
plt.plot(data["Low"], "-b", label="low")
plt.plot(data["Close"], "-k", label="Close")
plt.title("Impressions on facebook from Various Sources")
plt.xlabel("Days") # adding label on xaxis
plt.ylabel("Impressions") # adding label on yaxis
plt.show()
```



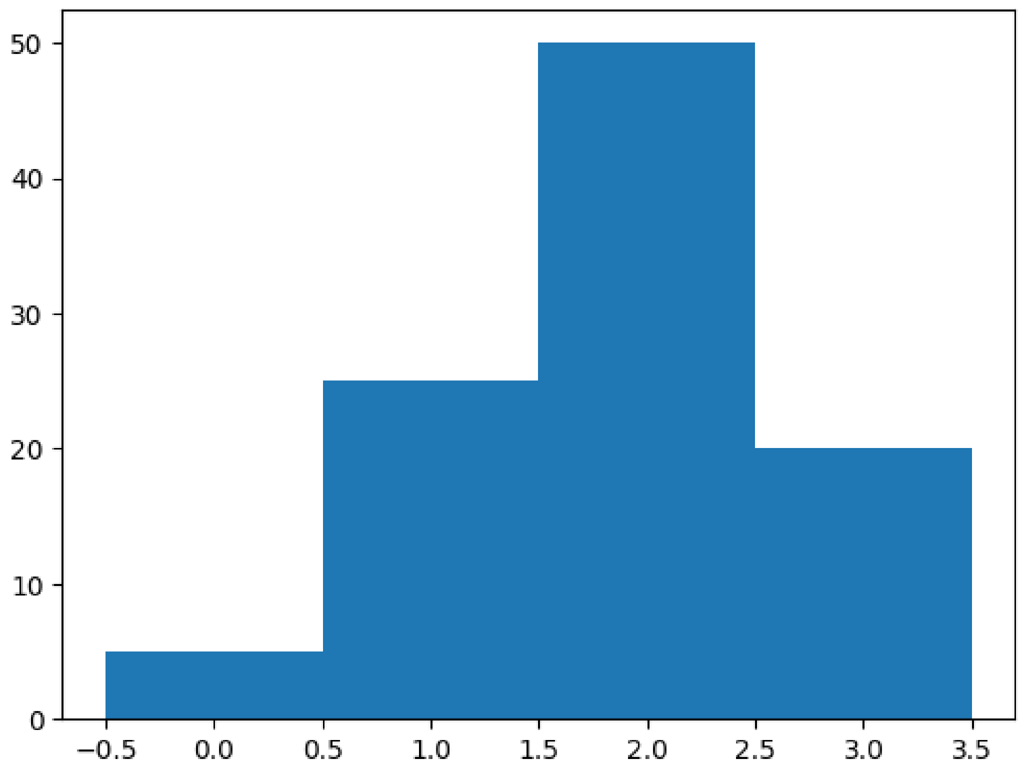
[8]:
```
#Jayesh Mali T084
#Adding Legend
#A graph's legend displays the labels and colours for each feature displayed in␣
↪the graph. Here's how to add a legend to our graphs:
# Adding Legend
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15, 10))
plt.plot(data["Open"], "-r", label="Open")
plt.plot(data["High"], "-g", label="High")
plt.plot(data["Low"], "-b", label="low")
plt.plot(data["Close"], "-k", label="Close")
plt.title("Impressions on facebook from Various Sources")
plt.xlabel("Days")
plt.ylabel("Impressions")
plt.legend(title="facebook Reach") # for adding legend with a title
plt.show()
```



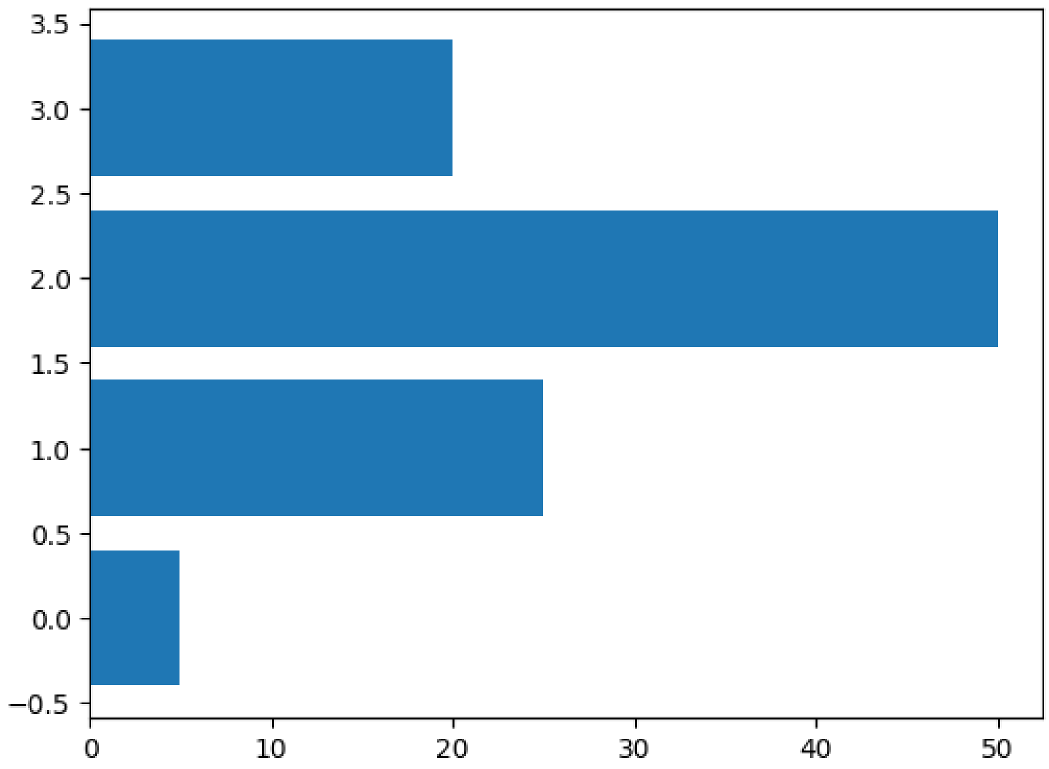
```
[1]: #Jayesh Mali T084
     #Bar Plots
     #In this section, I will take you through how to visualize Bar plots with␣
      ↪Python by using the matplotlib library. Let’s start by plotting a basic bar␣
      ↪plot:
     import pandas as pd
     import matplotlib.pyplot as plt
     data = [5., 25., 50., 20.]
     plt.bar(range(len(data)), data)
     plt.show()
```



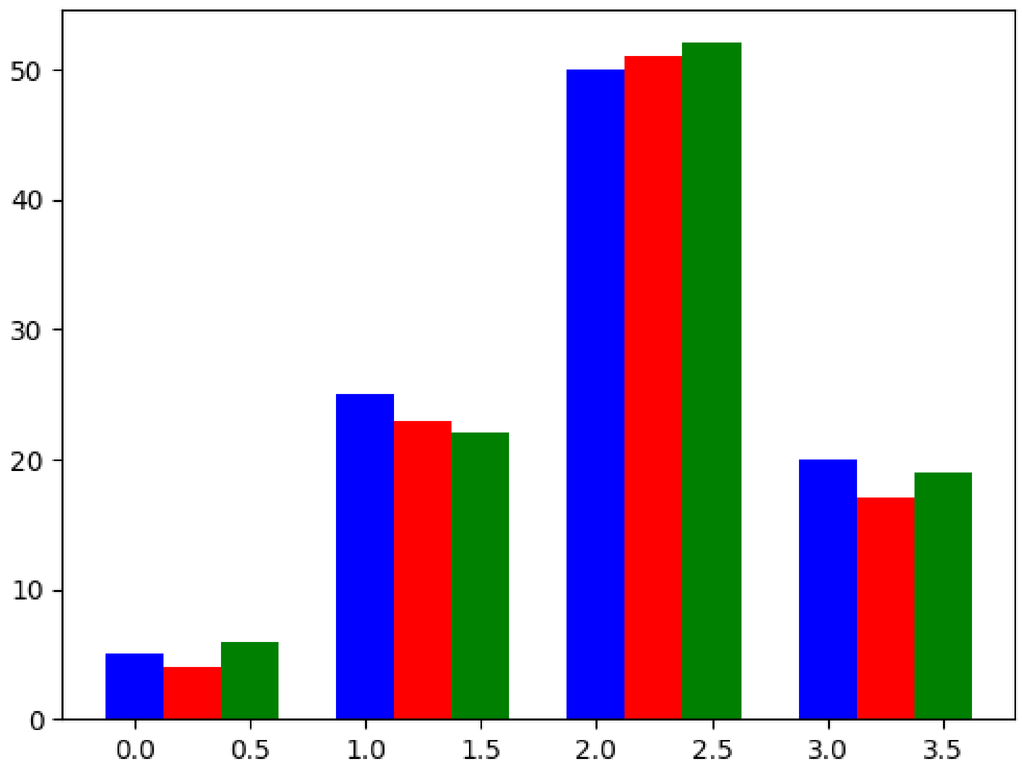
[2]:
```
#Jayesh Mali T084
data = [5., 25., 50., 20.]
plt.bar(range(len(data)), data, width=1.)
plt.show()
```



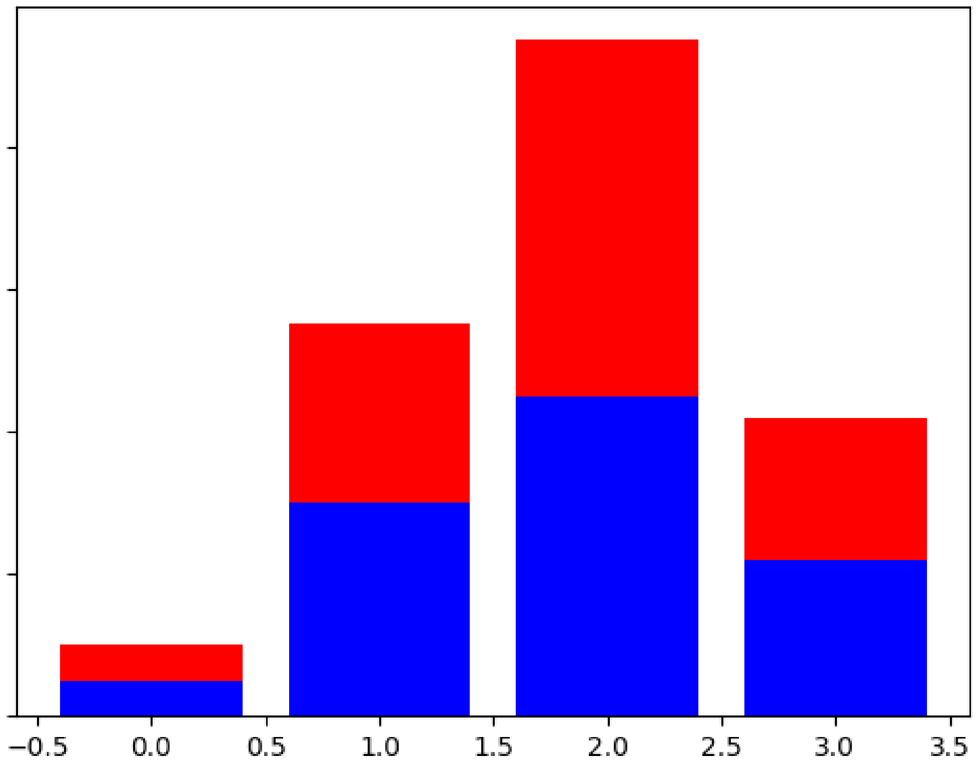
[3]:
```
#Jayesh Mali T084
#Horizontal Bar
#Now let’s see how to visualize the horizontal bar charts with Python
import pandas as pd
import matplotlib.pyplot as plt
data = [5., 25., 50., 20.]
plt.barh(range(len(data)), data)
plt.show()
```



[4]:
```
#Jayesh Mali T084
#import pandas as pd
#We can draw several histograms by playing with the thickness and the positions␣
 ↪of the bars as follows:
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()
```



```
[5]: #Jayesh Mali T084
     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()
```



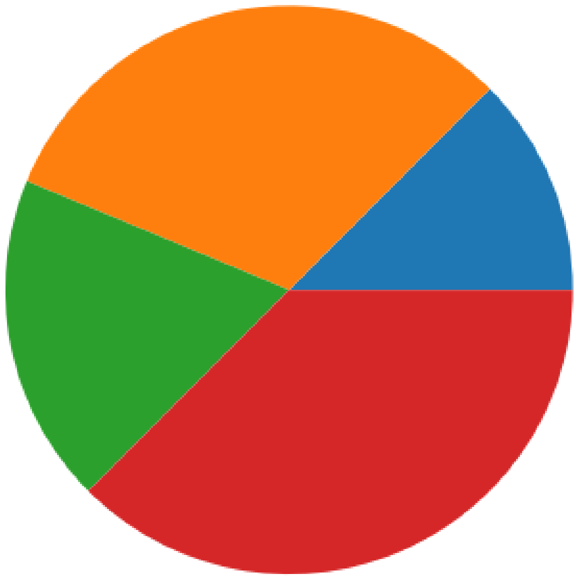
[1]:
```
#Jayesh mali T084
#There are so many data visualization libraries in Python that we can use for␣
 ↪visualizing treemaps, but the easiest way is to use the plotly library in␣
 ↪Python. So below is how we can use plotly to visualize a treemap using␣
 ↪Python:
#Visualizing a Treemap
import plotly.graph_objects 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()
```

[2]:
```
#Jayesh Mali T084
#I will start by importing the necessary Python libraries and a dataset that we␣
 ↪can use to visualize box plots using Python:
import pandas as pd
data = pd.read_csv("real_2013_air.csv")
print(data.head())
```

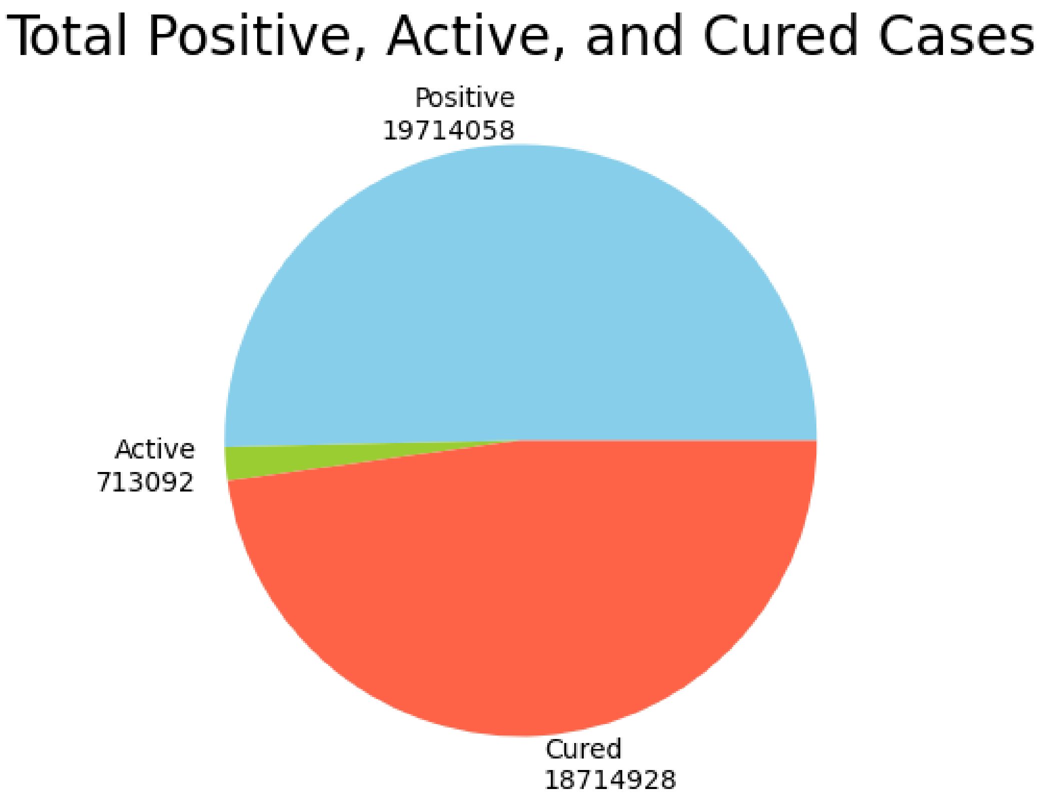
```
      T    TM   Tm     SLP   H   VV    V    VM      PM 2.5
0   7.4   9.8  4.8  1017.6  93  0.5  4.3   9.4  219.720833
1   7.8  12.7  4.4  1018.5  87  0.6  4.4  11.1  182.187500
2   6.7  13.4  2.4  1019.4  82  0.6  4.8  11.1  154.037500
3   8.6  15.5  3.3  1018.7  72  0.8  8.1  20.6  223.208333
4  12.4  20.9  4.4  1017.3  61  1.3  8.7  22.2  200.645833
```

[3]:
```
#Jayesh Mali T084
import plotly.express as px

# Assuming your DataFrame is named 'data' and the correct column name for the␣
  ↪y-axis is 'Total Views'
fig = px.box(data, y ="TM")
fig.show()
```

[4]:
```
#Jayesh Mali T084
#he pyplot.pie() function of the matplotlib library can be used to visualize a␣
  ↪pie chart:
import matplotlib.pyplot as plt
data = [20, 50, 30, 60]
plt.pie(data)
plt.show()
```

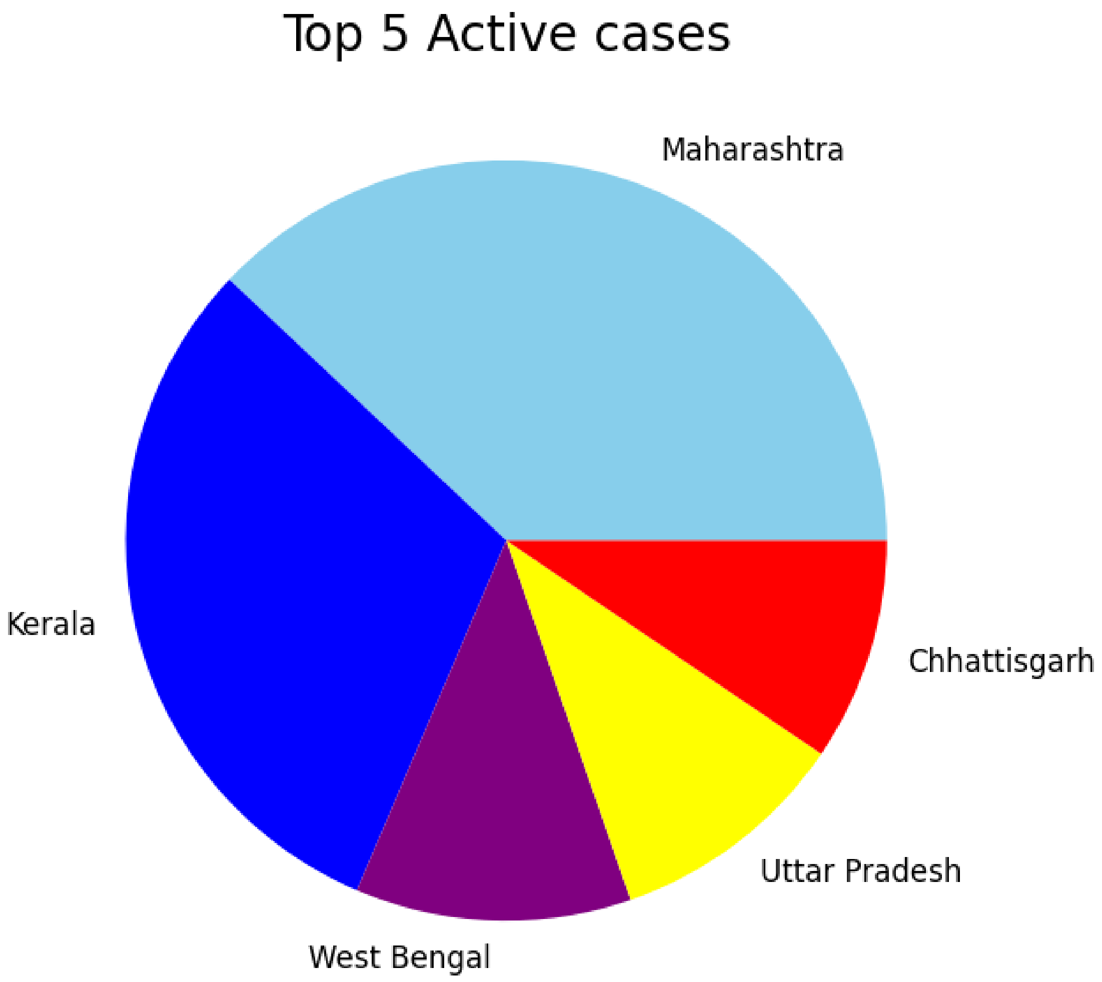


[5]:
```
#Jayesh Mali T084
#Here I will use the Covid-19 dataset to visualize the proportion of active,␣
↪positive, and cured cases in India according to the dataset
import pandas as pd
df = pd.read_json("datanew.json")
group_size = [sum(df.positive), sum(df.active), sum(df.cured)]
group_labels = ["Positive\n"+str(sum(df.positive)),
                "Active\n"+str(sum(df.active)),
                "Cured\n"+str(sum(df.cured))]
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()
```



[6]:
```
#Jayesh Mali T084
#Now let's visualize the distribution of top 5 cities with active cases of␣
↪covid-19 in India:
df.drop(df.tail(1).index, inplace = True)
df1 = df.sort_values(by='active', ascending=False)
```

```python
df3 = df1[:5]
states = df3.state_name
active = df3.active
colours = ["skyblue", "blue", "purple", "yellow", "red"]
plt.figure(figsize=(7,7))
plt.pie(active, labels=states, colors=colours)
plt.rc('font', size=12)
plt.title("Top 5 Active cases", fontsize=20)
plt.show()
```

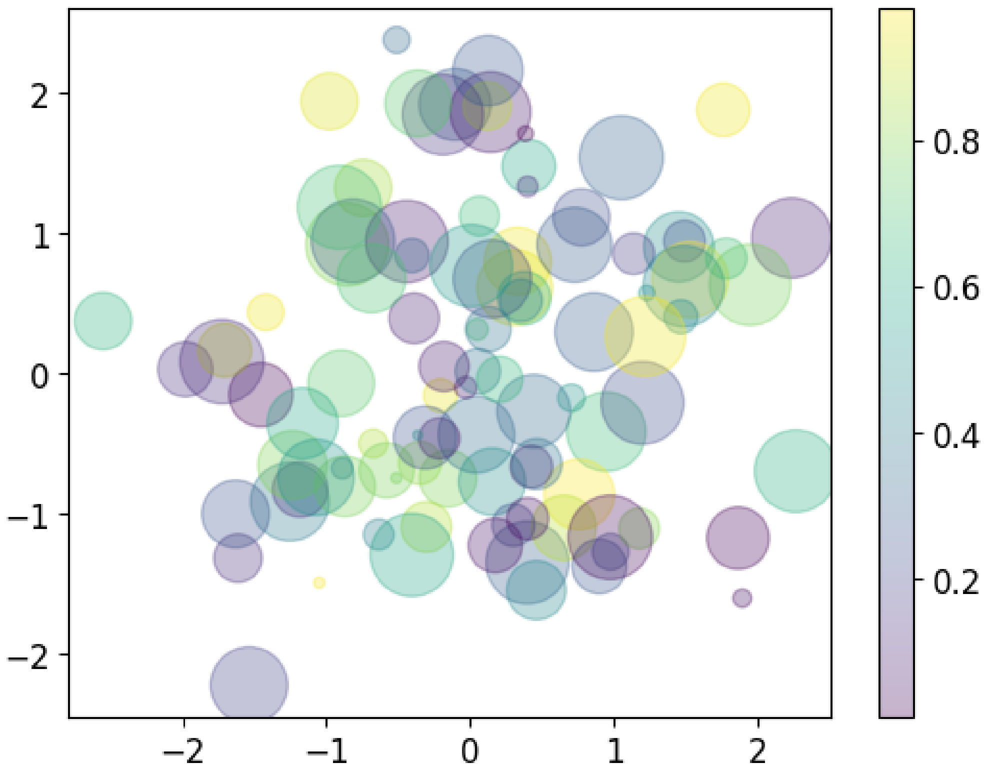


[7]:
```python
#Jayesh Mali T084
#I will be using Plotly as it is easy to visualize interactive visualizations␣
 ↪using plotly. So let’s start this task by collecting the latest stock price␣
 ↪data of Apple:
import pandas as pd
import yfinance as yf
import datetime
from datetime import date, timedelta
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())
```

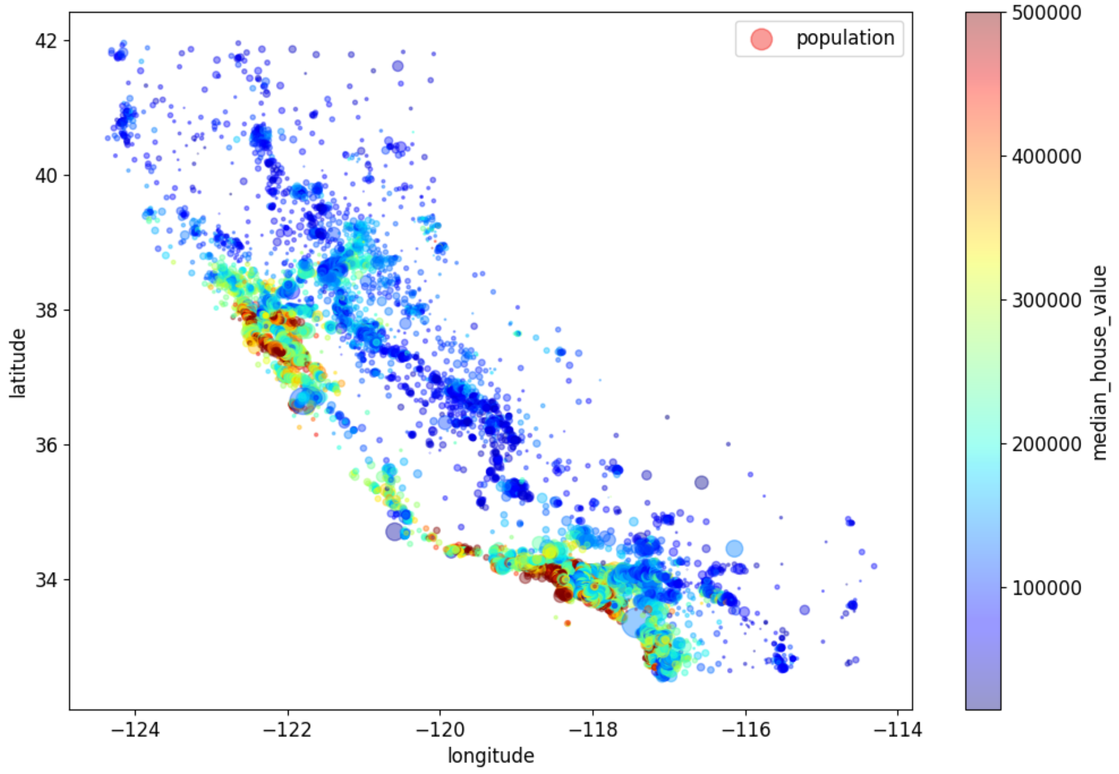
```
                  Open        High         Low       Close   Adj Close  \
Date
2023-02-13  150.949997  154.259995  150.919998  153.850006  153.228439
2023-02-14  152.119995  153.770004  150.860001  153.199997  152.581055
2023-02-15  153.110001  155.500000  152.880005  155.330002  154.702438
2023-02-16  153.509995  156.330002  153.350006  153.710007  153.089005
2023-02-17  152.350006  153.000000  150.850006  152.550003  151.933685

              Volume
Date
2023-02-13  62199000
2023-02-14  61707600
2023-02-15  65573800
2023-02-16  68167900
2023-02-17  59144100
```

[8]:
```
#Jayesh Mali T084
#The above data is collected by using the yfinance API. You can know more about␣
 ↪it from here. Now below is how you can visualize a time series graph using␣
 ↪Python:
import plotly.express as px
figure = px.line(data, x = data.index, y = "Close")
figure.show()
```

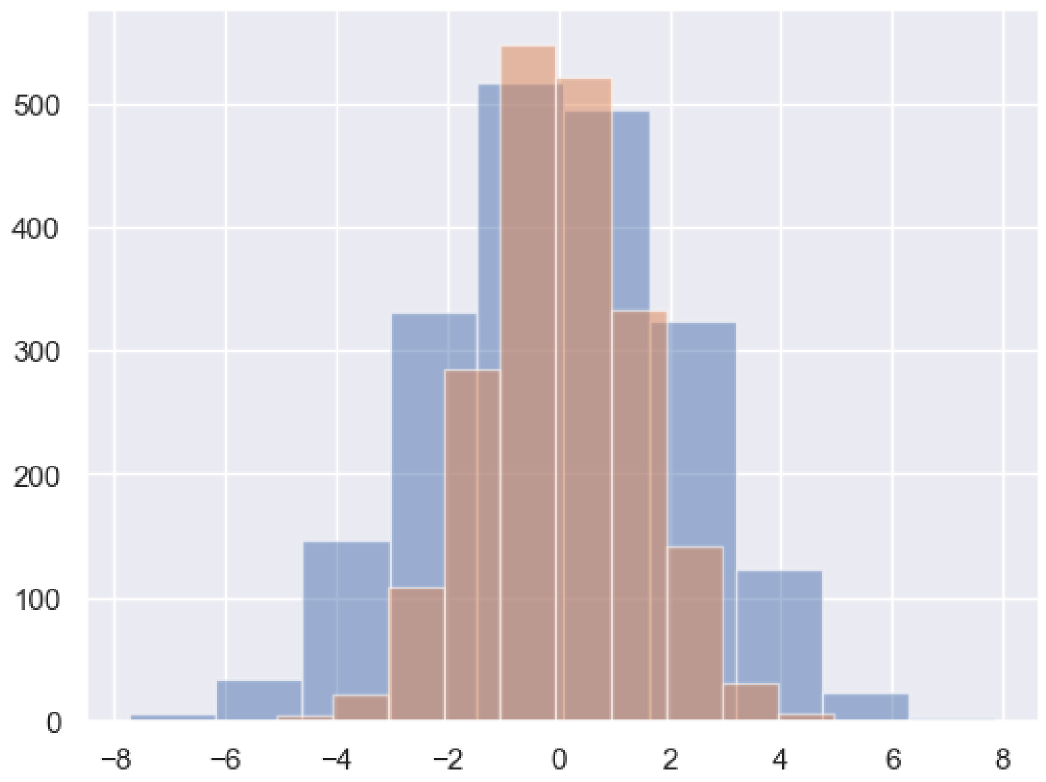
[9]:
```
#Jayesh Mali T084
#Scatter Plot
#I will first use numerical data generated by using Numpy to plot a scatter␣
 ↪plot and then I will use a real-time dataset to plot a scatter plot with␣
 ↪Python
import numpy as np
import matplotlib.pyplot as plt
rng = np.random.RandomState(0)
x = rng.randn(100)
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()
```



[10]:
```
#Jayesh Mali T084
#The idea is to visualize the densely populated areas with bigger circles and␣
↪the areas with high prices with darker circles and vice versa
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()
```



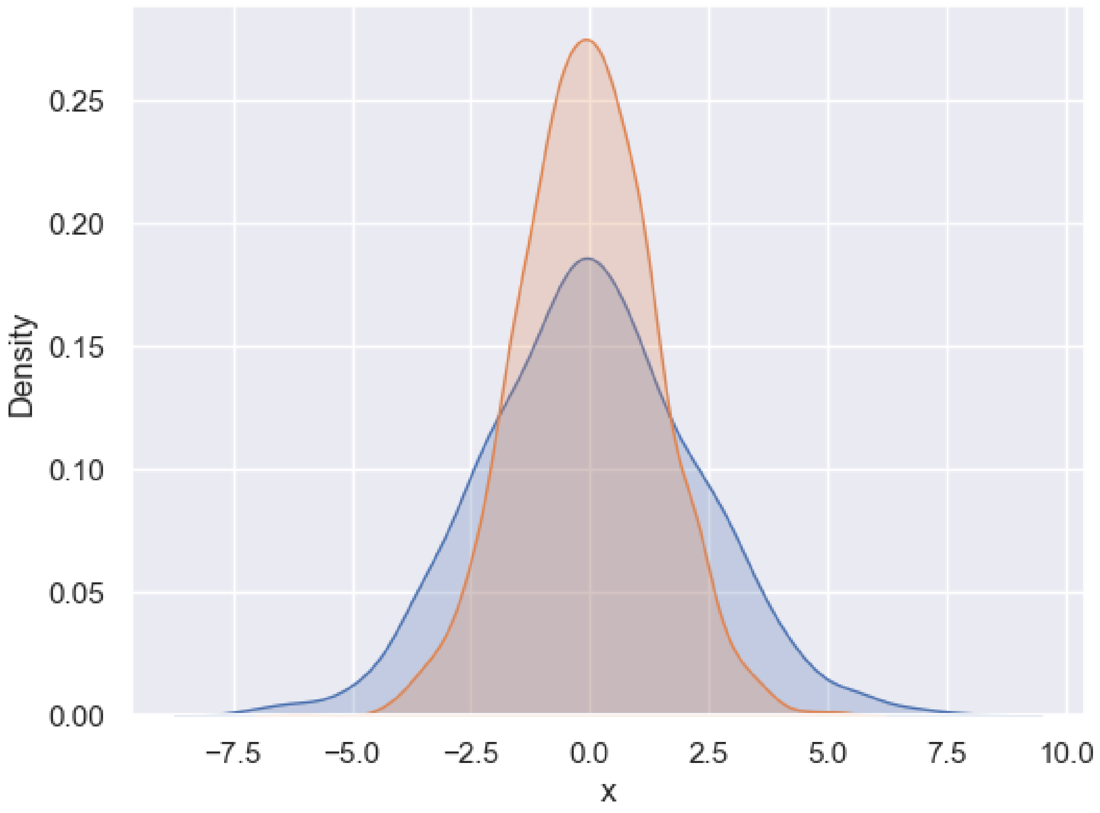
[11]:
```
#Jayesh Mali T084
#Histogram and Density
#when creating a histogram to analyze the distribution of the dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

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()
```



[12]:
```
#Jayesh Mali T084
#Density Plots:
# Density plots are created in such a way that the area under the curve is␣
↪always equal to 1. Here's how you can visualize a density plot using Python:
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()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_7364\1641966425.py:12:
FutureWarning:



`shade` is now deprecated in favor of `fill`; setting `fill=True`.
```

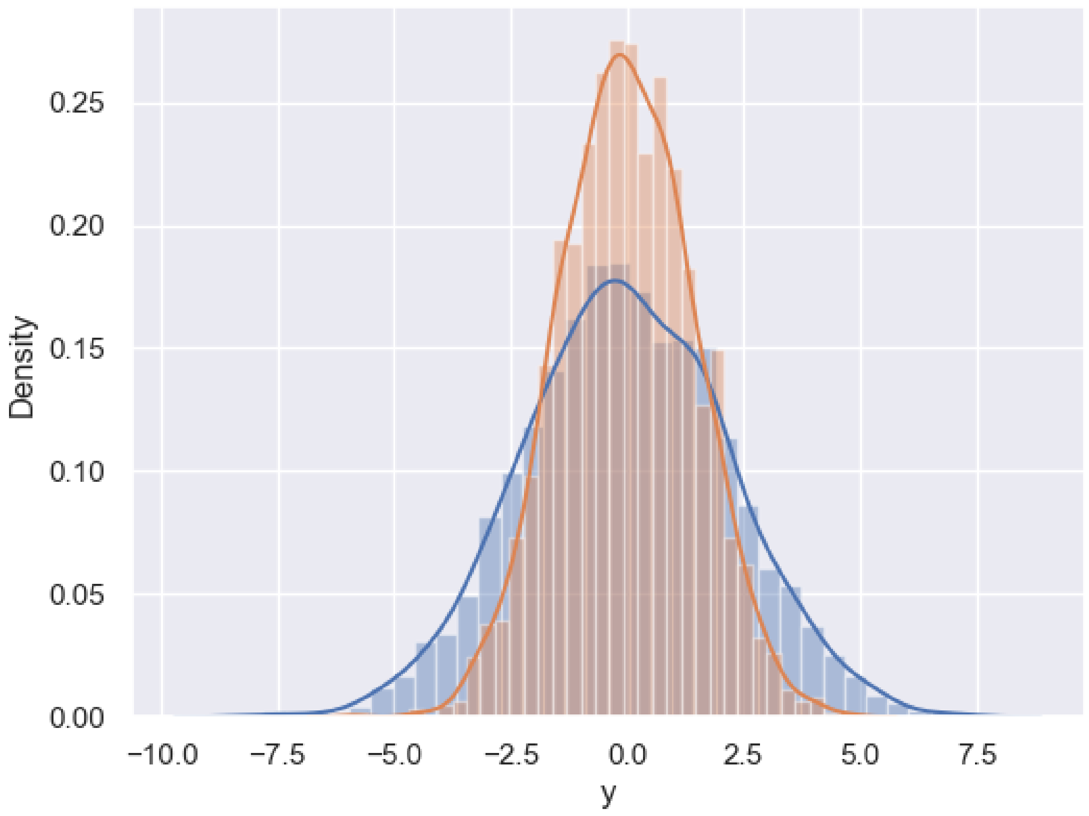
```
This will become an error in seaborn v0.14.0; please update your code.


C:\Users\Admin\AppData\Local\Temp\ipykernel_7364\1641966425.py:13:
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.
```



[13]:
```
#Jayesh Mali T084
#We can also visualize both histograms and density plots at once. Below is how␣
↪you can visualize both of them using Python:
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()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_7364\1233768488.py:11: 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


C:\Users\Admin\AppData\Local\Temp\ipykernel_7364\1233768488.py:12: 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
```


0.25
0.20
0.15
0.10
0.05
0.00
Density
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
y