Foundation of Data Science
(IT305B)

Prepared by

Mr. Umesh B. Sangule
Assistant Professor

Department of Information Technology

Unit-VI
DATA VISUALIZATION

Course Objectives : *To apply the data visualization techniques,*

Course Outcome(CO6) : *Apply various data visualization techniques using python libraries*

Describing data relationship

Correlation:

- A correlation exists between two variables when there is a relationship (or an association) between them.

  For example: ***Does the familiar saying "You get what you give" accurately describe the exchange of greeting cards?***

- An investigator suspects that a relationship exists between the number of greeting cards sent and the number of greeting cards received by individuals.

- If the suspected relationship does exist between cards sent and cards received, then an inspection of the data might reveal, as one possibility, a tendency for "big senders" to be "big receivers" and for "small senders" to be "small receivers."

Describing data relationship

1. Positive Relationship:

- two variables are **positively related** if pairs of scores tend to occupy similar relative positions (high with high and low with low) in their respective distributions.

POSITIVE RELATIONSHIP

FRIEND	SENT	RECEIVED
Doris	13	14
Steve	9	18
Mike	7	12
Andrea	5	10
John	1	6

- **For example:** John sent relatively few cards (1) and received relatively few cards (6), whereas Doris sent relatively many cards (13) and received relatively many cards (14).
  - ❖ *this relationship implies that **"You get what you give."** Insofar as relatively low values are paired with relatively low values, and relatively high values are paired with relatively high values, the **relationship is positive**.*

Describing data relationship

2. Negative Relationship:

- two variables are **negatively related** if pairs of scores tend to occupy dissimilar relative positions (high with low and vice versa) in their respective distributions..

NEGATIVE RELATIONSHIP

FRIEND	SENT	RECEIVED
Doris	13	6
Steve	9	10
Mike	7	14
Andrea	5	12
John	1	18

- **For example:** John sent relatively few cards (1), he received relatively many (18). From this pattern, we can conclude that the two variables are related.
  - ❖ *this relationship implies that **"You get the opposite of what you give."** Insofar as relatively low values are paired with relatively high values, and relatively high values are paired with relatively low values, **the relationship is negative**.*

Describing data relationship

3. Little or No Relationship:

- No regularity is apparent among the pairs of scores

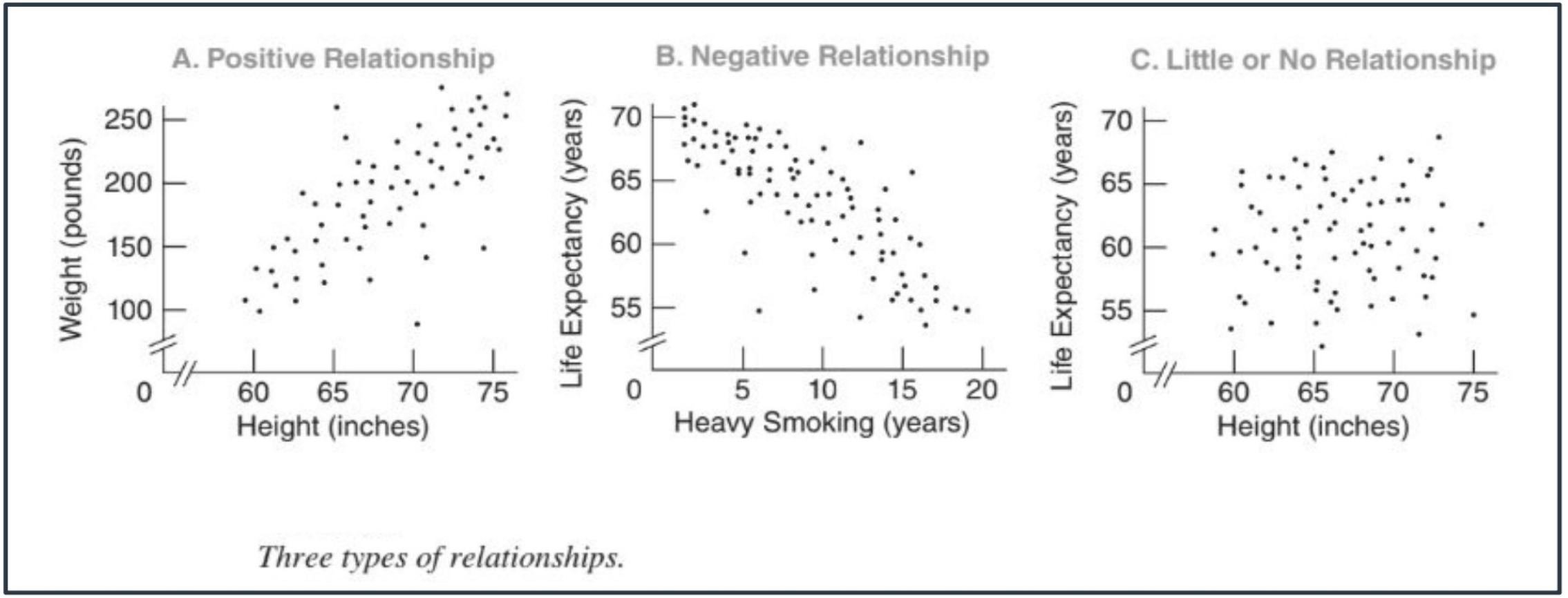
LITTLE OR NO RELATIONSHIP

FRIEND	SENT	RECEIVED
Doris	13	10
Steve	9	18
Mike	7	12
Andrea	5	6
John	1	14

- **For example:** Andrea and John sent relatively few cards (5 and 1, respectively), Andrea received relatively few cards (6) and John received relatively many cards (14).
  - ❖ *From this lack of regularity, we can conclude that little, if any, relationship exists between the two variables and that* ***"What you get has no bearing on what you give."***

Describing data relationship

Positive, Negative, Little or No Relationship:



Three types of relationships.

Describing data relationship

Positive, Negative, Little or No Relationship:

- A dot cluster that has a ***slope from the lower left to the upper right***, reflects a **positive relationship**.
  - ❖ *Small values of one variable are paired with small values of the other variable, and large values are paired with large values.*
- A dot cluster that has a ***slope from the upper left to the lower right***, reflects a **negative relationship**.
  - ❖ *Small values of one variable tend to be paired with large values of the other variable, and vice versa.*
- A dot cluster that ***lacks any apparent slope***, reflects ***little or no relationship***.
  - ❖ *Small values of one variable are just as likely to be paired with small, medium, or large values of the other variable.*

Bar Graph/Chart:

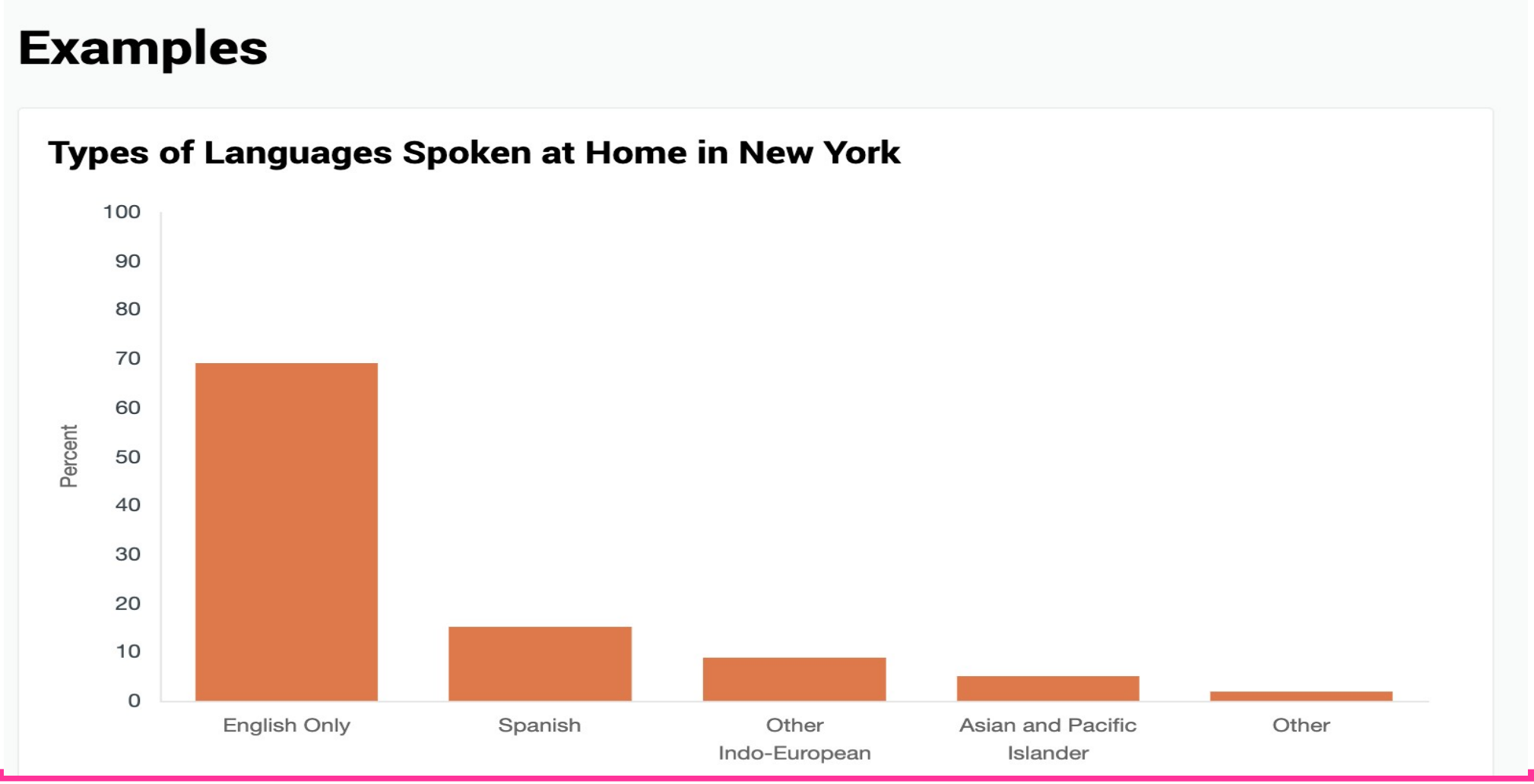
- A bar chart displays categorical data with rectangular bars whose length or height corresponds to the value of each data point.

- Bar charts can be visualized using vertical or horizontal bars. Bar charts are best used to compare a single category of data or several. When comparing more than one category of data, the bars can be grouped together to created a grouped bar chart.

- Bar charts use volume to demonstrate differences between each bar. Because of this, bar charts should always start at zero. When bar charts do not start at zero, it risks users misjudging the difference between data values.

Importing and visualization using Matplotlib/Seaborn python library

Bar Graph/Chart:



Bar Graph/Chart:

Requirements

Always

- Use independent categories of data.
- Label each category of data.
- When visualizing data between 0% and 100%, start tick marks at 0%.

Never

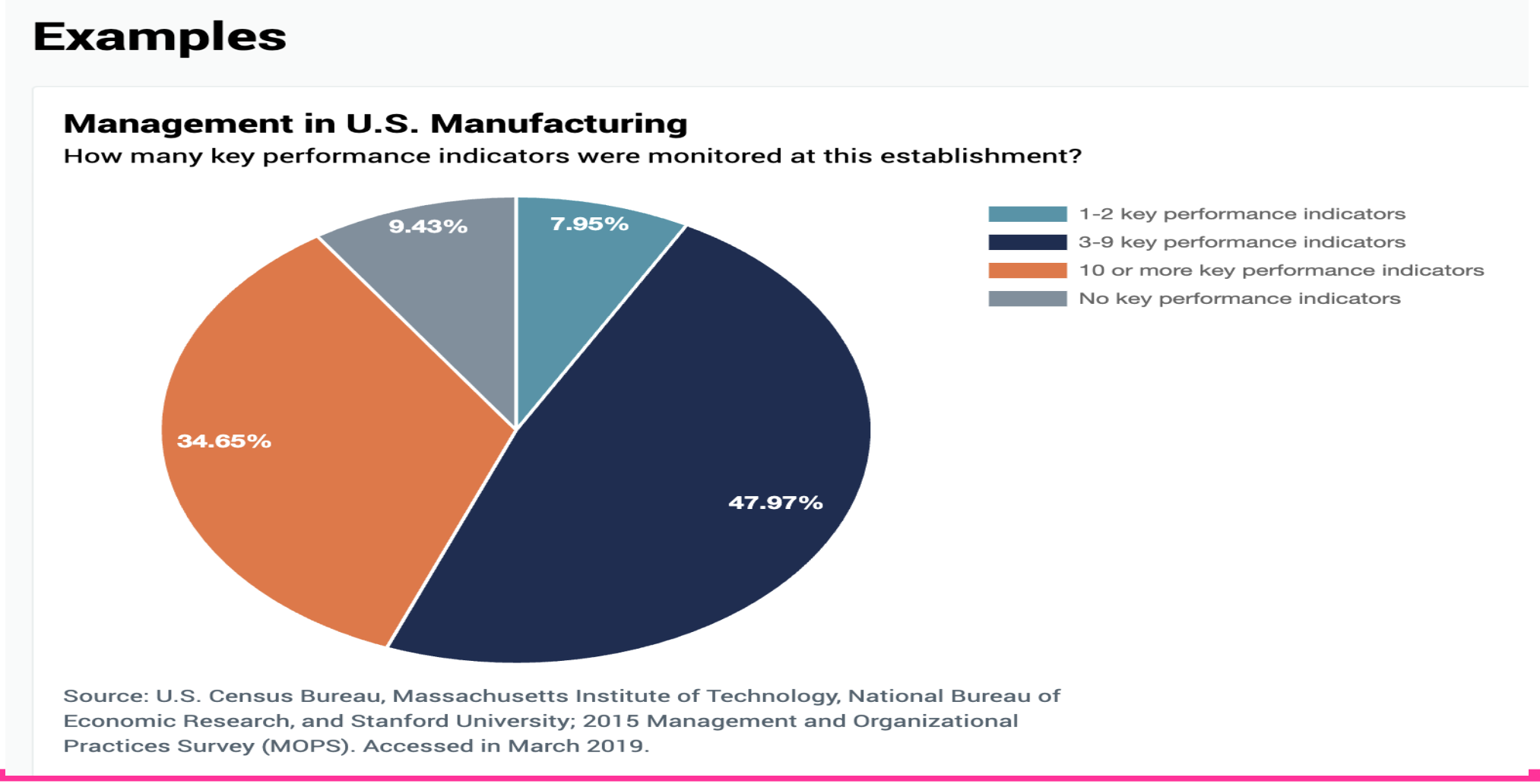
- Never omit the space between bars — otherwise the bar chart will appear to be a histogram.
- Never use three dimensional (3D) graphics as they distort the visual calculation of volume.

Pie-Chart:

- A pie chart is a circular graph that presents values as proportionate slices.
- Pie charts are one of the most commonly used data visualizations but are often not the most effective way to compare data values.
- This is mainly due to the size of each slice and the number of data categories being represented.
- Pie charts can also be represented as a doughnut chart, which is generally better for users to compare the size of each slice or arc.

Importing and visualization using Matplotlib/Seaborn python library

Pie-Chart:



Pie-Chart:

Requirements

Always

- Include a legend or label slices directly.
- Ensure that the values of the slices total to 100%.

Never

- Never use three dimensional (3D) graphics as they distort the visual calculation of volume.

Recommendations

Not Recommended

- Don’t include large gaps between each slice, such as in an exploded pie chart.
- Don’t use multi-level pie charts as they are difficult to decipher.

Line Graph/Plot:

- A line graph is a chart that displays a series of data points connected by line segments.

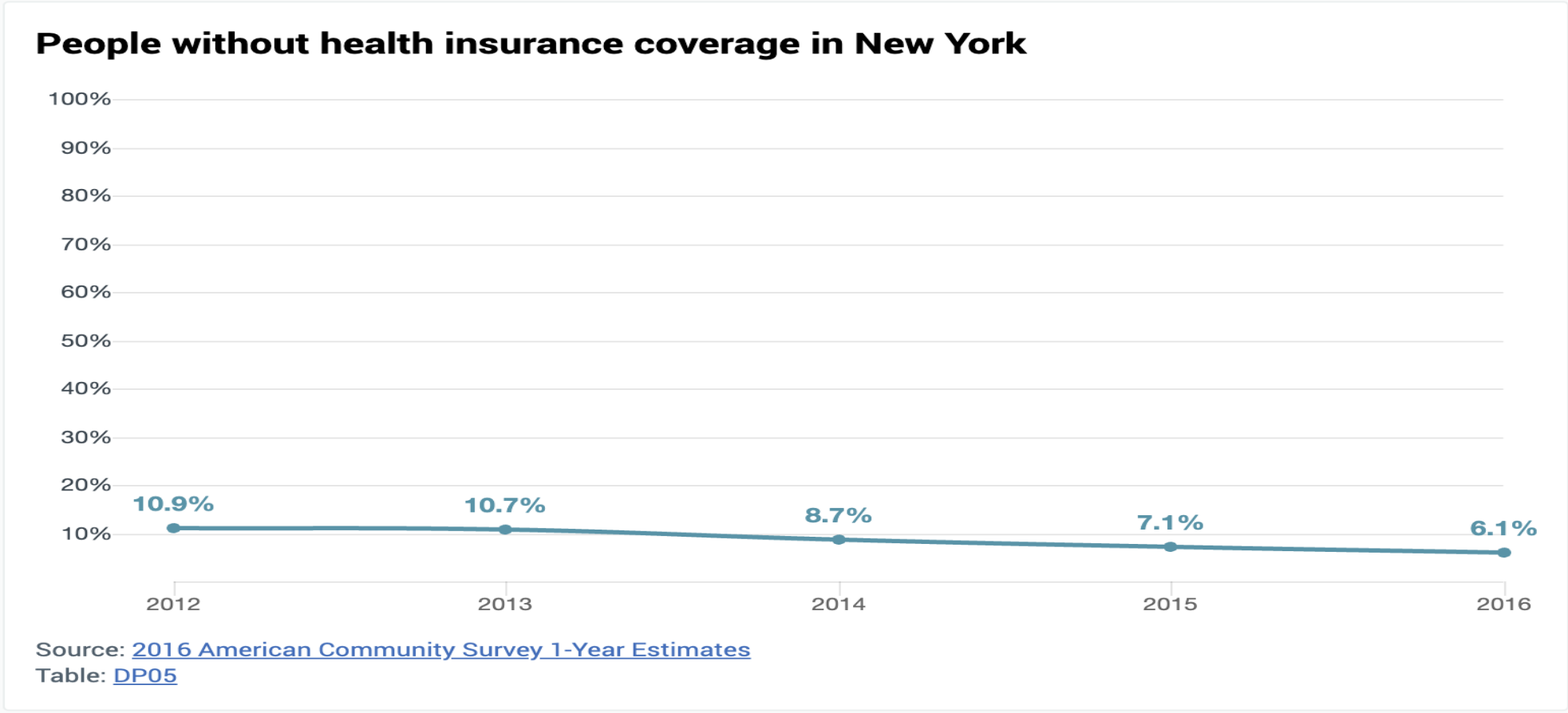
- Line graphs can be used to show how data changes over time and are often used to communicate trends, such as how household income changes each year.

- The x-axis can display continuous or discrete data such as days, years, or categories, and the y-axis is typically a continuous variable but can also be discrete. Line graphs can also be used to compare multiple trend lines, whereas bar charts can only represent one trend line.

Importing and visualization using Matplotlib/Seaborn python library

Line Graph/Plot:

Examples



Line Graph/Plot:

Requirements

Always

- Start the y-axis at zero.
- Label each line if there is more than one line.
- Ensure that each line is legible.

Never

- Never use a legend for a graph with a single line.

Recommendations

Not Recommended

- Don’t use horizontal lines unless conveying exact amounts.

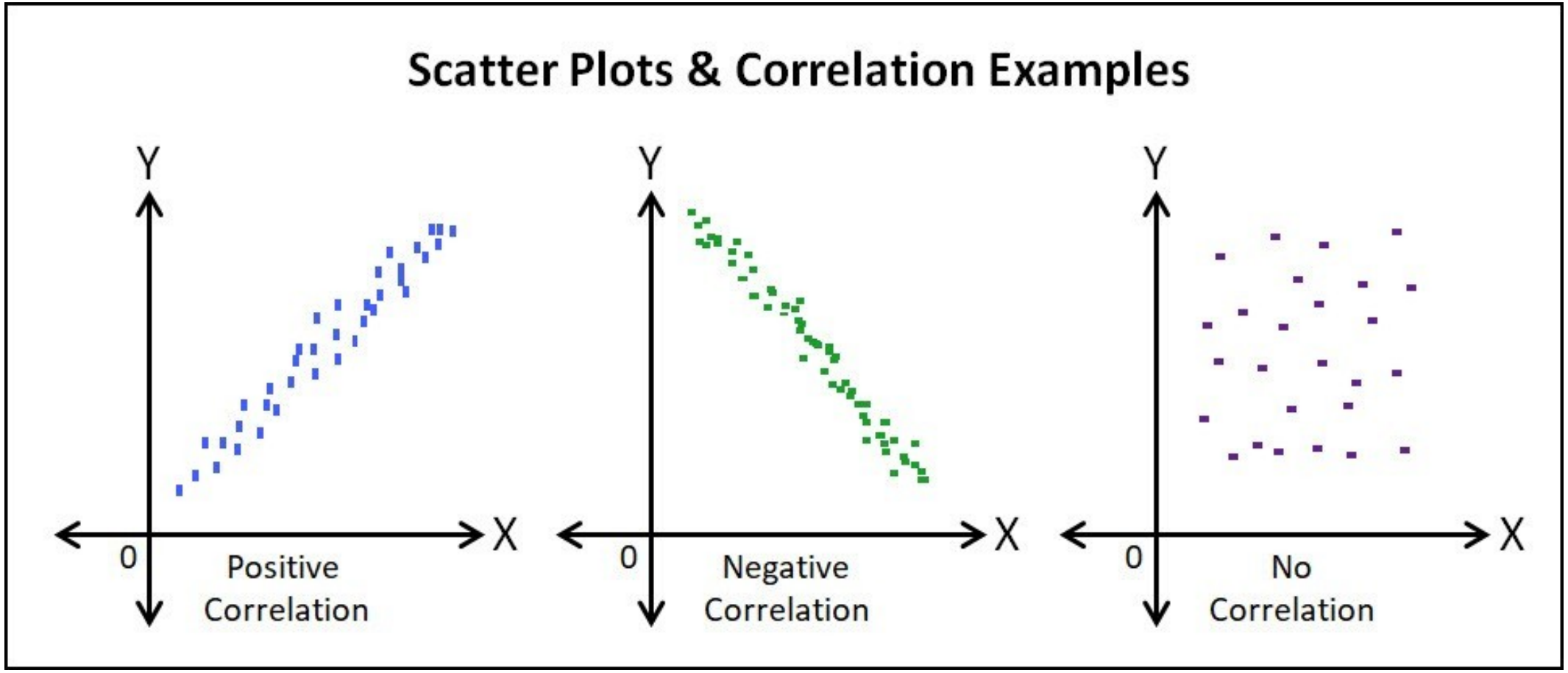
Scatter Plot:

- A scatter plot is a two-dimensional chart that shows the relationship between two variables.

- To do this, a series of values are plotted on an x-axis and a y-axis, with each axis representing a variable and each value representing a dot. Scatter plots often include an explanatory variable, such as years of education, and what might be considered a response variable, such as annual income.

- When the dots are plotted against these two axes, scatter plots communicate the strength and type of relationship that exists between these variables.

Scatter Plot:



Importing and visualization using Matplotlib/Seaborn python library

Scatter Plot:

Requirements

Always

- Include a legend if more than one set of values is being visualized.

Never

- Never overlap labels for points.
- Never use complex symbols as point markers.
- Never include more than 2 sets of values.

Recommendations

Recommended

- The title should explain the unit of analysis.
- Include point labels or markers for specific observations.

Not Recommended

- Don’t use a scatter plot chart if there are an excessive number of overlapping values.

Importing and visualization using Matplotlib/Seaborn python library

Histogram:

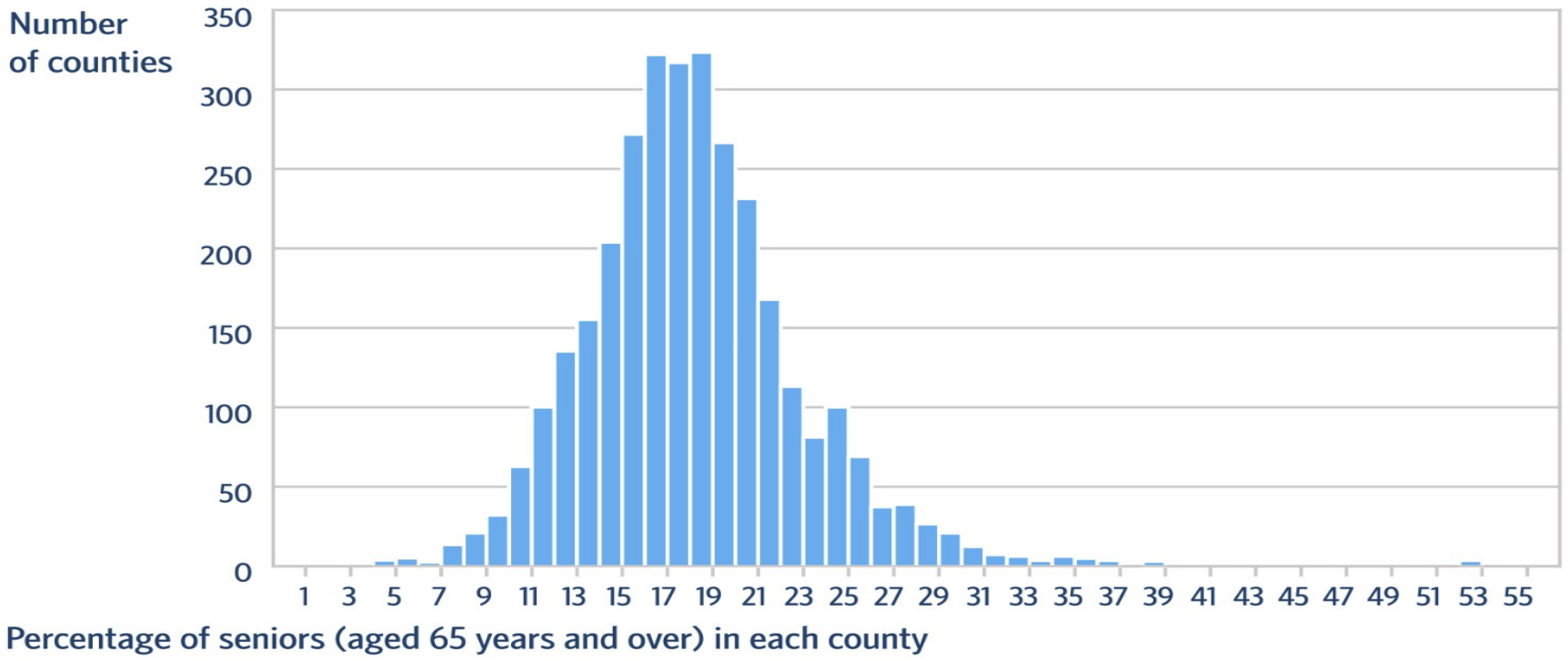
- A histogram is a chart that displays numeric data in ranges, where each bar represents how frequently numbers fall into a particular range.
- Like a bar chart, histograms consist of a series of vertical bars along the x-axis.
- Histograms are most commonly used to depict what a set of data looks like in aggregate.
- At a quick glance, histograms tell whether a dataset has values that are clustered around a small number of ranges or are more spread out.

Histogram:

Examples

Static Histogram

Item 1: Percentage of Population Aged 65 and Older



Importing and visualization using Matplotlib/Seaborn python library

Histogram:

Requirements

Always

- Include the frequency measurement on the y-axis and ranges on the x-axis.
- Start the y-axis at zero.
- Use a single color for all bars, except when using one additional color to highlight a single bar.
- Use at least three intervals/ranges.

Never

- Never add spacing between bars. Bars span the entire range of values for the continuous variable.

Recommendations

Not Recommended

- Don’t orient a histogram upside down.

Importing and visualization using Matplotlib/Seaborn python library

➢ *MatPlotLib:*

Python Matplotlib. matplotlib.pyplot is a python package used for 2D graphics.

- What Is Python Matplotlib?
- What is Matplotlib used for?
- Types Of Plots
  - Bar Graph
  - Histogram
  - Scatter Plot
  - Area Plot
  - Pie Chart

What Is Python Matplotlib?

matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

Importing and visualization using Matplotlib/Seaborn python library

➢ *MatPlotLib:*

What is Matplotlib used for?

Matploitlib is a Python Library used for plotting, this python library provides and objected-oriented APIs for integrating plots into applications.

Is Matplotlib Included in Python?

Matplotlib is not a part of the Standard Libraries which is installed by default when Python, there are several toolkits which are available that extend python matplotlib functionality. Some of them are separate downloads, others can be shipped with the matplotlib source code but have external dependencies.

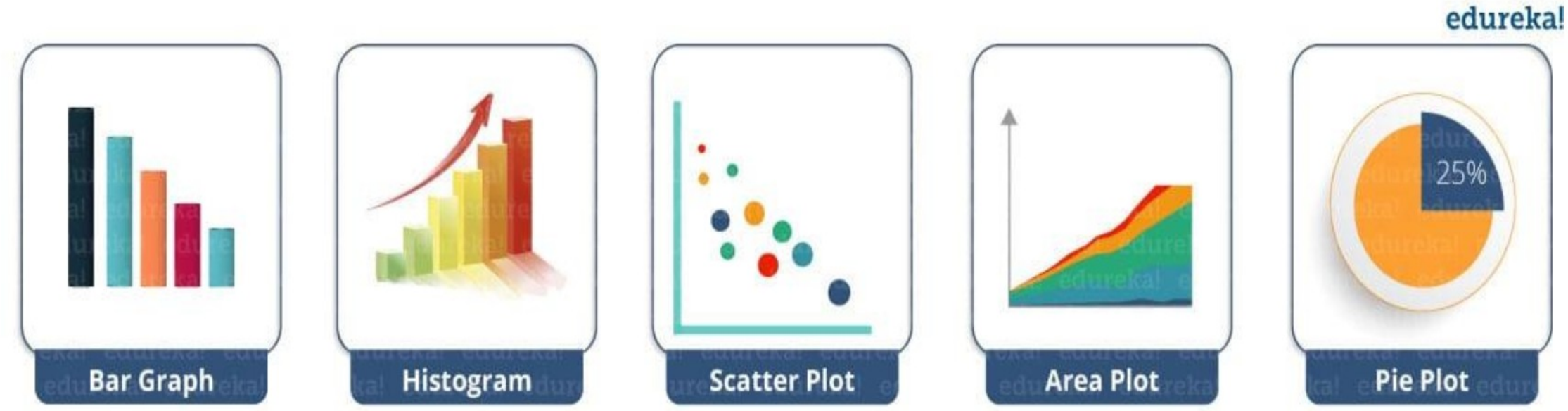
- **Basemap**: It is a map plotting toolkit with various map projections, coastlines and political boundaries.
- **Cartopy**: It is a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities.
- **Excel tools**: Matplotlib provides utilities for exchanging data with Microsoft Excel.

Importing and visualization using Matplotlib/Seaborn python library

➢ *MatPlotLib:*

Python Matplotlib : Types of Plots

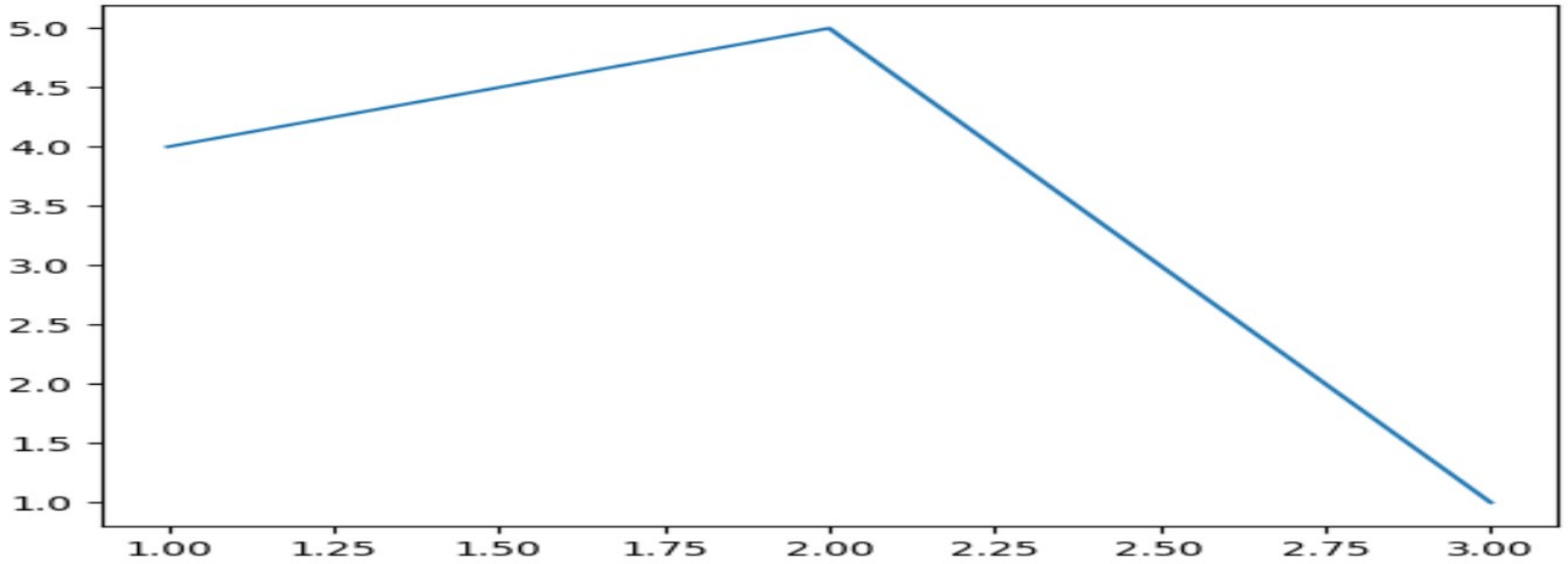
There are various plots which can be created using python matplotlib. Some of them are listed below:



Importing and visualization using Matplotlib/Seaborn python library

➢ *MatPlotLib:*

```
from matplotlib import pyplot as plt
  #Plotting to our canvas
  plt.plot([1,2,3],[4,5,1])
  #Showing what we plotted
  plt.show()
```



➢ *Scatterplot using Matplotlib:*

With Pyplot, you can use the scatter() function to draw a scatter plot.

The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

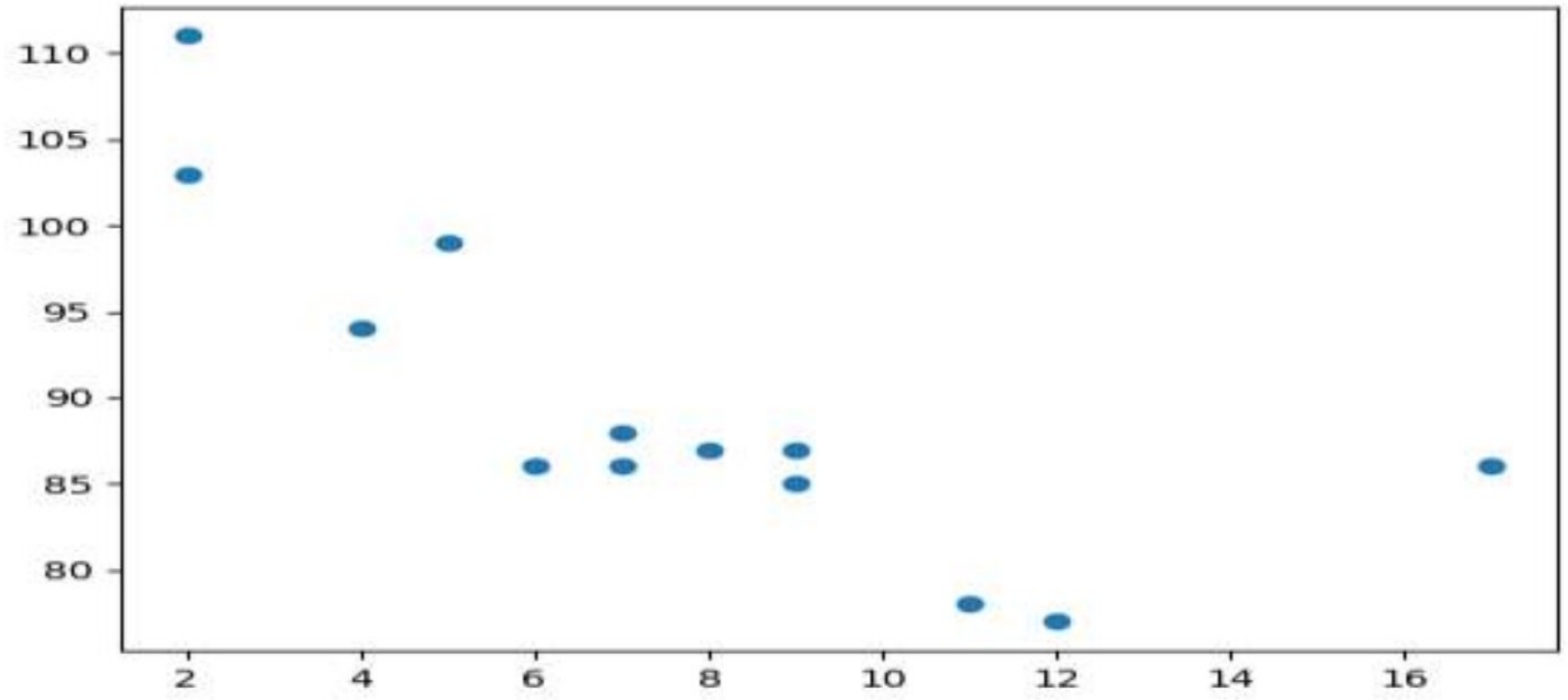
Example

A simple scatter plot:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```

Importing and visualization using Matplotlib/Seaborn python library

➢ ***Scatterplot using Matplotlib:***



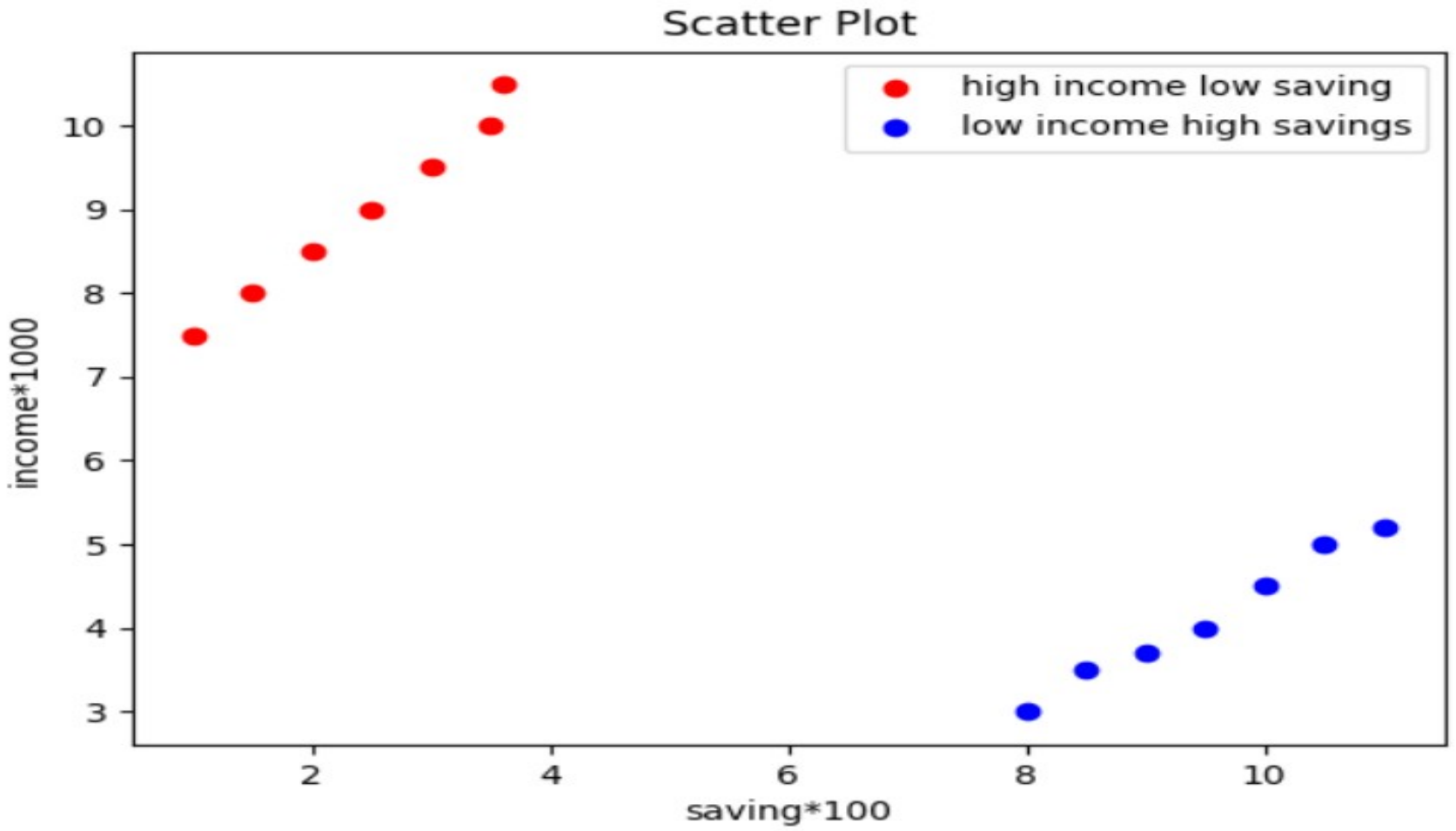
Importing and visualization using Matplotlib/Seaborn python library

➢ *Scatterplot using Matplotlib:*

```
import matplotlib.pyplot as plt
x = [1,1.5,2,2.5,3,3.5,3.6]
y = [7.5,8,8.5,9,9.5,10,10.5]

x1=[8,8.5,9,9.5,10,10.5,11]
y1=[3,3.5,3.7,4,4.5,5,5.2]

plt.scatter(x,y, label='high income low saving',color='r')
plt.scatter(x1,y1,label='low income high savings',color='b')
plt.xlabel('saving*100')
plt.ylabel('income*1000')
plt.title('Scatter Plot')
plt.legend()
plt.show()
```

Importing and visualization using Matplotlib/Seaborn python library

➢ ***Scatterplot using Matplotlib:***



➢ ***Barplot using Matplotlib:***

First, let us understand why do we need a bar graph. A bar graph uses bars to compare data among different categories. It is well suited when you want to measure the changes over a period of time. It can be represented horizontally or vertically. Also, the important thing to keep in mind is that longer the bar, greater is the value. Now, let us practically implement it using python matplotlib.

➢ ***Barplot using Matplotlib:***

With Pyplot, you can use the bar() function to draw bar graphs:

Example

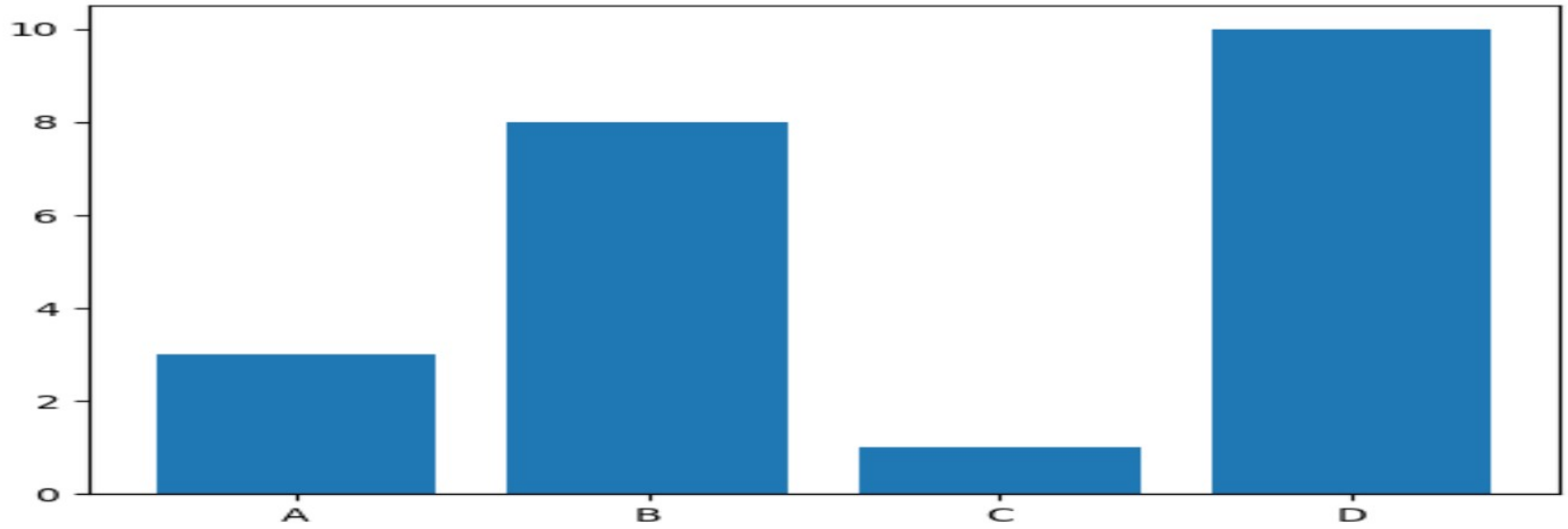
Draw 4 bars:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```

Importing and visualization using Matplotlib/Seaborn python library

➢ *Barplot using Matplotlib:*

Result:



Importing and visualization using Matplotlib/Seaborn python library

➢ ***Histogram using Matplotlib:***

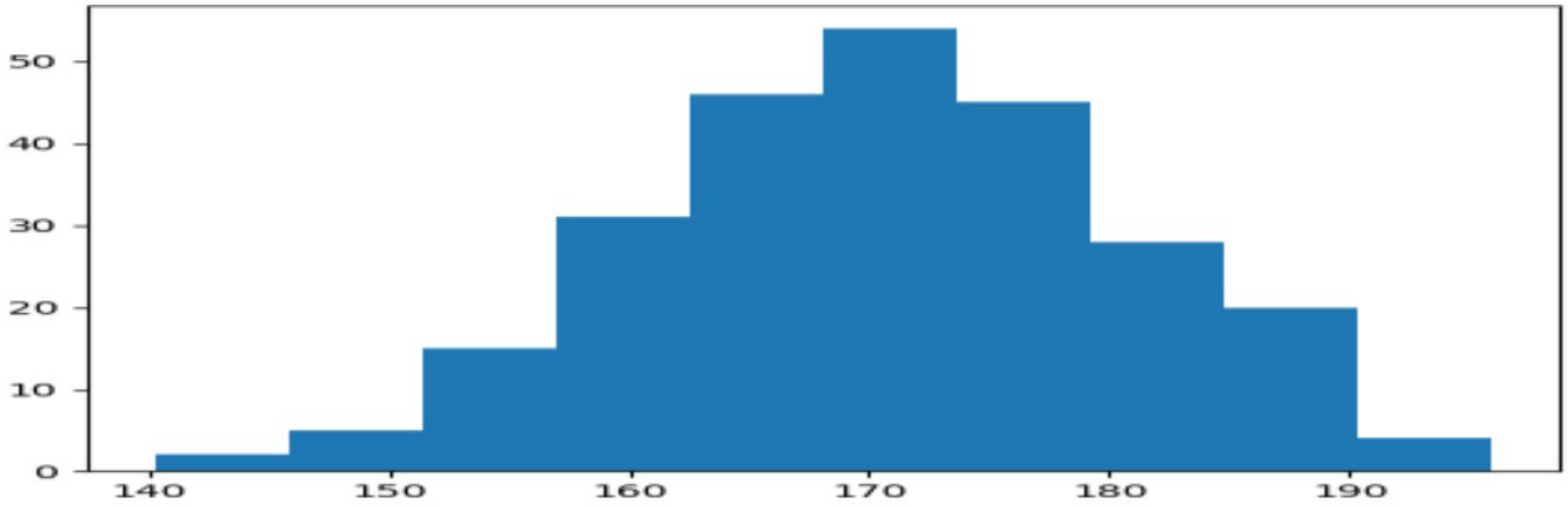
A simple histogram:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```

Importing and visualization using Matplotlib/Seaborn python library

➢ ***Histogram using Matplotlib:***

Result:



Importing and visualization using Matplotlib/Seaborn python library

➢ *Pie-Chart using Matplotlib:*

A simple pie chart:

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

Importing and visualization using Matplotlib/Seaborn python library

➢ *Pie-Chart using Matplotlib:*

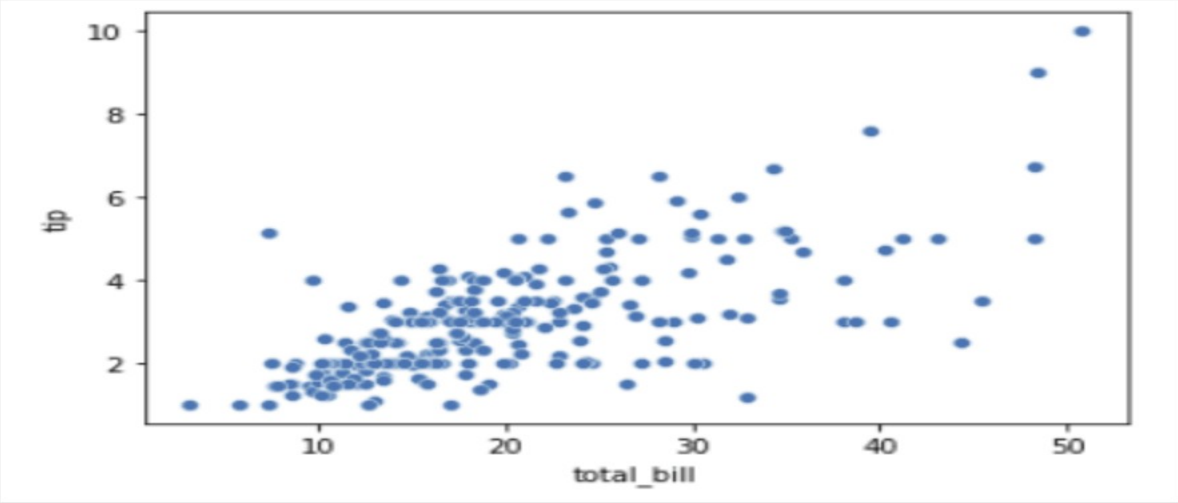
Result:

Importing and visualization using Matplotlib/Seaborn python library

➢ *Scatterplot using Seaborn:*

```
import seaborn as sns

tips = sns.load_dataset("tips")

sns.scatterplot(x="total_bill", y="tip", data=tips)
```

Output:

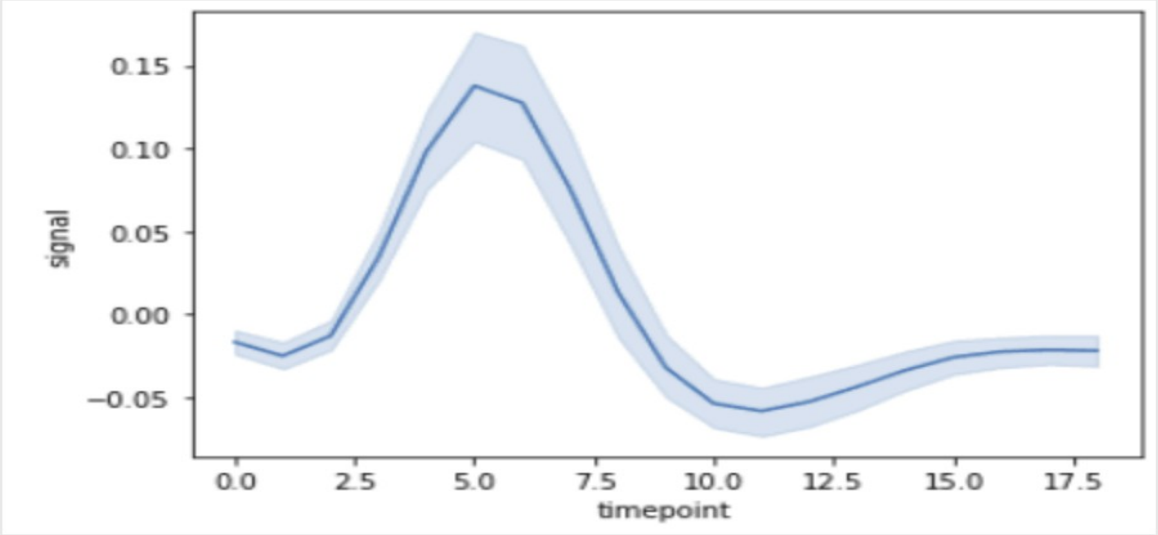


Importing and visualization using Matplotlib/Seaborn python library

➢ *Line Plot using Seaborn:*

```
import seaborn as sns

fmri = sns.load_dataset("fmri")

sns.lineplot(x="timepoint", y="signal", data=fmri)
```

Output:

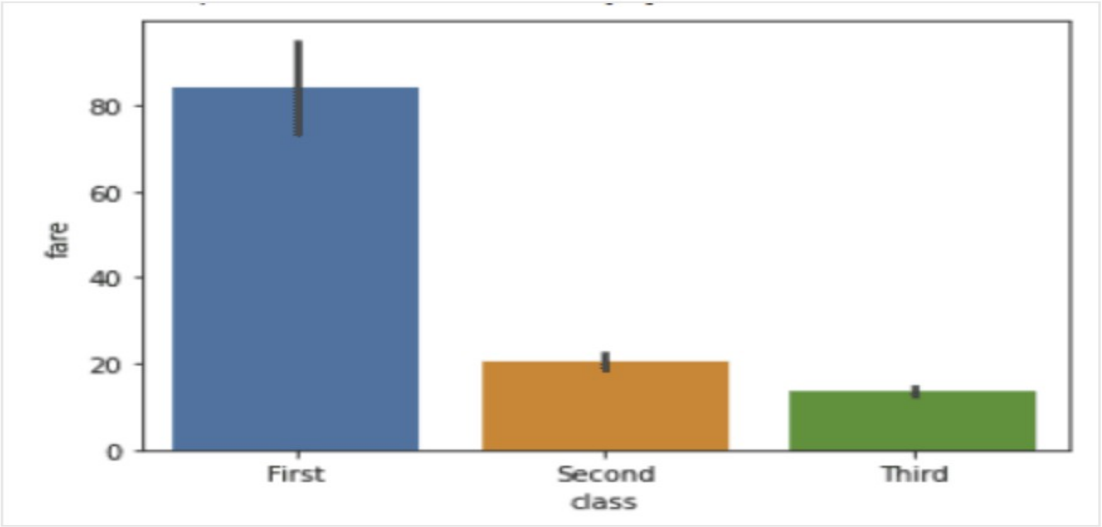


Importing and visualization using Matplotlib/Seaborn python library

➢ *Bar Plot using Seaborn:*

```
import seaborn as sns

titanic = sns.load_dataset("titanic")

sns.barplot(x="class", y="fare", data=titanic)
```

Output:

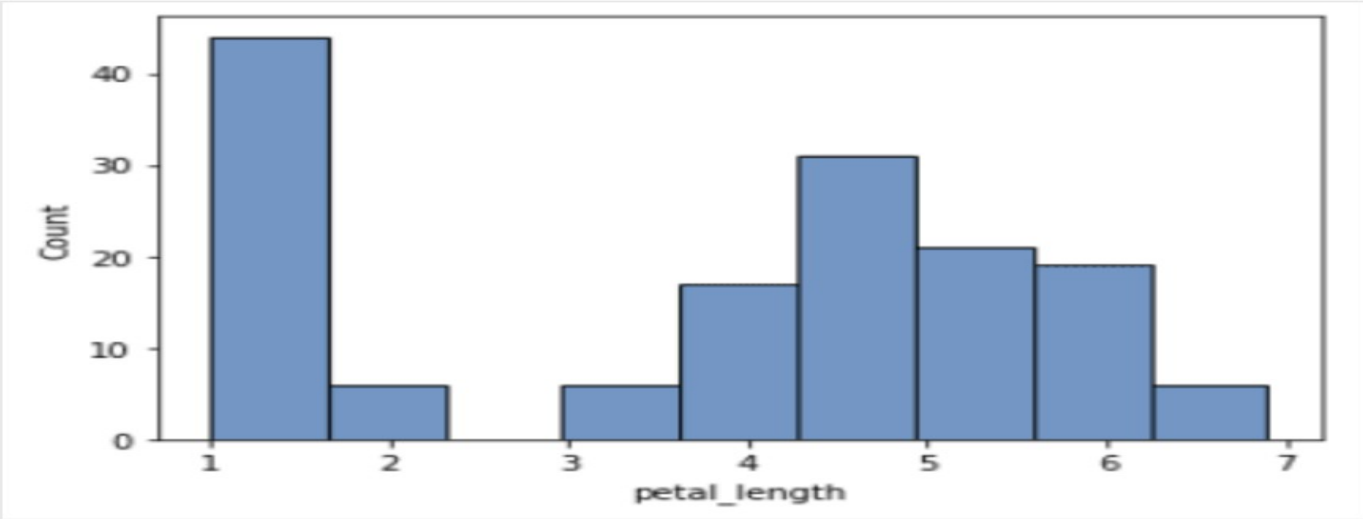


Importing and visualization using Matplotlib/Seaborn python library

➢ *Histogram using Seaborn:*

```
import seaborn as sns

iris = sns.load_dataset("iris")

sns.histplot(x="petal_length", data=iris)
```

Output:



Exploratory Data Analysis(EDA) Using Pandas library

- ***Exploratory Data Analysis(EDA):***
- Exploratory Data Analysis(EDA) is an important component as well as one of the most under-estimated steps in any Data Science project.
- EDA is essential for well-defined and structured data analysis and should be performed before the machine learning modeling phase.
- It involves finding insights from the data upon careful observation and further summarizing its main characteristics. Generally, the real-life data which we work upon contains a lot of ‘noise’, and therefore performing data analysis manually on such datasets becomes a complicated and tedious process.

Exploratory Data Analysis(EDA) Using Pandas library

- ***Exploratory Data Analysis(EDA):***

- Accordingly, Pandas is one of the most popular libraries of Python that helps to present the data in a way which is suitable for analysis via its Series and DataFrame data structures.

- It provides various functions and methods to both simplify as well as expedite the data analysis process.

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis(EDA):*

1. df.head( ): By default, it returns the first 5 rows of the Dataframe. To change the default, you may insert a value between the parenthesis to change the number of rows returned.

```
In [2]: df.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

2. df.tail( ): By default, it returns the last 5 rows of the Dataframe. This function is used to get the last n rows. This function returns the last n rows from the object based on position.

```
In [3]: df.tail()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis(EDA):*

3. df.info( ): It helps in getting a quick overview of the dataset. This function is used to get a brief summary of the dataframe. This method prints information about a DataFrame including the index dtype and column dtypes, non-null values, and memory usage.

```
In [7]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```


4. df.shape: It shows the number of dimensions as well as the size in each dimension. Since data frames are two-dimensional, what shape returns is the number of rows and columns.

```
In [4]: df.shape
Out[4]: (891, 12)
```

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis(EDA):*

5. df.size: Return an int representing the number of elements in this object. Return the number of rows if Series, otherwise returns the number of rows times the number of columns if DataFrame.

```
In [5]: df.size
Out[5]: 10692
```

6. df.ndim: Returns dimension of dataframe/series. 1 for one dimension (series), 2 for two dimensions (dataframe).

```
In [6]: df.ndim
Out[6]: 2
```

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis(EDA):*

7. df.describe( ): Return a statistical summary for numerical columns present in the dataset. This method calculates some statistical measures like percentile, mean and standard deviation of the numerical values of the Series or DataFrame.

```
In [8]: df.describe()
Out[8]:
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

8. df.sample( ): Used to generate a sample randomly either row or column. It allows you to select values randomly from a Series or DataFrame. It is useful when we want to select a random sample from a distribution.

```
In [9]: df.sample()
Out[9]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
64	65	0	1	Stewart, Mr. Albert A	male	NaN	0	0	PC 17605	27.7208	NaN	C

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis(EDA):*

9. df.isnull( ).sum( ): Return the number of missing values in each column.

```
In [10]: df.isnull().sum()

Out[10]: PassengerId      0
         Survived         0
         Pclass           0
         Name             0
         Sex              0
         Age            177
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          687
         Embarked         2
         dtype: int64
```

10. df.nunique( ): Return number of unique elements in the object. It counts the number of unique entries over columns or rows. It is very useful in categorical features especially in cases where we do not know the number of categories beforehand.

```
In [11]: df.nunique()

Out[11]: PassengerId    891
         Survived         2
         Pclass           3
         Name           891
         Sex              2
         Age             88
         SibSp            7
         Parch            7
         Ticket         681
         Fare           248
         Cabin          147
         Embarked         3
         dtype: int64
```

Exploratory Data Analysis(EDA) Using Pandas library

- ***Applications of Exploratory Data Analysis(EDA):***
- Grasping the overall size and structure of the data.

- Getting familiar with the available variables (columns) and deciding which of them would be potentially useful for solving the main problem of the project. In a real-world context, we do not analyze the data for the sake of the analysis itself. Instead, our goal is to answer a particular business question stated by the company.

- Identifying among the selected columns those with eventual issues. Such issues can be missing values, wrong data types, incorrect data representation, predominant values, duplicates, outliers, etc.

Exploratory Data Analysis(EDA) Using Pandas library

- ***Applications of Exploratory Data Analysis(EDA):***
- Detecting the unique values of the columns in interest, determining their meaning (especially if the dataset documentation does not provide enough information), and exploring their distribution.

- Figuring out the ways of dealing with the identified issues through data gathering, data cleaning, and data transformation.

- Exploring the statistics for the columns in interest and checking their distribution forms.
- Analyzing the correlation between various columns and validating their presence or absence.

Exploratory Data Analysis(EDA) Using Pandas library

- ***Importance of Pandas in EDA:***
- In this Code Example, if to ignore loading the data, importing the library, and printing outputs, we virtually ran only 15 lines of very simple code for EDA in pandas, as follows:

```
df.sample(3)
df.shape
df.isnull().sum().sum()
df.size
df.dtypes
df['company_size'].unique()
df.describe()
df['job_title'].nunique()
df['work_year'].is_monotonic_increasing
df['employment_type'].value_counts(normalize=True)
df['salary_in_usd'].nsmallest(3)
df['salary_in_usd'].nlargest(3)
df.corr()
df.plot(x='salary', y='salary_in_usd', kind='scatter')
df['salary_in_usd'].plot(kind='hist')
```

Exploratory Data Analysis(EDA) Using Pandas library

- ***Exploratory Data Analysis (EDA) Using Pandas Profiling:***
- Pandas profiling is a Python library that performs an automated Exploratory Data Analysis. It automatically generates a dataset profile report that gives valuable insights,

```
import numpy as np
import pandas as pd
```

Let's import the pandas profiling library:

```
from pandas_profiling import ProfileReport
```

```
data = pd.read_csv("datasets/Telco-Customer-Churn.csv")
```

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis (EDA) Using Pandas Profiling:*

To generate the standard profiling report, merely run:

```
profile = ProfileReport(data, title="Pandas Profiling Report")
```

```
data.profile_report()
```

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis (EDA) Using Pandas Profiling:*

To generate a HTML report file, save the `ProfileReport` to an object and use the `to_file()` function:

```
profile.to_file("your_report.html")
```

Alternatively, the report's data can be obtained as a JSON file:

```
# As a JSON string
json_data = profile.to_json()
```

```
# As a file
profile.to_file("your_report.json")
```

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis (EDA) Using Pandas Profiling:*

Overview: mostly global details about the dataset (number of records, number of variables, overall missingness and duplicates, memory footprint)

Overview

Overview | Alerts 25 | Reproduction

Dataset statistics

Number of variables	21
Number of observations	100
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	21.3 KiB
Average record size in memory	217.7 B

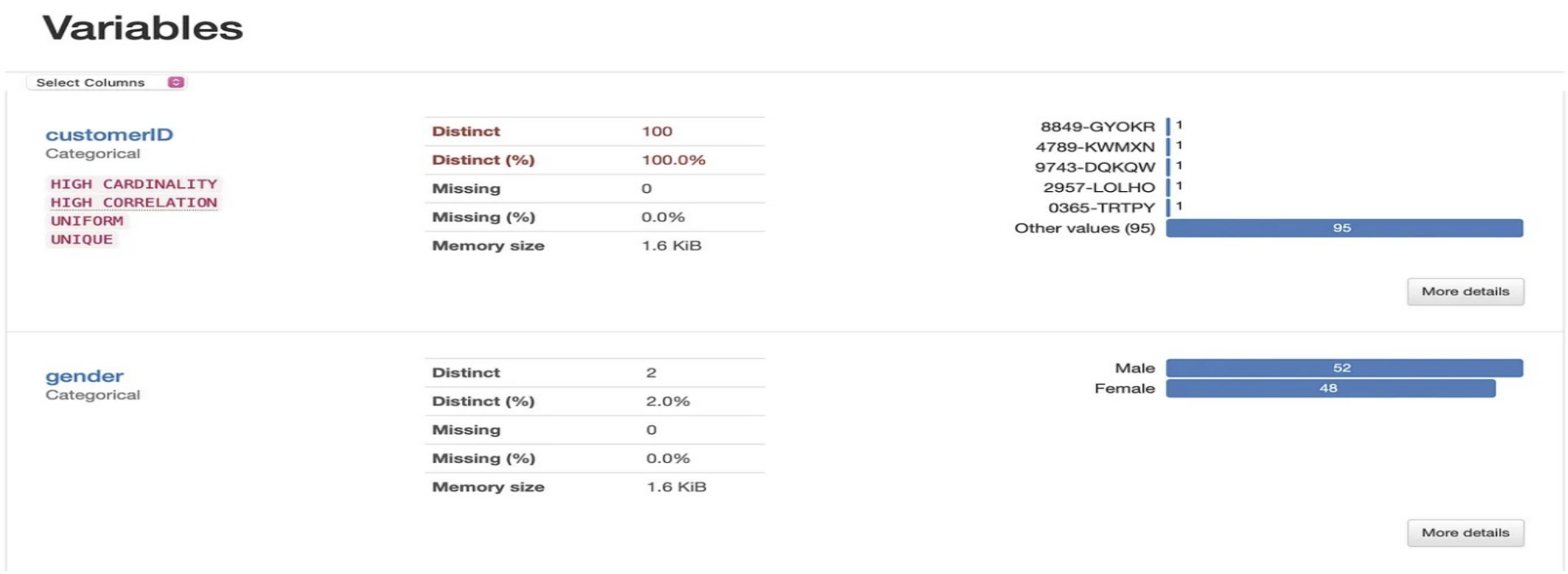
Variable types

Categorical	13
Boolean	5
Numeric	3

Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis (EDA) Using Pandas Profiling:*

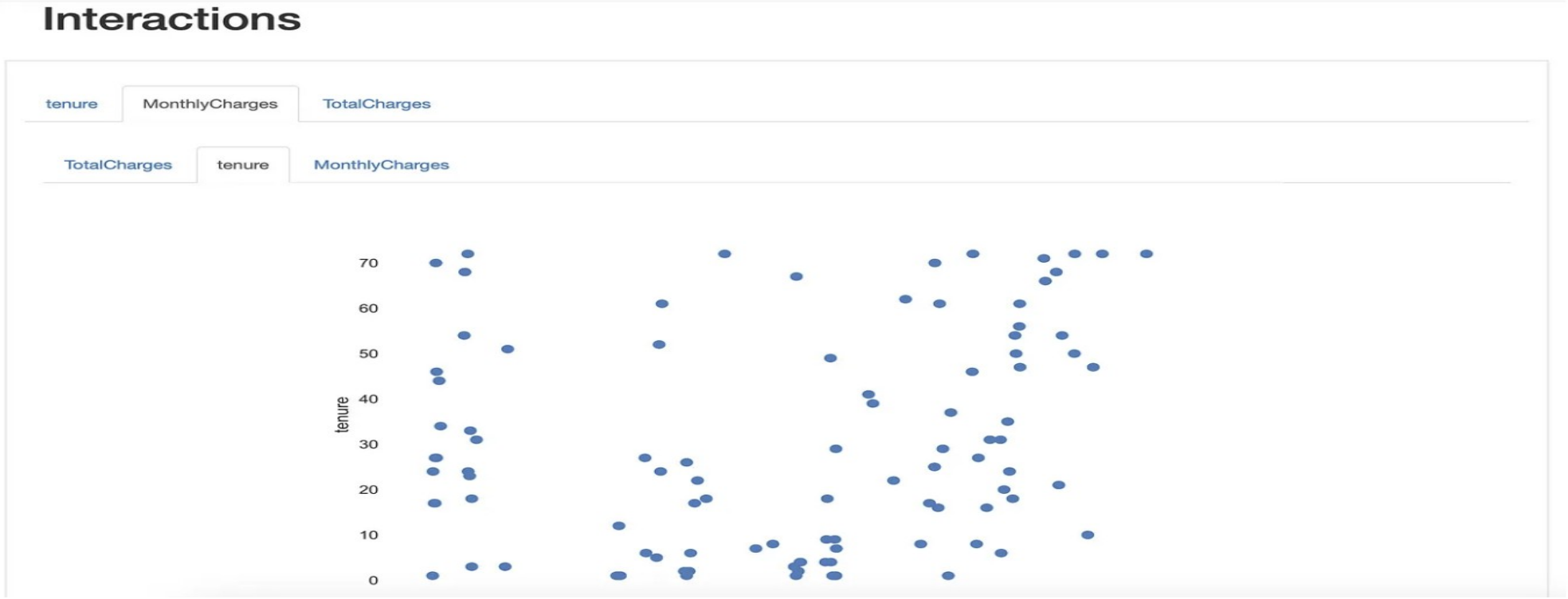
This section provides detail analysis on Variables/Columns/Features of the Dataset that totally depend on type of Variables/Columns/Features like Numeric, String , Boolean etc.



Exploratory Data Analysis(EDA) Using Pandas library

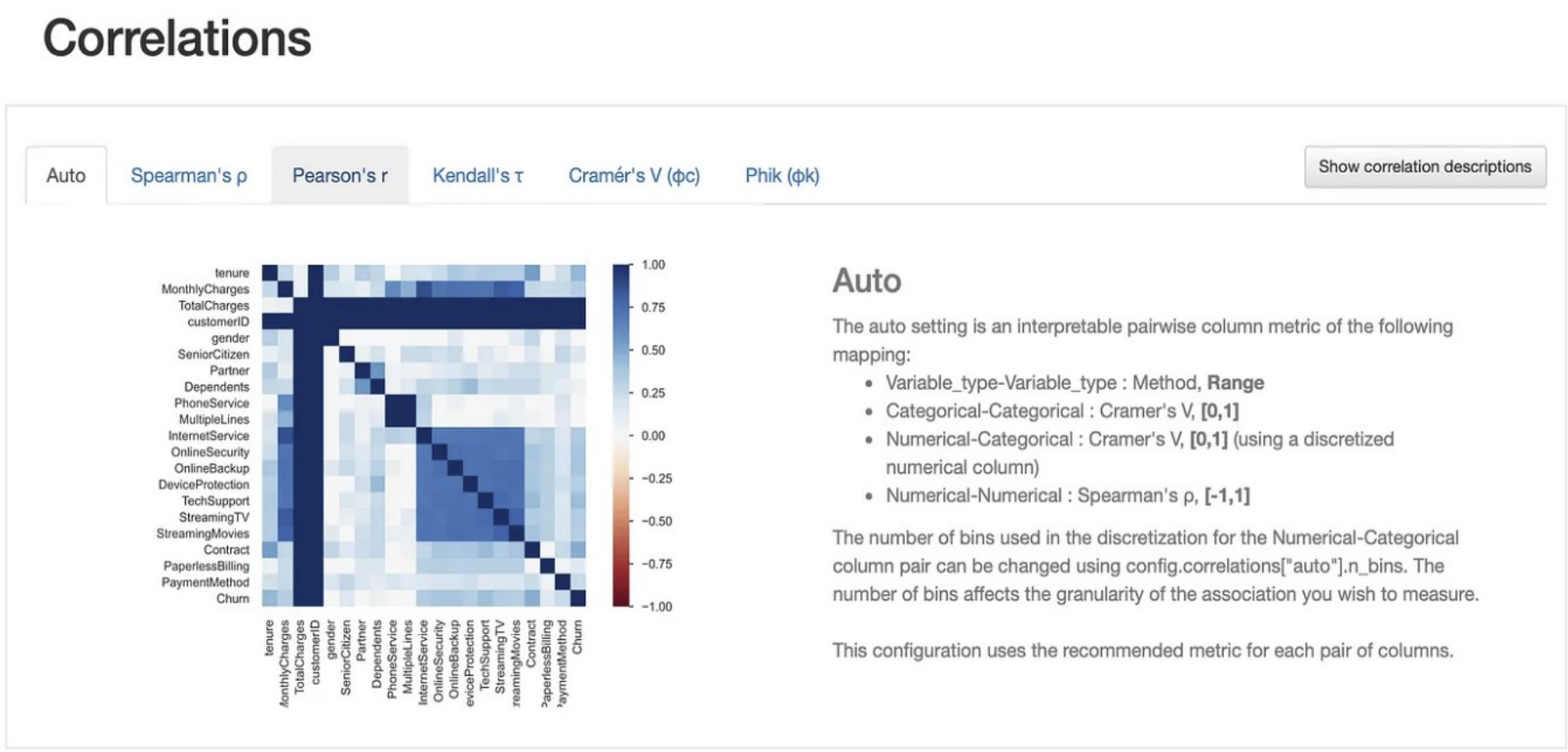
➢ *Exploratory Data Analysis (EDA) Using Pandas Profiling:*

Interaction sections gives more details with bivariate analysis/multivariate analysis.



Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis (EDA) Using Pandas Profiling:*

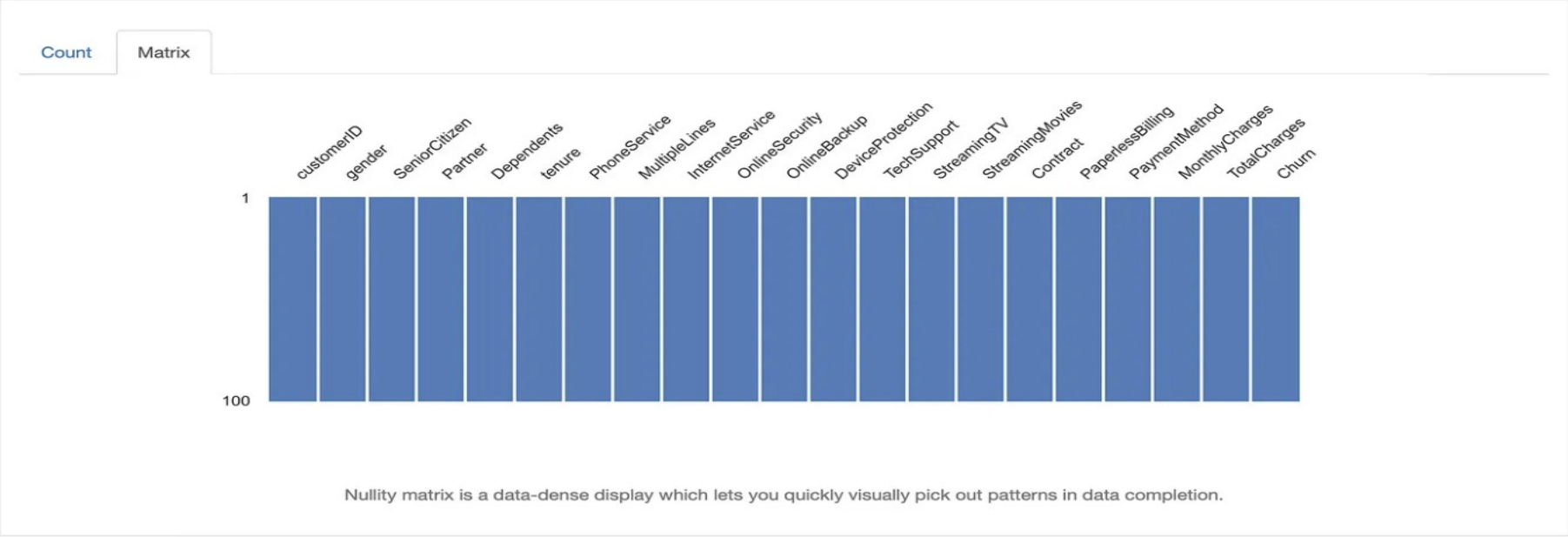


Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis (EDA) Using Pandas Profiling:*

This report also gives detail analysis of Missing values in the four types of graph Count, matrix, Heatmap and dendrogram.

Missing values



Exploratory Data Analysis(EDA) Using Pandas library

➢ *Exploratory Data Analysis (EDA) Using Pandas Profiling:*

This section displays the first and last 10 rows of the dataset.

Sample

First rows | Last rows

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
3240	8849-GYOKR	Female	0	Yes	No	54	Yes	Yes	Fiber optic	Yes	Yes
6600	4592-IWTJI	Female	0	Yes	Yes	3	No	No phone service	DSL	Yes	No
5573	5550-VFRLC	Female	1	No	No	1	Yes	No	Fiber optic	No	No
418	0578-SKVMF	Female	0	Yes	Yes	22	Yes	No	Fiber optic	No	Yes
3503	9821-BESNZ	Male	0	No	No	66	Yes	Yes	Fiber optic	No	Yes
2679	7341-LXCAF	Male	0	Yes	No	4	Yes	Yes	Fiber optic	No	No
3811	3620-MWJNE	Male	0	No	No	2	Yes	No	DSL	Yes	No
3737	8601-QACRS	Female	0	No	No	5	Yes	Yes	DSL	No	No
564	7319-VENRZ	Male	0	No	No	7	Yes	No	DSL	No	No
1870	3566-HJGPK	Male	0	No	No	1	Yes	No	DSL	No	No


THANK YOU