Chapter 12
Data Visualizations

Visualizations present the data in the form of graphs and charts. It reveals the patterns, trends, and correlations that might not be observed otherwise. Data visualization is an important step in data analytics to visualize complex data relationships that help in data-driven decisions.

Python provides multiple packages for data visualizations. Popular packages are

- matplotlib
- seaborn
- ggplot
- plotly

Matplotlib and seaborn are widely used visualization libraries, as the graphs can be produced easily and quickly.

12.1 Matplotlib

Matplotlib is the visualization library for two-dimensional plots and charts. John Hunter introduced it in 2002. It provides lots of features to produce elegant graphs. The Pyplot module of matplotlib provides an interface for graphs. It offers low-level libraries with an interface similar to MATLAB. It can be used in Python, IPython, and Jupyter notebooks.

First, install the package using

```
pip install matplotlib
```

Then, import the package before using the library functions.

```
import matplotlib.pyplot as plt
```

In the above plt is the shortcut name for maptplotlib.pyplot.

A. L. Muddana and S. Vinayakam, *Python for Data Science*,
https://doi.org/10.1007/978-3-031-52473-8_12

12.2 Seaborn

Seaborn is a Python data visualization library developed on top of matplotlib. It provides high-level interfaces for building statistical graphs with default matplotlib parameters like styles and color palettes to make the plots attractive and informative. It is a dataset-oriented library and integrated with data frames and numPy arrays.

Both matplotlib and seaborn offer a number of library functions for basic graphs like scatter, line, bar, box plot, *etc.*. The type of plot to choose depends on the type of data, target audience, context, and the question we are trying to answer.

First, import the seaborn module to start using its library functions

```
import seaborn as sns.
```

12.3 General Functions in Plotting

(i) Graph **title**

plt.***title***(*label, loc* = 'center',*fontdict*=None)

label: Actual title text
loc: Alignment of the label in the graph. The value can be center (default), left, right.
fontdict: Dictionary that controls the appearance of the label font like color, size, style, *etc.*

(ii) Label for the **x-axis**

plt.***xlabel***(label_text)

(iii) Label for the **y-axis**

plt.***ylabel***(label_text)

(iv) Display **grid lines** in the graph

plt.***grid***()

(v) Display the plots

plt.***show***()

(vi) Display tick locations and labels for x-axis and y-axis

plt.**xticks**(*ticks*=None, *labels*=None)
plt.**yticks**(*ticks*=None, *labels*=None)

ticks: List of tick locations
labels: List of labels at tick locations

(vii) Set the **limits** for **x-axis** and **y-axis** values

plt.***xlim***(left_value , right_value)
plt.***ylim***(bottom_value , top_value)

(viii) Describe the elements of the graph.

plt.***legend***(*loc*=None)

loc: where to place the legend like upper left, upper right, center, lower left, lower right *etc.*. Default is 'best'.

12.4 Basic Graphs and Plots

Following are the widely used basic graphs and plots provided by matplotlib and seaborn.

(i) **Scatter plot** exhibits the relationship between a dependent and an independent variable. It shows the correlation between the two variables, which may be a positive relation, negative relation, or no correlation at all. It is also useful in identifying the outliers in the dataset. This plot is used to see the correlation, before building the regression model. The scatter plot displays one dot for each data point.

(a) **Scatter plot using matplotlib**

*plt.**scatter**(x,y, size =None, color=None, marker=None, alpha=None)*

x, y:	Array of values for x & y-axis Both the arrays should be of the same size
color or *c:*	Color of the data points. The argument can be specified as a single value, in which case each data point takes the same color or array of colors for each data point The color array should be of the same size as the x and y array sizes
marker:	Specify the shape of each data point Different markers are dot (default), circle (o), star (*), plus (+), *etc.*
size:	The size of each data point The argument value can be a single or array of sizes
alpha:	Transparency of each data point The value ranges from 0 (transparent) to 1 (opaque).

Example

```
import pandas as pd
# load the dataset into dataframe- data
data = pd.read_csv('/content/sample_data/Housing.csv')
x = data['area']
y = data['price']
```

```
plt.scatter(x, y, c = 'green', marker='o', s=30 )
plt.title( 'Area vs Price of house', fontdict={ 'color': 'blue',
  'size':20, 'style': 'italic' } )
plt.xlabel('Area of the House')      # name to x-axis
plt.ylabel('Price of the house')     # name to y-axis
plt.show()     # to display the graph
```

Output

(b) **Regression plot using seaborn**

The plot is a combination of scatter plot and fitting the regression line.

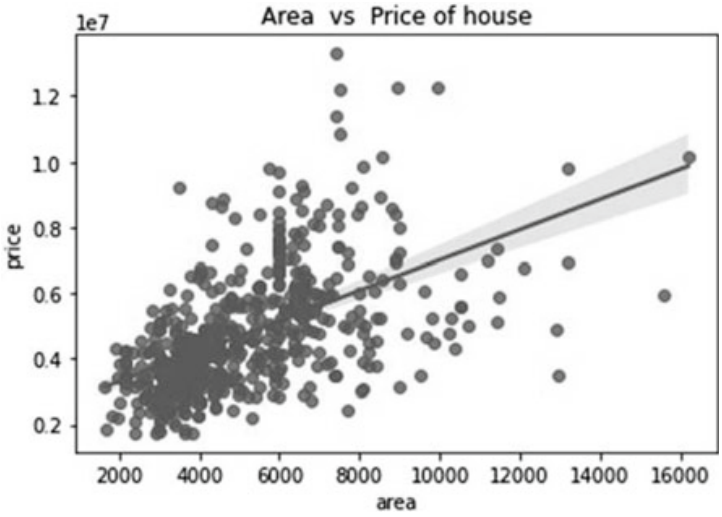
seaborn.***regplot***(x=None, y=None, data, scatter=True, fit_reg=True)

data:	Dataframe
x, y:	Columns of the dataframe
scatter:	Boolean value indicating whether to draw a scatter plot or not. Default is True
fit_reg:	Boolean value indicating whether to fit a regression line or not. Default is True

Example

```
import pandas as pd
data = pd.read_csv('/content/sample_data/Housing.csv')
sns.regplot(x= 'area', y= 'price', data=data)
plt.title('Area vs Price of house')
plt.show()
```

Output



Note: In the above seaborn plot, even though x-axis and y-axis labels are not specified, they are taken as column names, i.e., **area** and **price**.

(ii) **Line graph** draws line by connecting the data points. The graph represents the trends, patterns, and fluctuations in the values of continuous variables over time. It helps in making projections beyond the given data.

(a) **Using matplotlib**

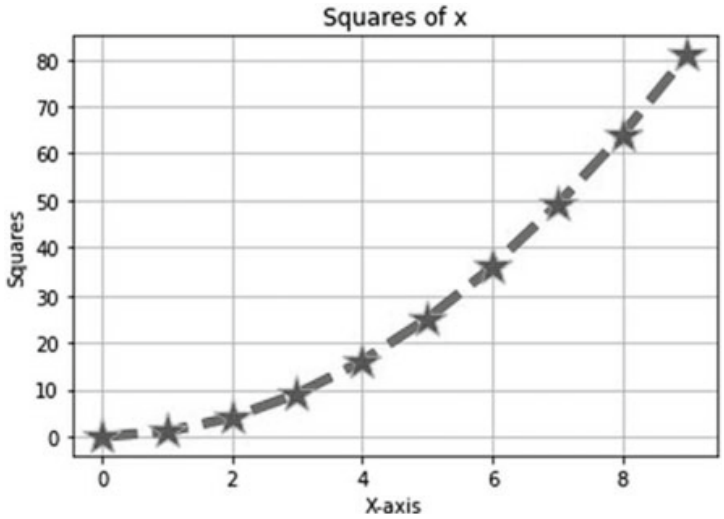
*plt.***plot***(x,y,color=None,linewidth=None,marker=None, linestyle=None,markersize=None,markerfacecolor=None, markeredgecolor=None)*

x, y :	Array of values for x & y-axis
color or c:	Line color
linestyle or ls:	solid(-), dotted(.), dashed(–), dashed-dotted(-.)
linewidth or lw:	Float value representing the width of the line
marker or m:	Data point marker like a circle(o), cross(x), star(*), plus(+)
markersize or ms:	Size of the data point
markerfacecolor:	Color for the marker
markeredgecolor:	Edge color for the marker.

Example

```
x = list(range(10))
y1 = [ i *i for i in x]
plt.plot(x,y1,c='red',linewidth=5,linestyle='–',marker='*',
markersize=20,markerfacecolor='green',markeredgecolor='orange')
plt.title('Squares of x')
plt.xlabel('X-axis')
plt.ylabel('Squares')
plt.grid()      # shows grid lines for x, y values
plt.show()
```

Output

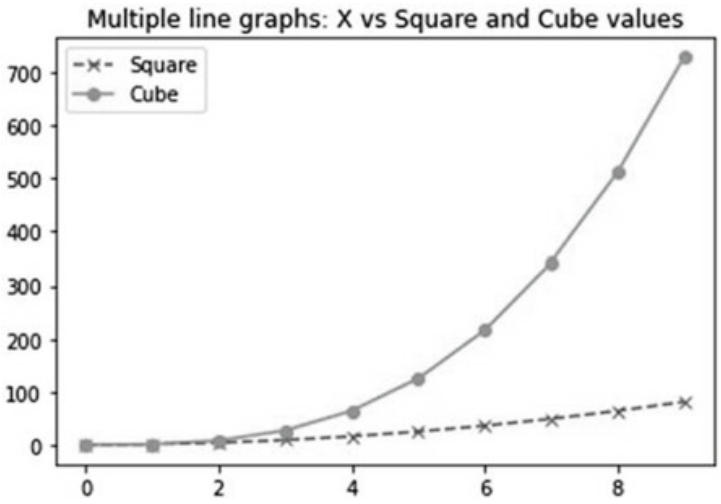


Multiple line graphs can be displayed in the same plot

Example

```
x = list(range(10))
y1 = [ i *i for i in x]
y2 = [ i**3 for i in x]
# Line graph of Square values
plt.plot(x, y1, label='Square', ls='dashed', marker='x')
# Second Line graph of Cube values
plt.plot(x, y2, label='Cube', marker ='o')
plt.title('Multiple line graphs: X vs Square and
  Cube values')
# Legend labels for each line graph
plt.legend()
plt.show()
```

Output



(b) **Using seaborn**

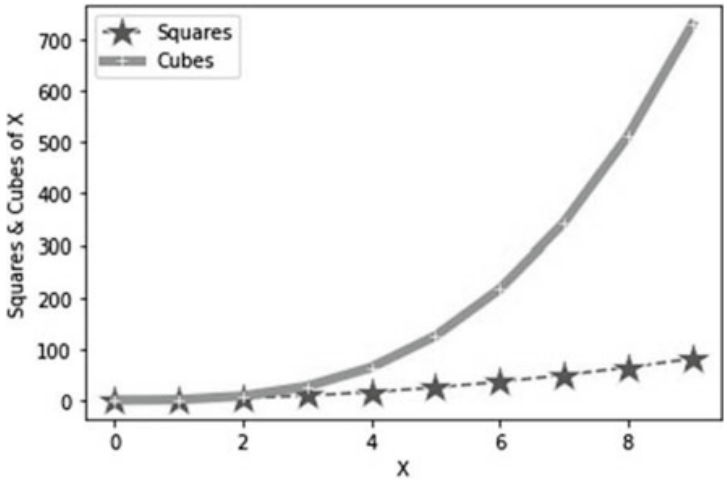
*seaborn.***lineplot**(*x, y, data*)

data: Dataframe
x, y: Columns of the dataframe

Example

```
import pandas as pd
x = list(range(10))
y1 = [ i *i for i in x]
y2 = [ i**3 for i in x]
df=pd.DataFrame('X' : x, 'Y1': y1, 'Y2' : y2 )
# Draw line graph
sns.lineplot(x='X',y='Y1',data=df,linestyle='–',
  markersize=2,marker='*')
sns.lineplot(x='X',y='Y2',data=df, linewidth=5,marker='+')
plt.legend(['Squares','Cubes'])
plt.ylabel('Squares & Cubes of X')
plt.show()
```

Output



(iii) **Histogram** shows how the values of a continuous variable are distributed. Data is divided into discrete consecutive intervals. Each interval is called a bin. The bin is represented by a bar and the graph consists of a sequence of bars. The number of values in each interval determines the height of the bar. The width of the bars can be specified as an argument to the function.

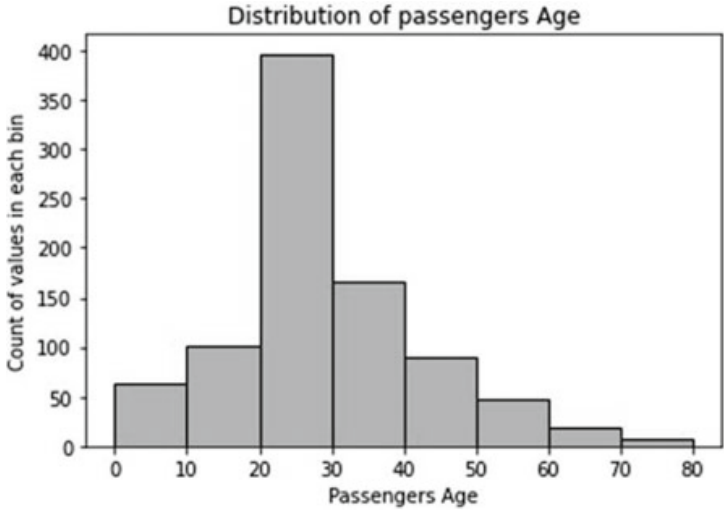
(a) **Using matplotlib**

*plt.***hist***(x, bins=None, edgecolor=None, orientation=None) x:* Array of values of a continuous variable
bins: Number of bins in the range of x.*min*() to x.*max*()
orientation: whether the bars are oriented vertically (default) or horizontally
edgecolor: Color of bin edges

Example

```
import pandas as pd
titanic=pd.read_csv('/content/sample_data/
  Titanic_train.csv')
age = titanic['Age'].fillna(titanic['Age'].mean())
# Specify bin edges
bin_edges= [0,10,20,30,40,50,60,70,80]
plt.hist(age,bins=bin_edges,color='orange',
  edgecolor='black', width=10)
plt.title('Distribution of passengers Age')
plt.xlabel('Passengers Age')
plt.ylabel('Count of values in each bin')
# X-axis values
plt.xticks(bin_edges)
plt.show()
```

Output



Note: In the above example, the variable, *bin_edges*, represents the interval ranges, i.e., the first bin is [0, 10) where 10 is not inclusive, [10, 20), [20,30), and so on. But in the last bin, [70, 80], the last value 80 is also inclusive.

(b) **Using Seaborn**

*sns.***histplot***(x, bins, fill, binwidth, kde, alpha)*

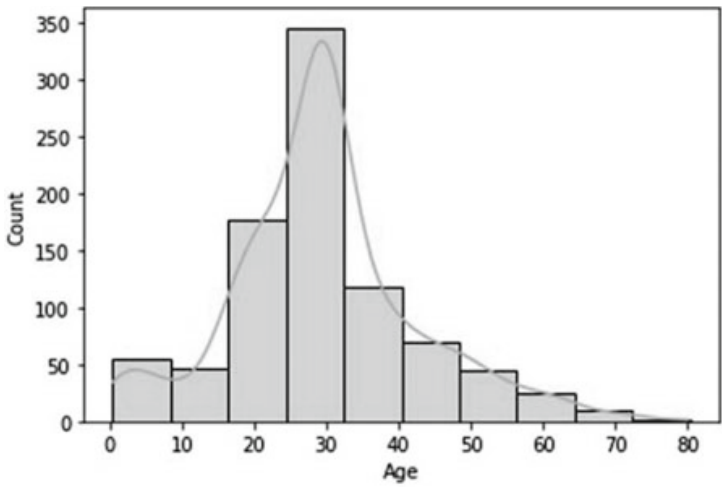
Following are the most used arguments

x: List of numeric values
bins: Integer that specifies the number of bins
or vector of values that specify bin edges
fill: whether to fill the histogram bins
binwidth: Width of bins as integer
kde: when set to true, displays the distribution as curve
alpha: opacity of bars. The value ranges from 0 to 1.

Example

```
import pandas as pd
titanic = pd.read_csv('/content/sample_data/Titanic.csv')
age = titanic['Age'].fillna(titanic['Age'].mean())
bin_edges= [0,10,20,30,40,50,60,70,80]
sns.histplot(x=age,bins=bin_edges,fill = True,
  color='orange', binwidth=8, kde = True, alpha=0.7)
plt.show()
```

Output



(iv) **Pie chart** is a circular graph representing the proportions of different components in the given whole. Each proportion or piece of the pie chart is called a wedge. Each wedge indicates a parts-of-whole relationship. Each value of the component is specified as the percentage and sum of all segments totaling to 100%. Pie charts are preferable when there are few components in the data. It is widely used in business applications to show the contribution of each item in the data.

(a) **Using matplotlib**

*plt.***pie***(x,colors=None,explode=None,*
labels=None,autopct=None,shadow=False)

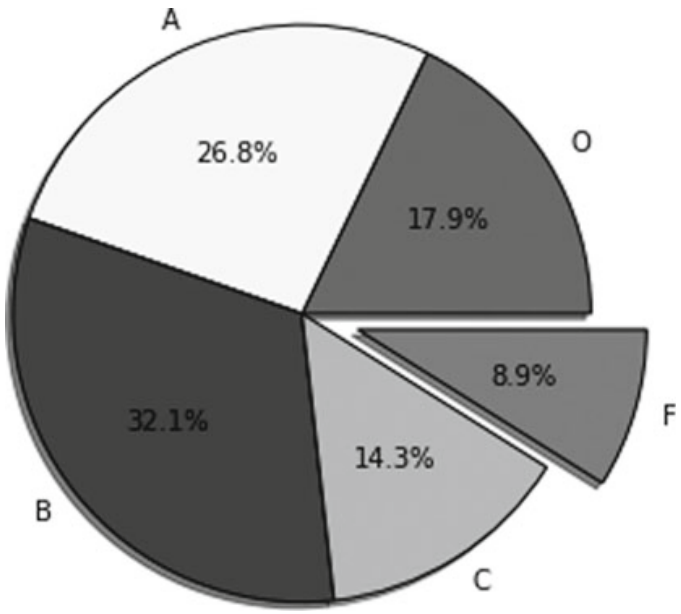
x:	Array of values
colors:	List of colors for each wedge
explode:	List of values representing how far the wedge is from the center
labels:	Array of labels for each wedge
autopct:	Label of the wedges indicating the proportion of the wedge as the percentage
shadow:	Boolean value indicating whether to add shadow to the pie chart

Example

```
results = [10, 15, 18, 8, 5]
grades = ['O','A','B','C','F']
expl = [0, 0, 0, 0, 0.2]
cols = ['green','yellow', 'purple', 'orange', 'red' ]
plt.figure(figsize=(5,5))
```

```
plt.pie(results, autopct='%1.1f%%', explode=expl,
  colors=cols, shadow=True, labels=grades)
plt.show()
```

Output

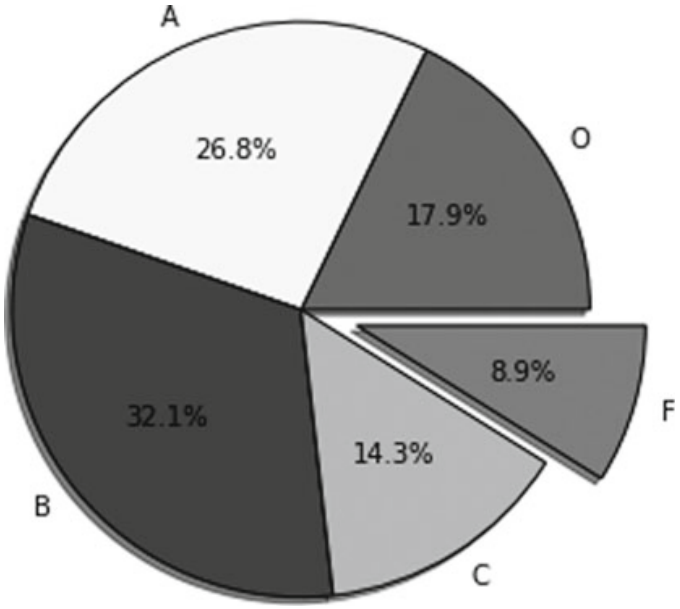


(b) **Seaborn** does not have built-in library function for the pie chart. Use pie( ) function of matplotlib and make use of seaborn color palette for wedges.

Example

```
results = [10, 15, 18, 8, 5]
grades = ['O','A','B','C','F']
# To highlight the weg
expl = [0, 0, 0, 0, 0.2]
cols= sns.color_palette('bright')
plt.pie(results, autopct='%1.1f%%', explode=expl,
  colors=cols, labels=grades)
plt.title('Distribution of grades among 60 students
  of a class')
plt.show()
```

Output



(v) **Bar chart** represents the relationship between categorical and corresponding numeric values. One axis represents the category and the other represents the numeric value. Each category is represented by a rectangular bar. The height of bars is proportional to corresponding numeric values. Same chart can also be used to compare two or more values in the same category. This graph is preferable and effective when data contains few categories. Bars can be laid either vertically or horizontally.

(a) **Using matplotlib**

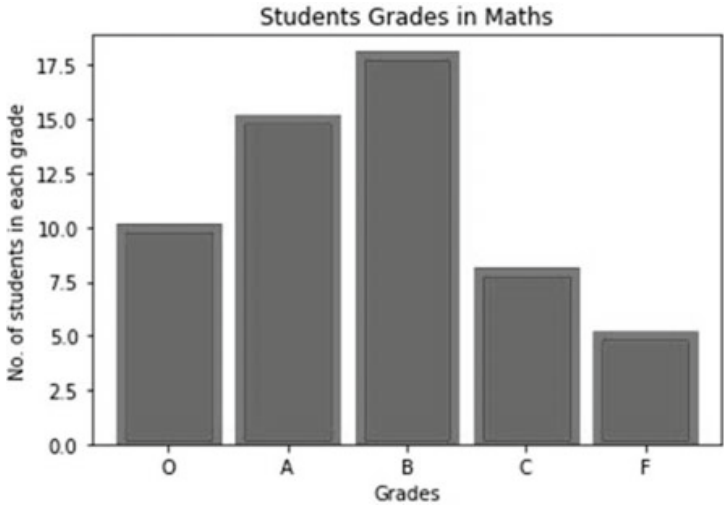
plt.bar(x, height, width, align, color, edgecolor, linewidth)

x: List of x-axis values or categories
height: Height of bar indicating y-axis values
width: Bars width
align: Alignment of bars to x coordinates. Default is center.
color: Color of bar faces
edgecolor: Color of bar edges

Example

```
maths = [10, 15, 18, 8, 5]
grades = ['O','A','B','C','F']
plt.bar(x=grades, height=maths, linewidth=5,
  edgecolor='red')
plt.xlabel('Grades')
plt.ylabel('No. of students in each grade')
plt.title('Students Grades in Maths')
plt.show()
```

Output

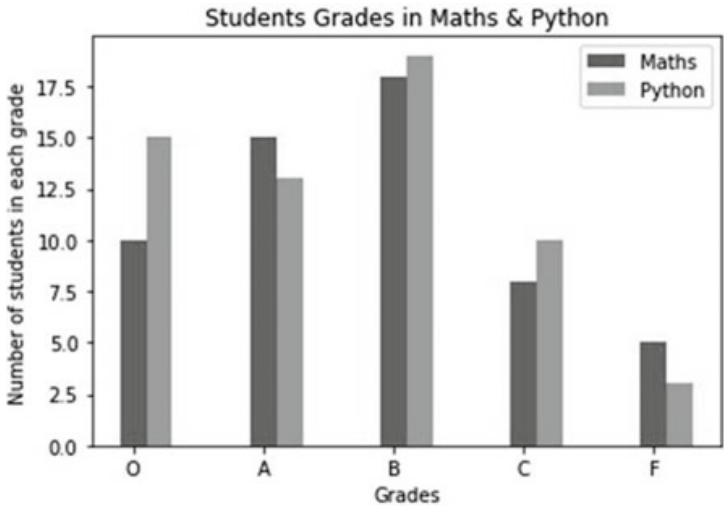


Multiple bar graphs in the same chart.

Example

```
idx1 = list(range(len(grades)))
idx2 = [i+0.2 for i in idx1]
python = [15, 13, 19, 10,3]
maths = [10, 15, 18, 8, 5]
grades = ['O','A','B','C','F']
plt.bar(grades, maths, width=0.2, label='Maths')
plt.bar(idx2, python, width=0.2, label='Python')
plt.title('Students Grades in Maths & Python')
plt.xlabel('Grades')
plt.ylabel('Number of students in each grade')
plt.legend()
plt.xticks(grades)
plt.show()
```

Output



(b) **Using seaborn**

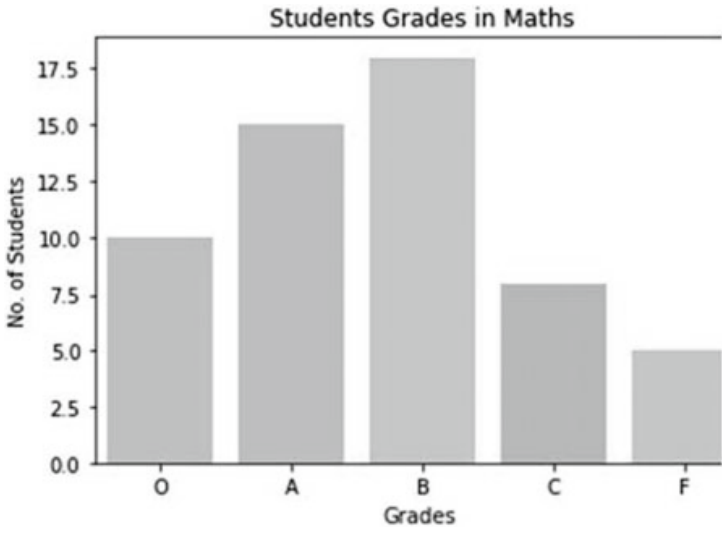
*sns.**barplot**(x, y, data, palette)*

x, y: Values along x and y axis
data: Data frame
palette: Color variations

Example

```
maths = [10, 15, 18, 8, 5]
grades = ['O','A','B','C','F']
g = sns.barplot(x = grades, y=maths, palette='pastel')
g.set_title('Students Grades in Maths')
g.set_ylabel('No. of Students')
g.set_xlabel('Grades')
plt.show()
```

Output



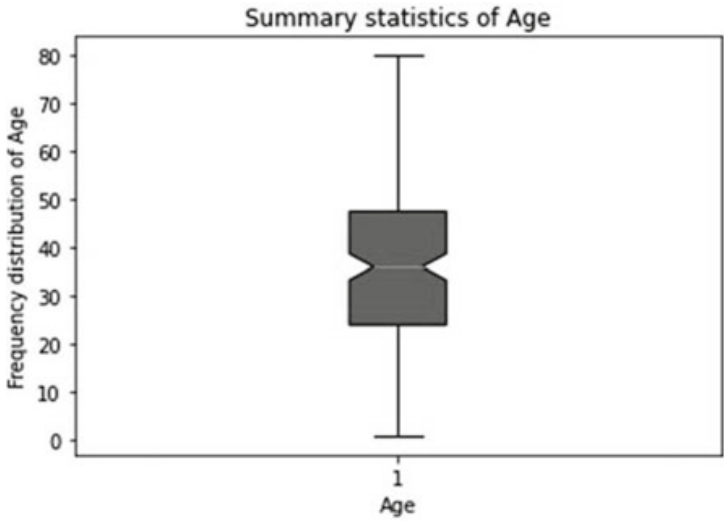
(vi) **Box Plot** is used to visualize data distribution based on five number summaries namely, minimum, first quartile, median, third quartile, and maximum values in the data. It is also called the whisker plot. It is a rectangular plot with lines extending from bottom to top. The box in the graph extends from quartile1 (Q1) to quartile3 (Q3) with a line indicating quartile2 (Q2) or median. It also shows the outliers.

*plt.***box***(data, notch=None, vert=None, patch_artist=None) data:* List of values.
vert: Whether to display the plot vertically or not.
notch: Whether a clear indication of median required.
patch_artist: Whether to fill the quartiles.

Example

```
import pandas as pd
titanic = pd.read_csv('/content/sample_data/Titanic_train.csv')
# Drop rows with null values
titanic = titanic.dropna()
plt.boxplot(titanic['Age'],vert=True, notch=True,
  patch_artist=True)
plt.xlabel('Age')
plt.ylabel('Frequency distribution of Age')
plt.title('Summary statistics of Age')
plt.show()
```

Output



Using seaborn

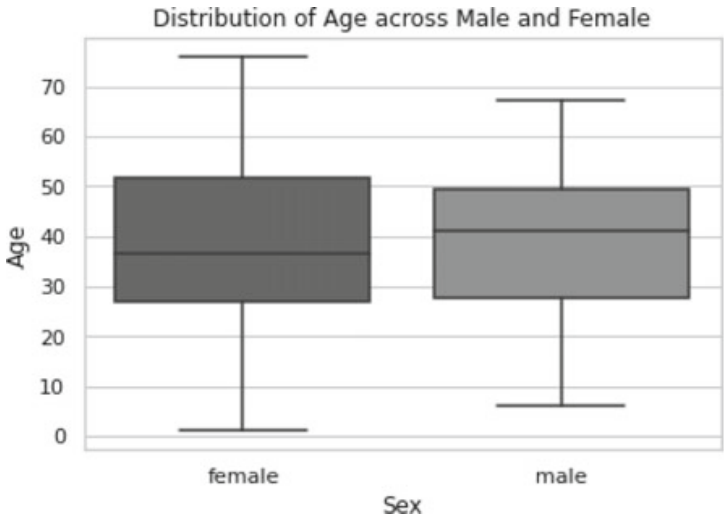
*sns.***boxplot***(x, y, data)*

data: Dataframe
x: Column of the dataframe on x-axis
y: Column of the dataframe on y-axis

Example

```
import pandas as pd
titanic = pd.read_csv('/content/sample_data/Titanic.csv')
# Drop rows with null values
titanic = titanic.dropna()
sns.set(style='whitegrid')
g=sns.boxplot(x = 'Sex', y = 'Age', data=titanic)
g.set_title('Distribution of Age across Male and Female')
plt.show()
```

Output



(vii) **Heatmap of seaborn**

Heatmap is a two-dimensional graphical representation of values in the matrix. The values are represented using different shades of color based on the magnitude of the value. Darker shades indicate higher data values.

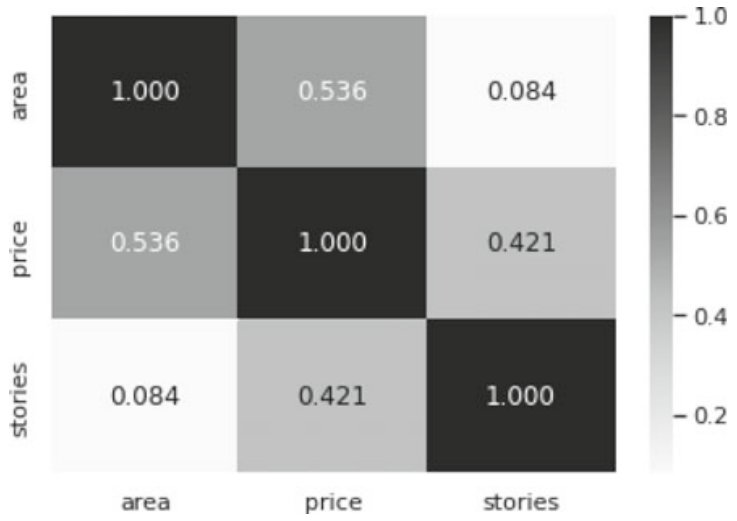
*seaborn.***heatmap***(data, annot, fmt, cmap, cbar)*

data:	Two-dimensional array of data
annot:	Boolean value. When set to True data values are written in the cells of the heat map. The default value is False
fmt:	String format to be used to specify the number of decimal places for the values. Default is two decimal places
cmap:	Map data values to color space
cbar:	Boolean value indicating whether to draw a color bar in the graph. Default value is True.

Example

```
import pandas as pd
df = pd.read_csv('/content/sample_data/Housing.csv')
df_new = df[['area','price','stories']]
# correlation among the attributes
cr = df_new.corr()
sns.heatmap(data= cr, annot=True, fmt='0.3f', cmap='Blues')
plt.show()
```

Output



(viii) **Facetgrid**

Facetgrid maps multiple axes into grid cells that show the distribution of a variable and relationship between multiple variables. It takes dataframe as input and categorical columns as row and column arguments to the functions. Data is split into subsets based on the categorical values. Each graph in the grid is the visualization of a subset of data.

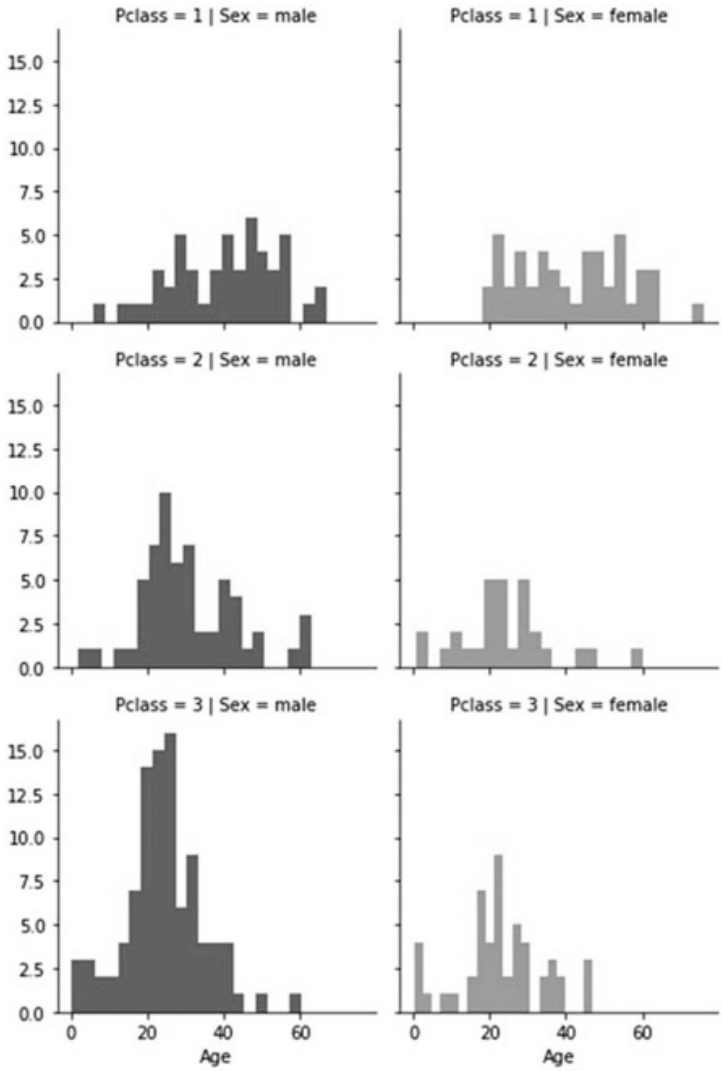
seaborn.*facetgrid(data, row, col, hue)*

data: Dataframe
row, col: Define subsets of dataframe based on categorical values.
hue: Different categories are plotted with different colors

Example

```
import pandas as pd
df = pd.read_csv('/content/sample_data/Titanic.csv')
gr = sns.FacetGrid(df, row='Pclass', col='Sex', hue='Sex')
gr.map(plt.hist, 'Age', bins=20)
```

Output



12.5 Subplots

Multiple plots can be drawn in one canvas, creating a grid of plots. Each plot is called a subplot.

figure_object, axis_objects = plt.*subplot(nrows, ncols, sharex, sharey)*

nrows, ncols:	Integers indicating the number of rows and columns in the grid. Default value is 1
sharex, sharey:	Controls sharing of x-axis or y-axis among the subplots True or 'all' :- share x-axis/y-axis for all subplots False :- each subplot x-axis/y-axis is independent 'row' :- x-axis/y-axis is shared along the row of subplots 'col' :- x-axis/y-axis is shared along the column of subplots.

Returns tuple containing figure_object and array of axis_objects.

We can set properties of each subplot like title, xlabel, ylabel using

set_title(), set_xlabel(), set_ylabel()

Set the title for the entire figure using

figure_object.***suptitle***(text)

Unused grid cell in the figure can be made empty using

*axis_object.***axis***('off')*

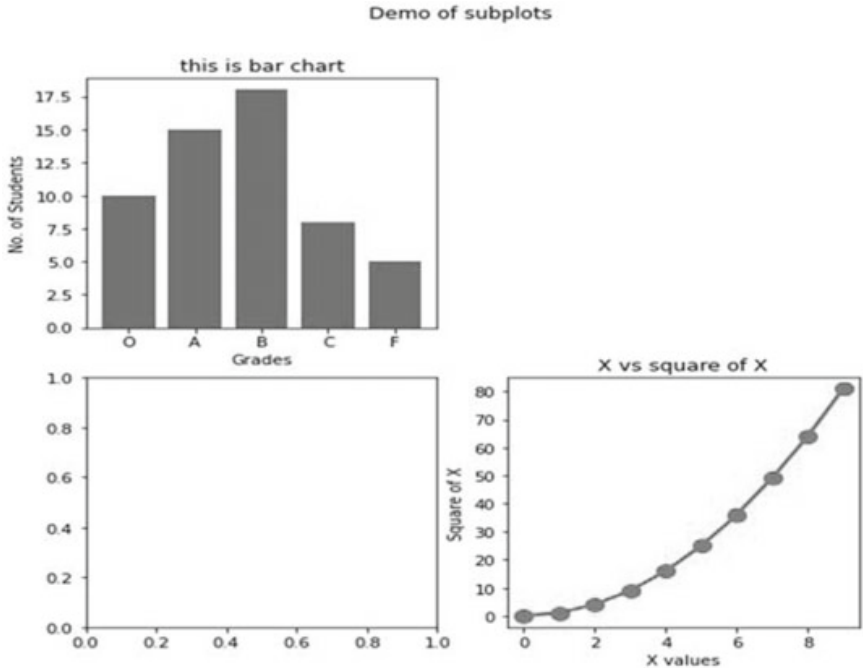
Example

```
results = [10, 15, 18, 8, 5]
grades = ['O','A','B','C','F']
fig, ax = plt.subplots(2,2, figsize=(8,8))
fig.suptitle("Demo of subplots")
# Draw bar graph
ax[0,0].bar(grades,results)
ax[0,0].set_title("this is bar chart")
ax[0,0].set_xlabel('Grades')
ax[0,0].set_ylabel('No. of Students')

x = list(range(10))
y1 = [ i *i for i in x]
# Draw line graph
ax[1,1].plot(x, y1, color='red', lw = 2.0, ls = 'solid', marker = 'o',
  markersize = 10)
ax[1,1].set_title("X vs square of X")
ax[1,1].set_xlabel('X values')
ax[1,1].set_ylabel('Square of X')

# Hiding the subplot at 0,1 index
ax[0,1].axis('off')
plt.show()
```

Output



12.6 Case Study: Data Visualizations

Perform data visualizations on the mobile dataset consisting of 21 attributes and 2000 instances. The attributes include continuous and categorical data types. Library functions of **Matplotlib** and **Seaborn** modules are used to draw the graphs.

(i) Load the dataset into a dataframe.

```
import pandas as pd
df = pd.read_csv('/content/sample_data/mobile_data.csv')
print('Dataset shape : ',df.shape)
print('Columns : ',df.columns)
```

Output

```
Dataset shape : (2000, 21)
Columns : Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc',
'four_g','int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc',
'px_height','px_width', 'ram', 'sc_h', 'sc_w', 'talk_time',
'three_g','touch_screen', 'wifi', 'price_range'],
  dtype='object')
```

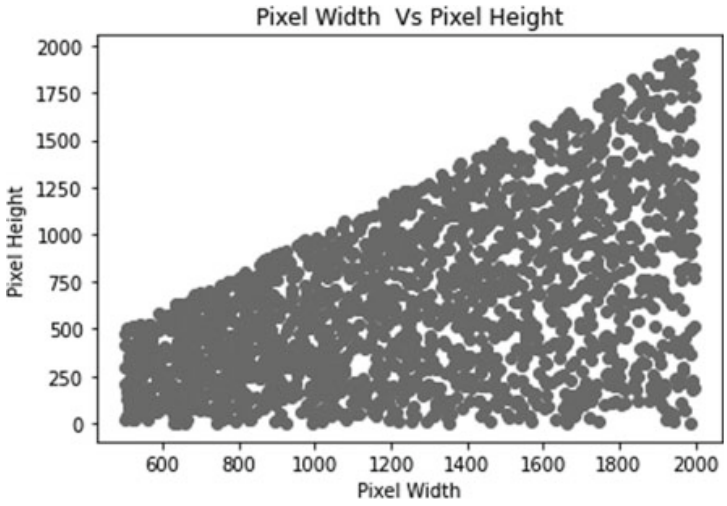
(ii) Import the visualization libraries—*matplotlib* and *seaborn.*

```
import matplotlib.pyplot as plt
import seaborn as sb
```

(iii) Visualize the correlation between *pixel_width* and *pixel_height* using a *scatter* plot of matplotlib. pyplot.

```
plt.scatter(x=df['px_width'], y=df['px_height'])
plt.xlabel('Pixel Width')
plt.ylabel('Pixel Height')
plt.title('Pixel Width Vs Pixel Height')
plt.show()
```

Output

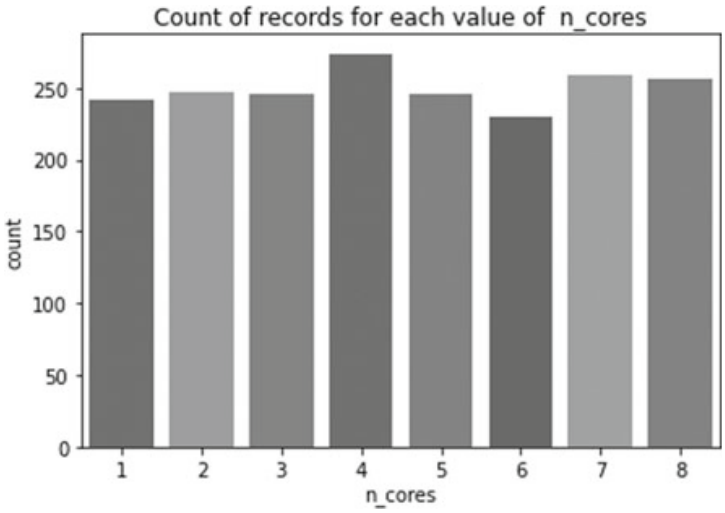


The plot shows a positive correlation between px_width & px_height.

(iv) Count the number of records for each value of the *n_cores* attribute using *countplot()* function of *seaborn*.

```
sb.countplot(df['n_cores'])
plt.title('Count of records for each value of n_cores')
plt.show()
```

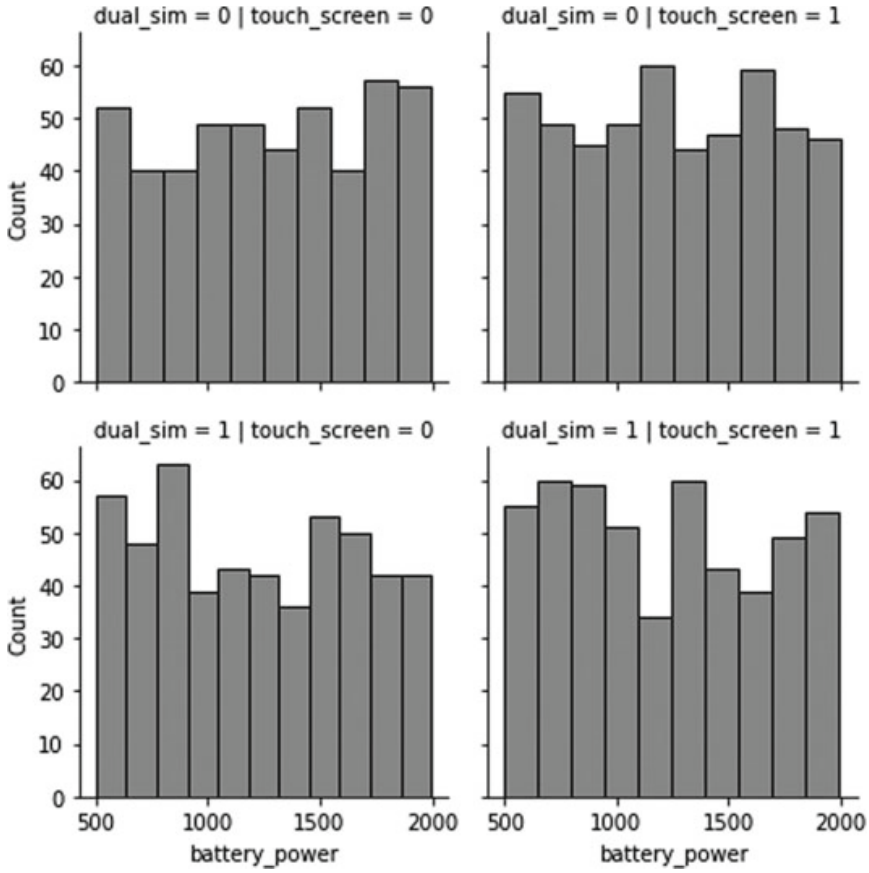
Output



(v) Let us draw a grid of histogram plots on *battery_power* over different values of *dual_sim & touch_screen*.

```
fg = sb.FacetGrid(df, row='dual_sim', col='touch_screen' )
fg.map(sb.histplot, 'battery_power')
plt.show()
```

Output

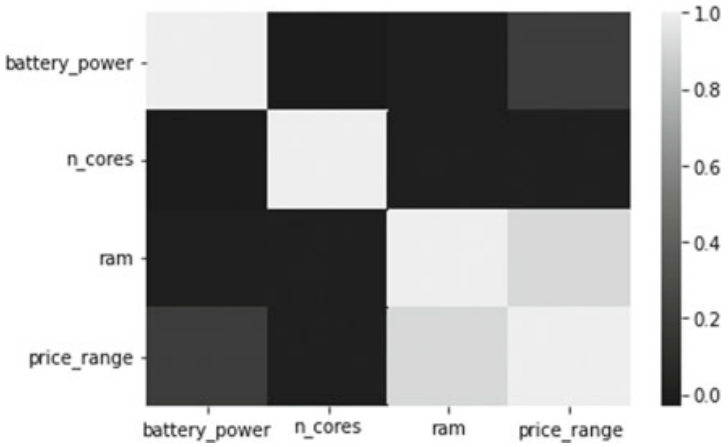


(vi) Let us check the correlation among *price_range*, *battery_power*, *n_cores*, *ram* attributes using *corr()* function on dataframe. Then visualize the correlation using the *heatmap()* function of *seaborn*.

```
cols= ['battery_power','n_cores','ram', 'price_range' ]
cm = df[cols].corr()
print(cm)
sb.heatmap(cm)
plt.show()
```

Output

	battery_power	n_cores	ram	price_range
battery_power	1.000000	−0.029727	−0.000653	0.200723
n_cores	−0.029727	1.000000	0.004868	0.004399
ram	−0.000653	0.004868	1.000000	0.917046
price_range	0.200723	0.004399	0.917046	1.000000

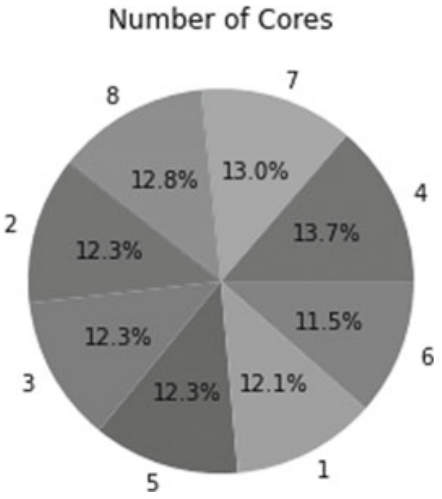


The visualization shows a high positive correlation between *ram* and *price_range*.

(vii) Visualize the proportion of instances on each value of *n_cores* using a *pie* chart of *matplotlib*.

```
cores= df['n_cores'].value_counts()
lab = cores.index
plt.pie(cores.values, autopct='%1.1f%%',labels=lab)
plt.title('Number of Cores')
plt.show()
```

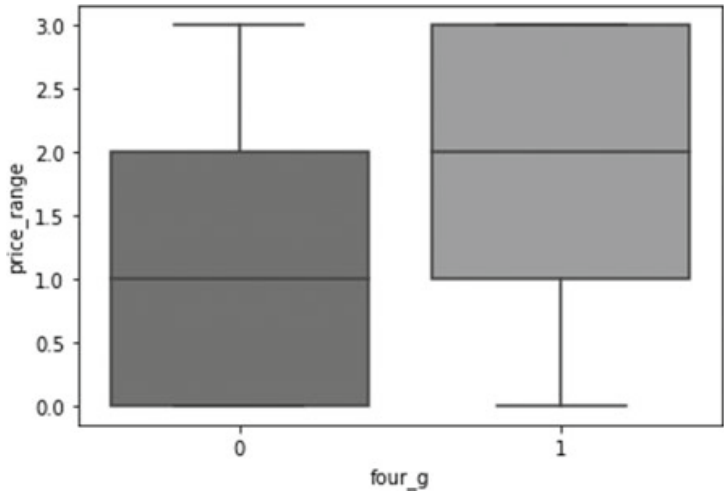
Output



(viii) Plot the data distribution summary on *four_g* and *price_range* attributes using *boxplot()* of *seaborn*.

```
sb.boxplot(x= 'four_g', y='price_range', data=df)
plt.show()
```

Output



Exercises

1. Take a public dataset, identify dependent and independent variables, and draw an attractive scatter plot by taking appropriate arguments.
2. Take COVID-19 data of total confirmed and deceased cases of the top five countries. Draw pie charts (one for confirmed and one for deceased) and highlight the country with maximum cases. Give appropriate title and percentage in the wedges.

3. Take a public dataset containing both categorical and continuous variables. Draw a facetgrid chart.
4. Take a public dataset and draw box plot, line graph, pie chart, and bar graph in the same canvas using subplots. Give appropriate title and labels for each graph.
5. Take COVID-19 data of total confirmed and deceased cases of the top ten countries. Draw a bar graph of confirmed and deceased cases in the same chart. Use appropriate arguments to make the graph attractive.

Review Questions

(1) What is the default location for **legend** in a given plot?

(a) Upper left
(b) Upper right
(c) Center
(d) best

(2) The built-in function for generating a **scatter plot** using matplotlib is

(a) regplot()
(b) reg()
(c) scatter()
(d) scatterplot()

(3) The ———— argument of the **hist()** function represents the number of bars for given data x.

(a) bins()
(b) x.min()
(c) x.max()
(d) bars()

(4) The middle horizontal line in a box plot represents ————.

(a) Quartile 1
(b) Quartile 2
(c) Quartile 3
(d) Mean

(5) The ————— attribute of **heatmap()** represents the data values to be written in each cell.

(a) cmap
(b) annot
(c) cbar
(d) fmt

(6) Which plot among the following represents the relationship between dependent variable and independent variable?

(a) Bar
(b) Scatter
(c) Line
(d) MultiLine

(7) What does the **is explode** argument in a Pie Chart mean?

(a) Proportions of Different Component
(b) Color of Wedge
(c) Distance of wedge from the center
(d) Size of Wedge

(8) Which of the following graphs is used to visualize changes in data over time?

(a) Line
(b) Pie
(c) Histogram
(d) Box

(9) Which of the following is false about Searborn built-in functions?

(a) Seaborn has built-in pie chart()
(b) Seaborn has lineplot()
(c) Seaborn is built on top of Matplotlib
(d) Seaborn has histplot()

(10) The histplot() built-in function in seaborn has an argument **kde** that accepts ———————— value to display distribution as curve.

(a) Integer
(b) Float
(c) Boolean
(d) String