Infosys | Springboard

Internship Program

Skill Match Resume Matcher and Skill Recommender

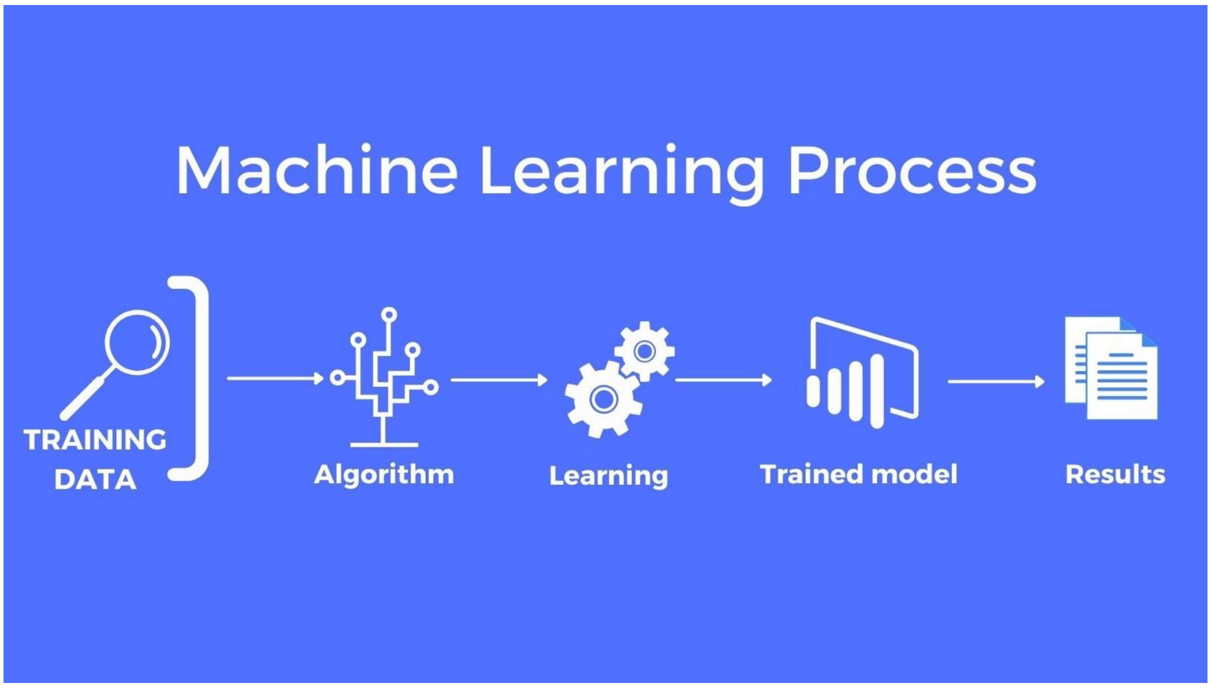
Training Program – Milestone 1

NAME: **RAVULU BINDU PRIYA**
Gmail: ravulubindhupriya@gmail.com

Introduction to Machine Learning

Machine Learning (ML) is a subset of Artificial Intelligence (AI) which provides machines the ability to learn automatically & improve from experience without being explicitly programmed.

Artificial Intelligence (AI) is a technology that enables machines and computers to perform tasks that typically require human intelligence. It helps systems learn from data, recognize patterns and make decisions to solve complex problems.



Applications:

Face recognition, virtual assistants, healthcare prediction, weather forecasting, and recommendation systems.

The Machine Learning Process

1. Define problem
2. Collect and prepare data
3. Visualize and explore data
4. Train the model
5. Evaluate and refine

Key Concepts

1. **Data Input:** Machine needs data like text, images or numbers to analyze. Good quality and enough quantity of data are important for effective learning.
2. **Algorithms**: Algorithms are mathematical methods that help the machine find patterns in data. Different algorithms help different tasks such as classification or regression.
3. **Model Training:** During training, the machine adjusts its internal settings to better predict outcomes. It learns by reducing the difference between its predictions and actual results.
5. **Feedback Loop:** Machine compares its predictions with true outcomes and uses this feedback to correct errors.
6. **Experience and Iteration:** Machine repeats training many times with data helps in refining its predictions with each pass, more data and iterations improve accuracy.
6. **Evaluation and Generalization:** Model is tested on unseen data to ensure it performs well on real-world tasks.

Tool: Anaconda Navigator

Used for managing Python environments.
Includes **Spyder** (IDE) and **Google Colab Notebook** for running ML code.

NumPy (Numerical Python)

NumPy (Numerical Python) is a powerful Python library designed for scientific computing. It provides support for creating and manipulating large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is widely used for numerical data processing, enabling tasks such as linear algebra, statistical analysis, and Fourier transforms.

Creating Arrays(examples)

```
import numpy as np

n1 = np.array([10, 20, 30, 40])

print(n1)
```

Output:

```
[10 20 30 40]
```

Multi-Dimensional Array

```
n2 = np.array([[10, 20, 30, 40], [40, 30, 20, 10]])
print(n2)
```

Output:

```
[[10 20 30 40]
 [40 30 20 10]]
```

#Initializing Numpy Arrays Using arange() and full()

```
1.n1 = np.arange(10, 20)
print(n1)
```

Output: `[10 11 12 13 14 15 16 17 18 19]`

```
2.n1 = np.full((2, 2), 10)
print(n1)
```

Output:

```
[[10 10]
 [10 10]]
```

```
3. n1=np.arange(10,50,5)
print(n1)
```

Output:

```
[10 15 20 25 30 35 40 45]
```

```
4.n2=np.array([[10,20,30,40],[40,30,20,10]])
print(n2)
```

Output:

```
[[10 20 30 40]
[40 30 20 10]]
```

Random Integers

```
np.random.randint(1, 100, 5)
```

Output (example):

```
[95 88 26 22 76]
```

Joining Arrays

```
1.import numpy as np
n1=np.array([1,2,3,4])
n2=np.array([4,3,2,1])
np.vstack((n1,n2))
```

Output:

```
array([[1, 2, 3, 4],
[4, 3, 2, 1]])
```

```
2.import numpy as np
n1=np.array([1,2,3,4])
n2=np.array([4,3,2,1])
np.column_stack((n1,n2))
```

Output:

```
array([[1, 4],
[2, 3],
[3, 2],
[4, 1]])
```

```
3.import numpy as np
n1=np.array([1,2,3,4])
n2=np.array([4,3,2,1])
np.sum((n1,n2))
```

Output: np.int64(20)

Matrix Multiplication

```
n1 = np.array([[1, 2, 3], [4, 5, 6]])

n2 = np.array([[1, 2], [3, 4], [5, 6]])

np.dot(n1, n2)
```

Output:

```
[[22 28]
 [49 64]]
```

Pandas – Data Analysis Library

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

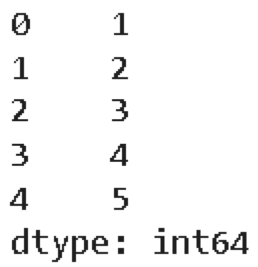
Elaborate:

A Series is a one-dimensional labeled array (like a single column), while a DataFrame is a two-dimensional labeled data structure (like a spreadsheet).

Creating Series

```
1.import pandas as pd

s1 = pd.Series([1, 2, 3, 4, 5])

print(s1)
```

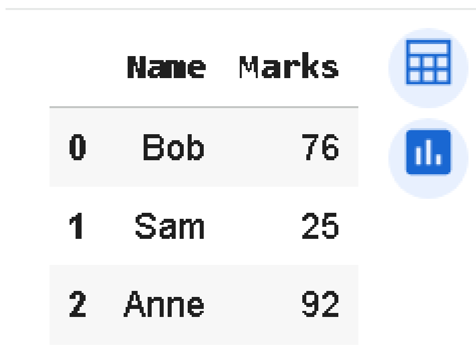
Output:



```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

```
2. import pandas as pd

pd.DataFrame({"Name":['Bob','Sam','Anne'],"Marks":[76,25,92]})
```

Output:

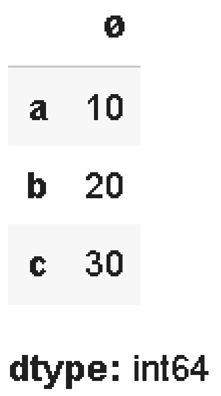


```
   Name  Marks
0  Bob   76
1  Sam   25
2  Anne  92
```

3. import pandas as pd

pd.Series({"a":10,"b":20,"c":30})

Output::

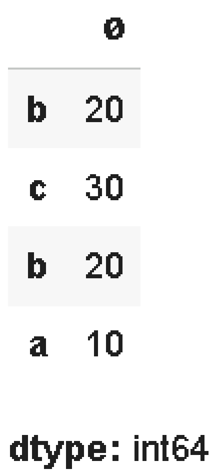


```
   0
a  10
b  20
c  30

dtype: int64
```

4. import pandas as pd

pd.Series({"a":10,"b":20,"c":30},index=["b","c","b","a"])

Output:



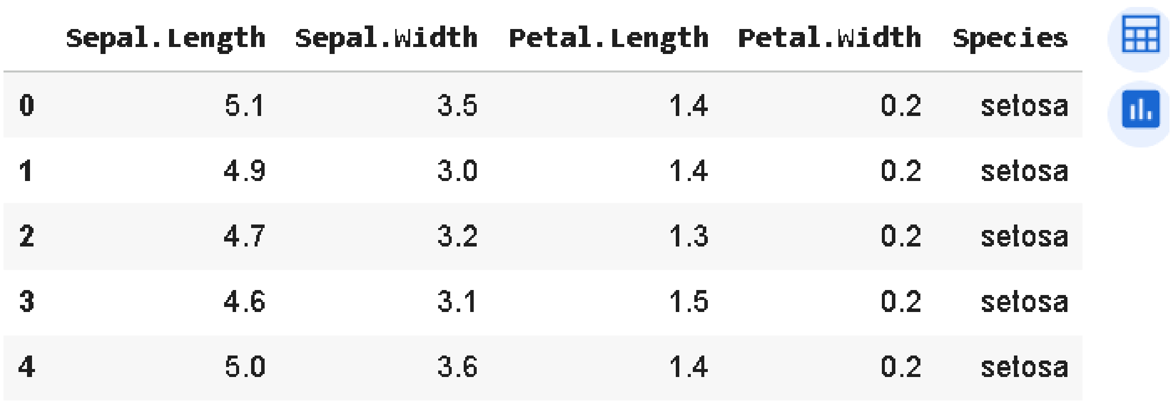
```
   0
b  20
c  30
b  20
a  10

dtype: int64
```

DataFrame from CSV

iris = pd.read_csv("iris.csv")

1.iris.head()

Output:



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

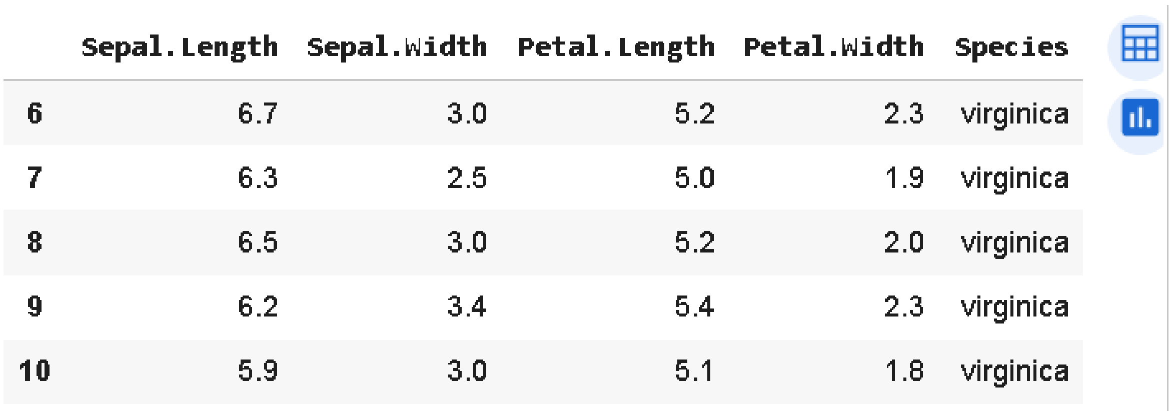
---

2. ds.shape

Output:: (11,5)

3. ds.tail()

Output:



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6	6.7	3.0	5.2	2.3	virginica
7	6.3	2.5	5.0	1.9	virginica
8	6.5	3.0	5.2	2.0	virginica
9	6.2	3.4	5.4	2.3	virginica
10	5.9	3.0	5.1	1.8	virginica

4.ds.describe()

Output:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	11.000000	11.000000	11.000000	11.000000
mean	5.572727	3.200000	3.145455	1.063636
std	0.768233	0.379473	1.952621	0.966719
min	4.600000	2.500000	1.300000	0.200000
25%	4.950000	3.000000	1.400000	0.200000
50%	5.400000	3.100000	1.700000	0.400000
75%	6.250000	3.450000	5.150000	1.950000
max	6.700000	3.900000	5.400000	2.300000

5.print(ds.iloc[0:3 ,0:2])

Output:

```
   Sepal.Length  Sepal.Width
0           5.1          3.5
1           4.9          3.0
2           4.7          3.2
```

6.print(ds.loc[0:4,("Sepal.Length","Petal.Length")])

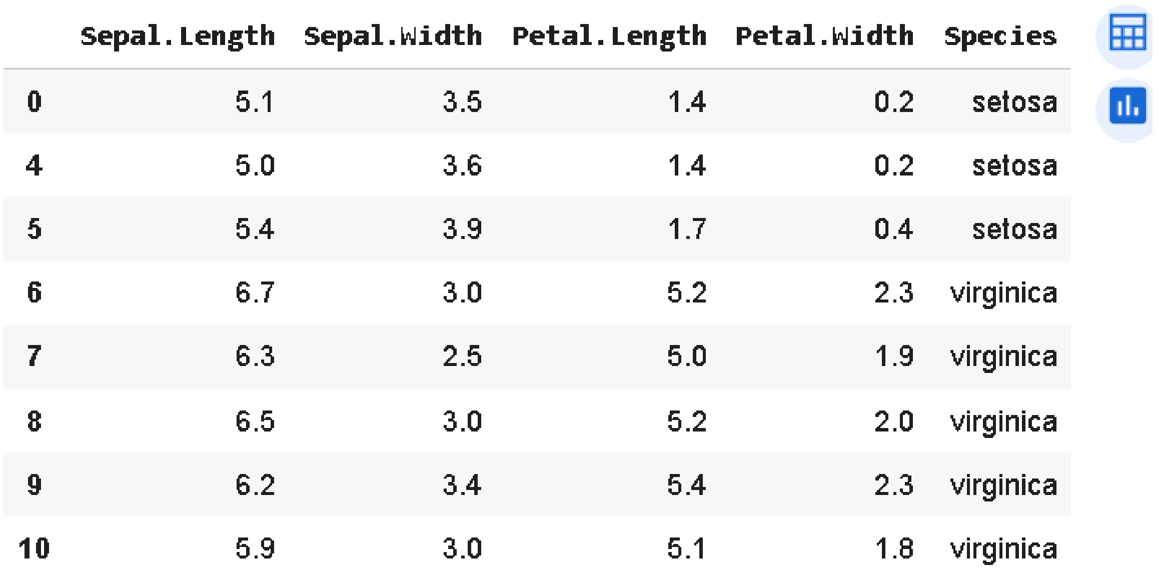
Output:

```
   Sepal.Length  Petal.Length
0           5.1           1.4
1           4.9           1.4
2           4.7           1.3
3           4.6           1.5
4           5.0           1.4
```

7.ds.drop('Sepal.Length',axis=1)

ds.drop([1,2,3], axis=0)

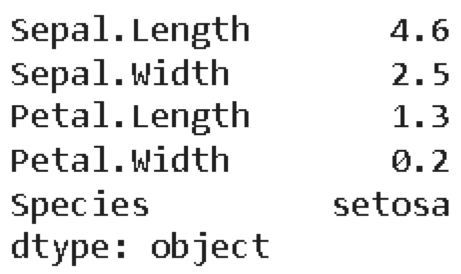
Output:



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	6.7	3.0	5.2	2.3	virginica
7	6.3	2.5	5.0	1.9	virginica
8	6.5	3.0	5.2	2.0	virginica
9	6.2	3.4	5.4	2.3	virginica
10	5.9	3.0	5.1	1.8	virginica

9.print(ds.min())

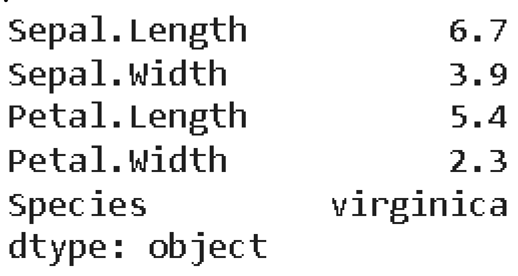
Output:



```
Sepal.Length       4.6
Sepal.Width        2.5
Petal.Length       1.3
Petal.Width        0.2
Species         setosa
dtype: object
```

10.print(ds.max())

Output:



```
Sepal.Length          6.7
Sepal.Width           3.9
Petal.Length          5.4
Petal.Width           2.3
Species         virginica
dtype: object
```

11.print(ds.mean(numeric_only=True))

Output:

```
Sepal.Length    5.572727
Sepal.Width     3.200000
Petal.Length    3.145455
Petal.Width     1.063636
dtype: float64
```

12.print(ds.median(numeric_only=True))

Output:

```
Sepal.Length    5.572727
Sepal.Width     3.200000
Petal.Length    3.145455
Petal.Width     1.063636
dtype: float64
```

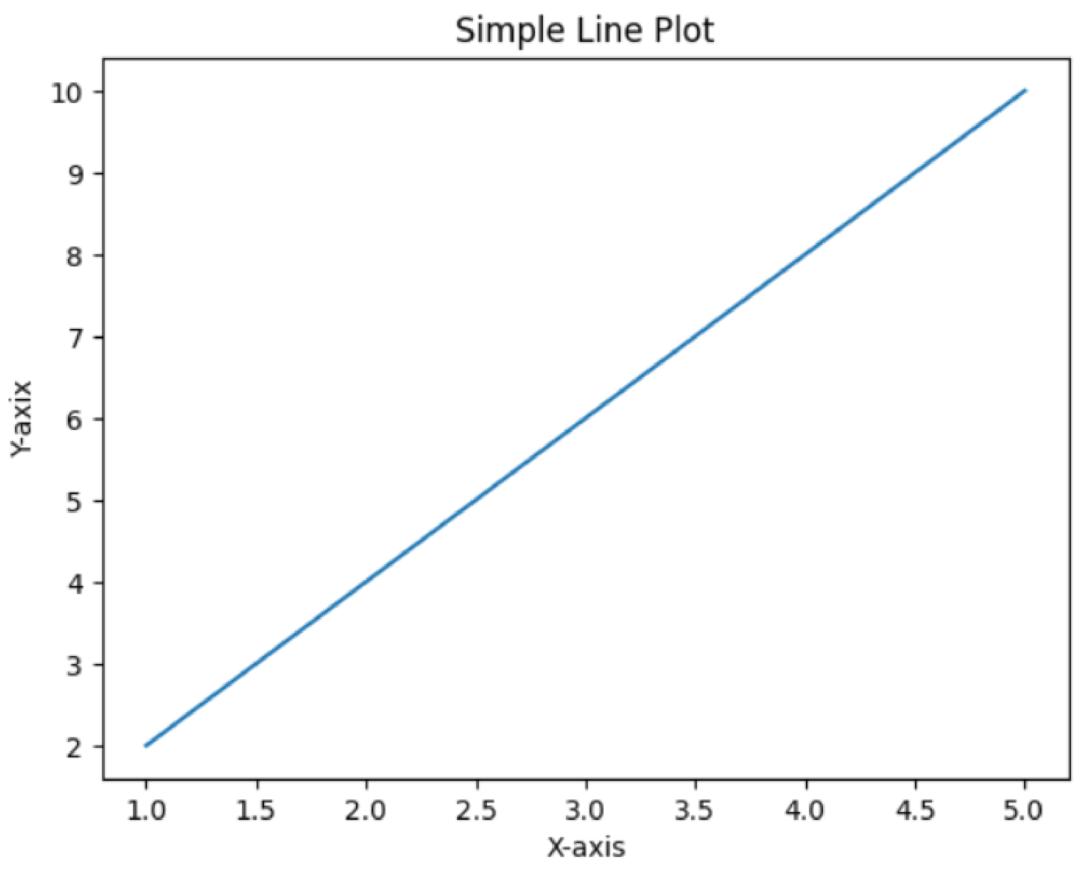
Data Visualization using Matplotlib

Line Plot

A line plot displays data trends over continuous intervals. It connects data points with straight lines, making it ideal for showing progression, growth, or time-based trends.

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

plt.plot(x, y)

plt.title("Simple Line Plot")

plt.xlabel(“X-axis”);

plt.xlabel(“Y-axis”);

plt.show()
```

Output: Line graph showing linear increase.



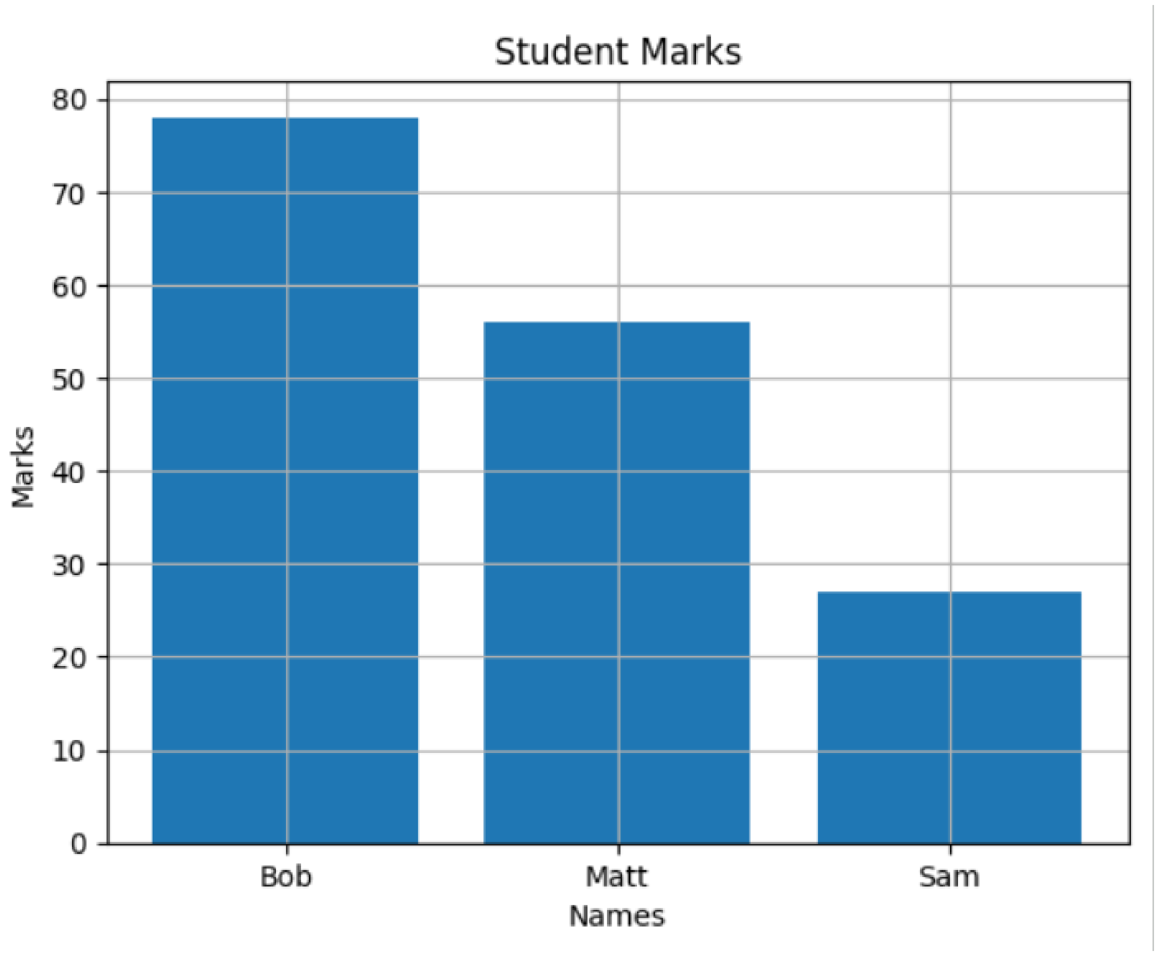
Bar Plot

Bar plots represent categorical data with rectangular bars. The bar length is proportional to the category value. It’s useful for comparing multiple groups or variables.

```
student = {"Bob": 78, "Matt": 56, "Sam": 27}

plt.bar(student.keys(), student.values())

plt.bar(names, values)

plt.title("Student Marks")

plt.xlabel("Names")

plt.ylabel("Marks")

plt.grid(True)

plt.show()
```

Output:

Vertical bars showing student marks.



Histogram

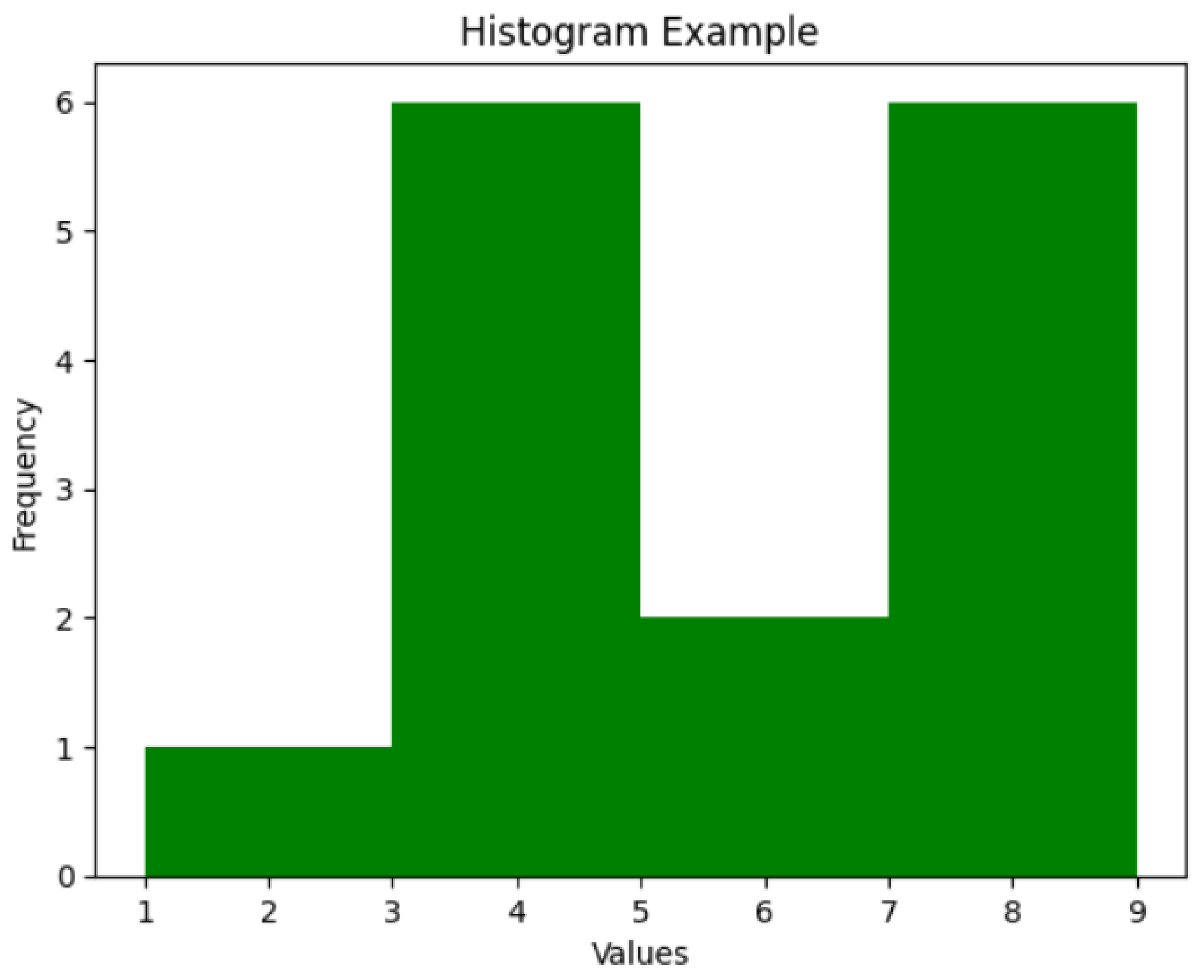
A histogram shows the frequency distribution of numerical data by dividing it into bins. It helps visualize data spread, skewness, and central tendency.

```
data = [1,3,3,3,3,9,9,5,4,4,8,8,8,6,7]

plt.hist(data,color='g',bins=4)

plt.title("Histogram Example")

plt.xlabel("Values")

plt.ylabel("Frequency")
```

plt.show()

Output:

Green histogram showing frequency distribution.



Box Plot

A box plot displays the spread and skewness of data through quartiles. It highlights the median, interquartile range, and outliers, helping identify data variability and anomalies.

import matplotlib.pyplot as plt

one=[1,2,3,4,5,6,7,8,9]

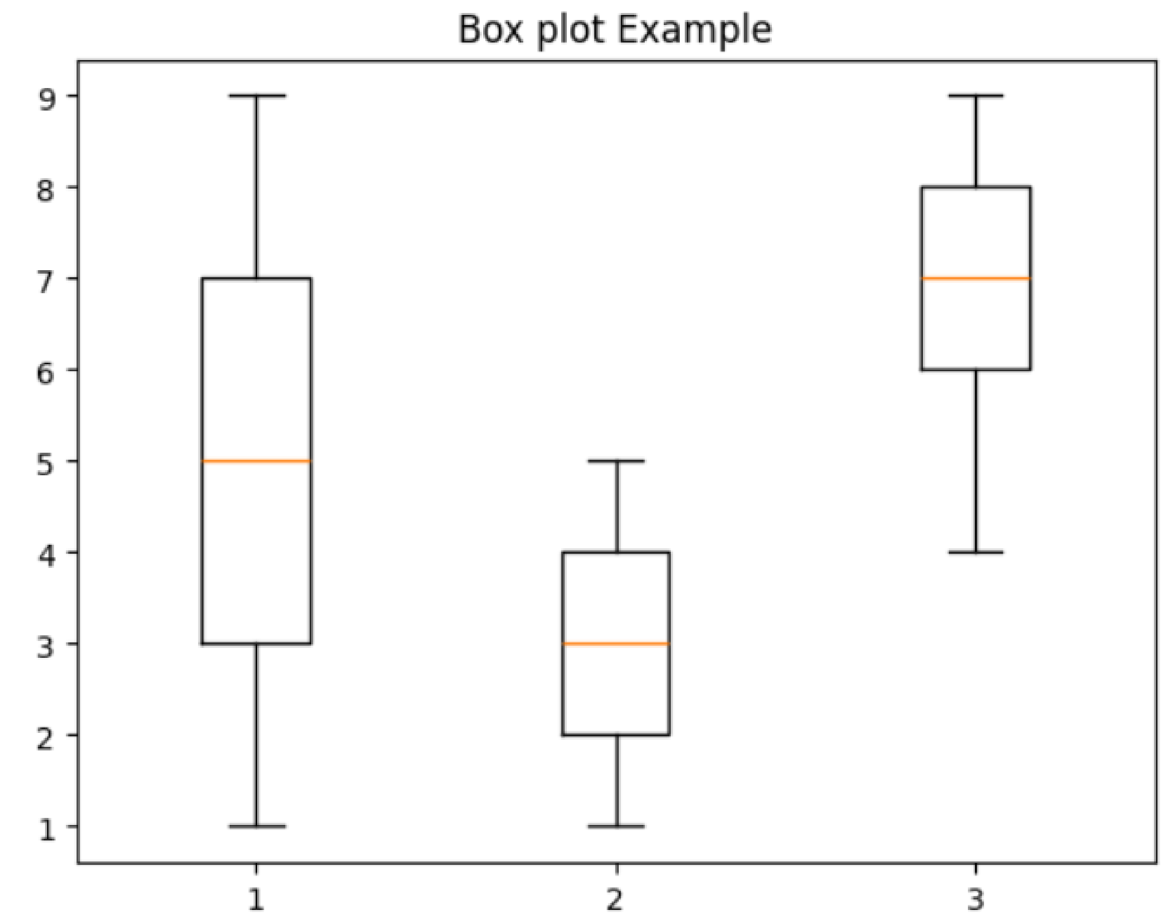
two=[1,2,3,4,5,4,3,2,1]

three=[6,7,8,9,8,7,6,5,4]

```
data=list([one,two,three])

plt.title("Box plot Example")

plt.boxplot(data)

plt.show()
```

Output:

Box plot comparing three datasets.



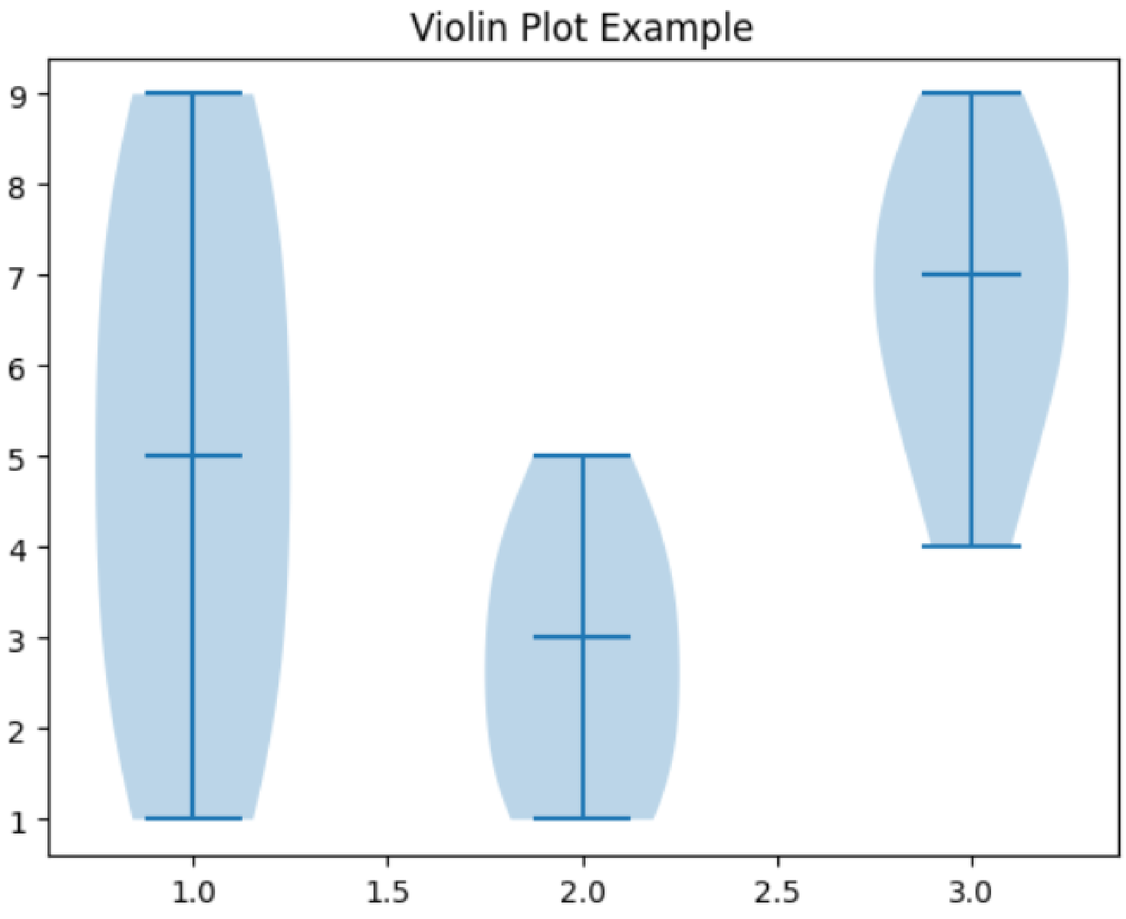
Vilolin Plot

The violin plot combines features of boxplot and density plot. It shows the distribution of data and its probability density across categories, giving a deeper insight into variations.

```
 import matplotlib.pyplot as plt

one=[1,2,3,4,5,6,7,8,9]

two=[1,2,3,4,5,4,3,2,1]
```

```
three=[6,7,8,9,8,7,6,5,4]

data=list([one,two,three])

plt.title("Box plot Example")

plt.boxplot(data)

plt.show()
```

Output: Violin Plot Example



Pie Chart

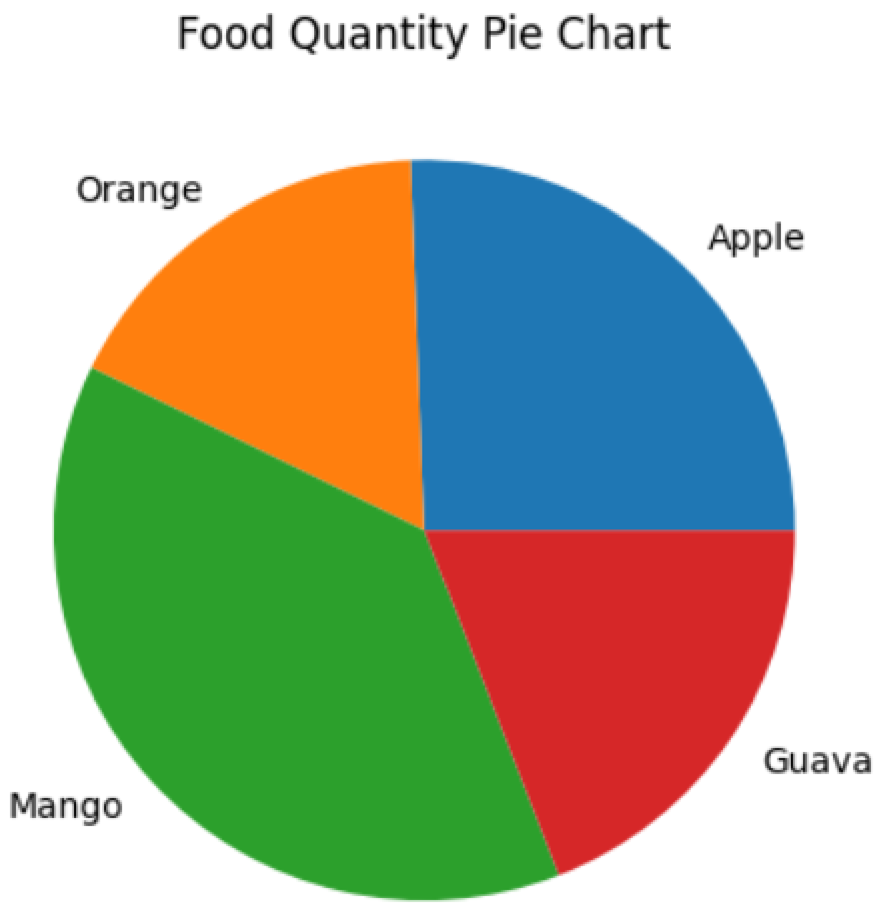
A pie chart visualizes the proportion of different categories as slices of a circle. Each slice's size corresponds to the category's percentage share of the total.

```
fruit = ['Apple', 'Orange', 'Mango', 'Guava']

quantity = [67, 34, 100, 29]
```

```
plt.pie(quantity, labels=fruit, autopct='%0.1f%%')
plt.title("Food Quantity Pie Chart")
plt.show()
```

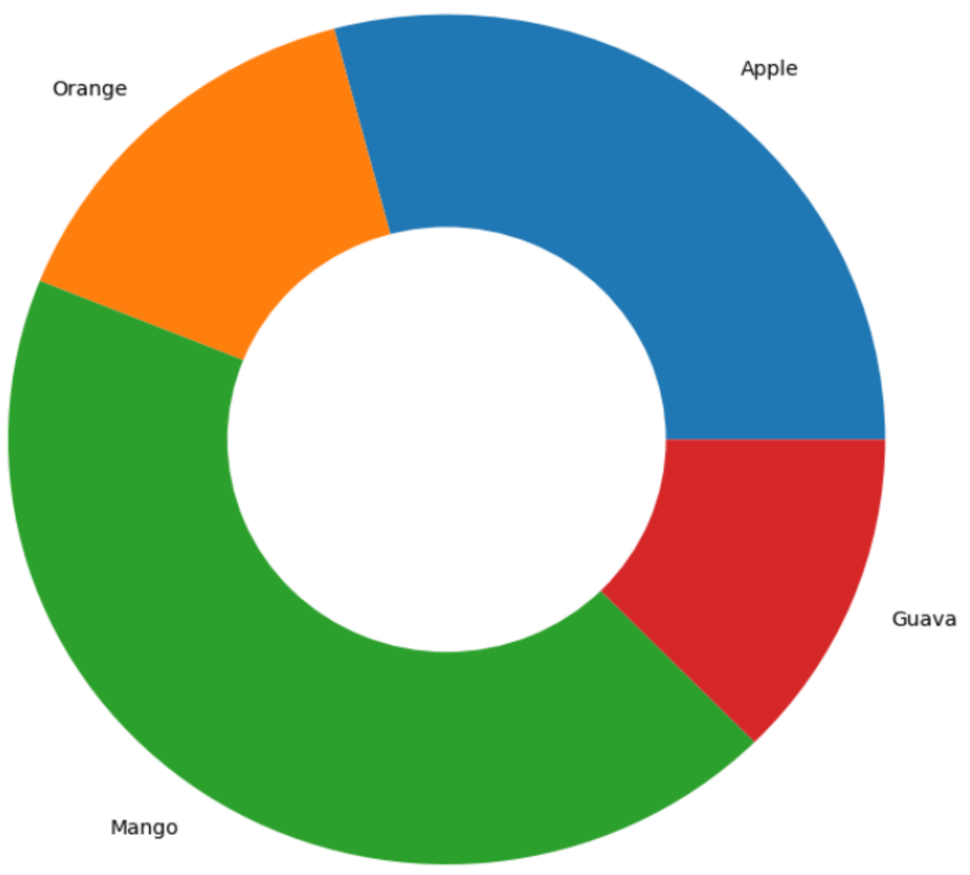
Output:

Pie chart showing fruit distribution.



```
2. fruit=['Apple','Orange','Mango','Guava']
quantity=[67,34,100,29]
plt.pie(quantity,labels=fruit,radius=2)
plt.pie([1],colors=['w'],radius=1)
plt.show()
```

Output:



Data Visualization using Seaborn

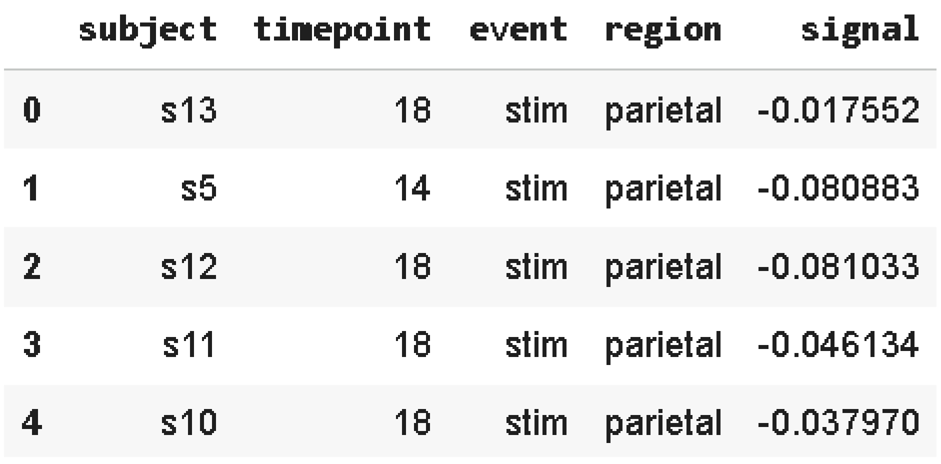
Seaborn is a powerful Python library built on Matplotlib, designed to simplify the creation of attractive and informative statistical visualizations. It integrates seamlessly with Pandas, making it easy to visualize data directly from DataFrames.

1.Frmi Dataset

```
import seaborn as sns

from matplotlib import pyplot as plt

fmri=sns.load_dataset("fmri")

fmri.head()
```

Output:

Table with columns — total_bill, tip, sex, smoker, day, time, size.

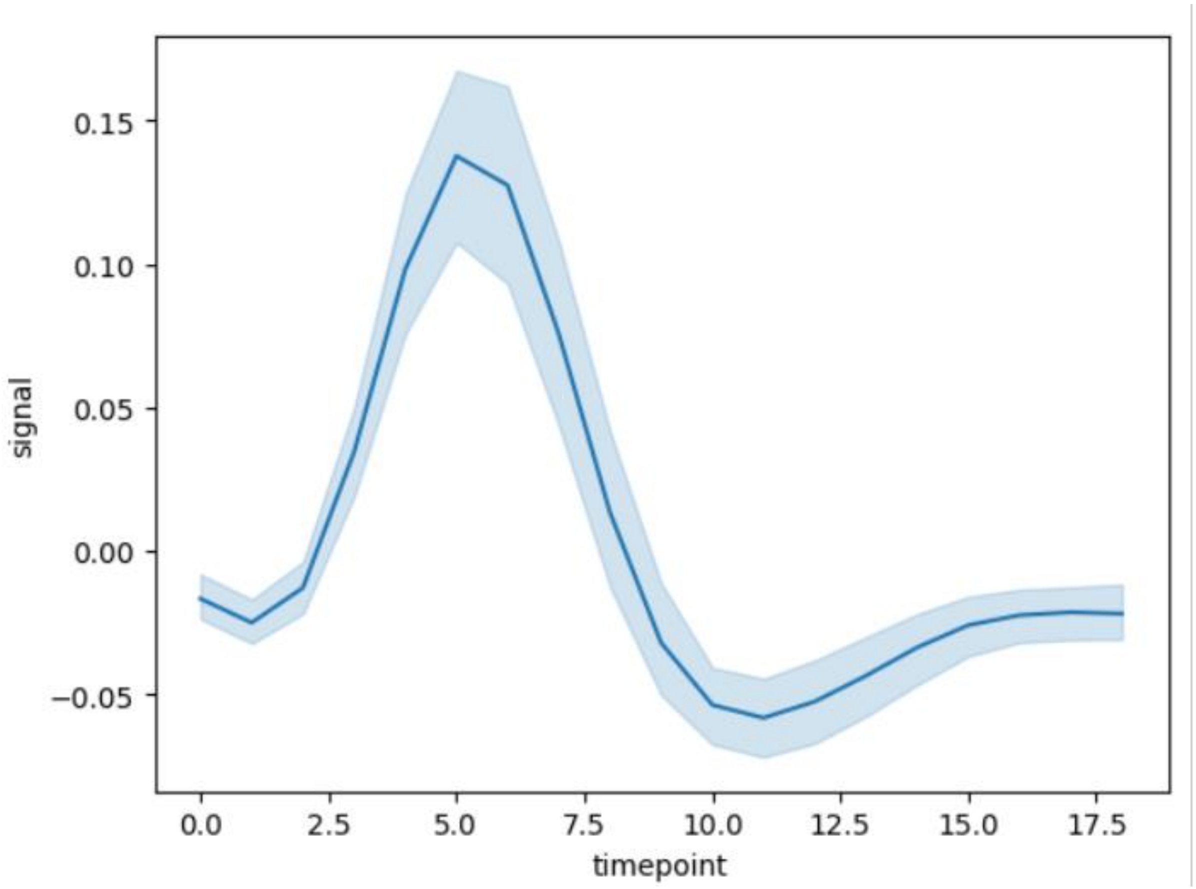


	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

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

plt.show()

Output:



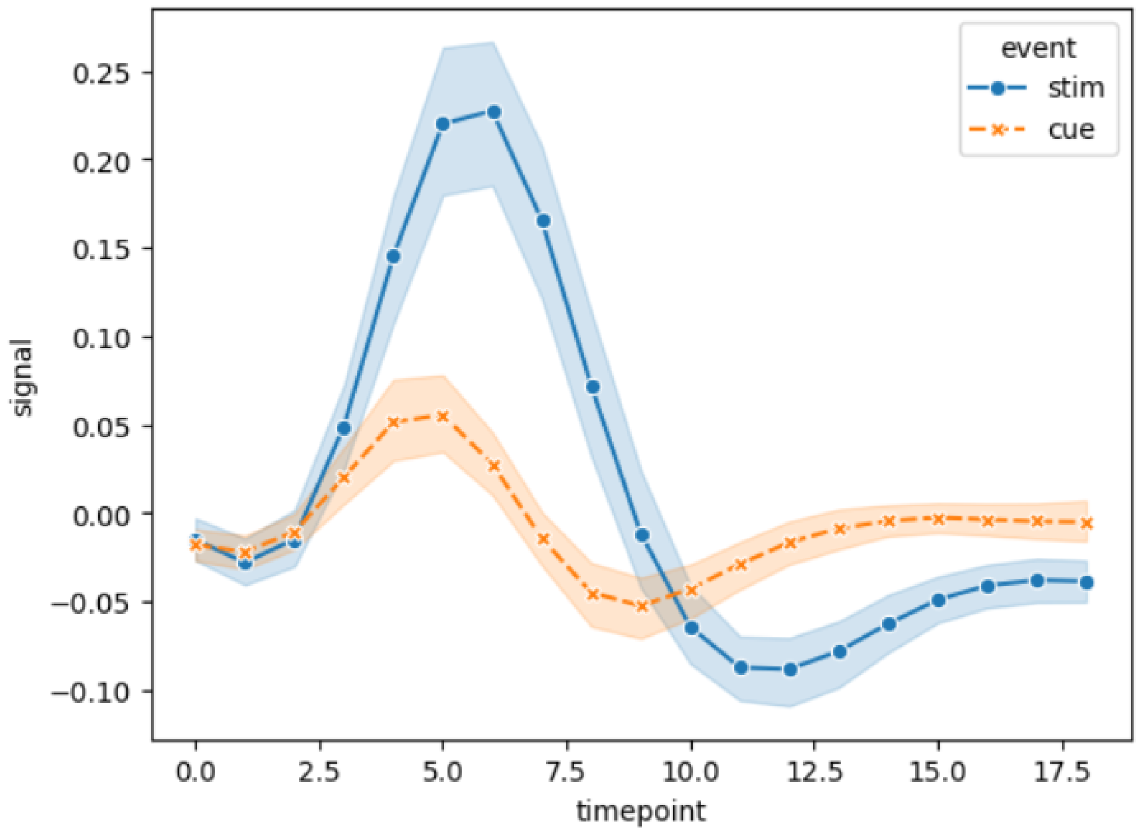
sns.lineplot(x="timepoint",y="signal",

hue="event",style="event",

markers=True,data=fmri)

Output:

```
<Axes: xlabel='timepoint', ylabel='signal'>
```



2.Pokemon dataset

import pandas as pd

import seaborn as sns

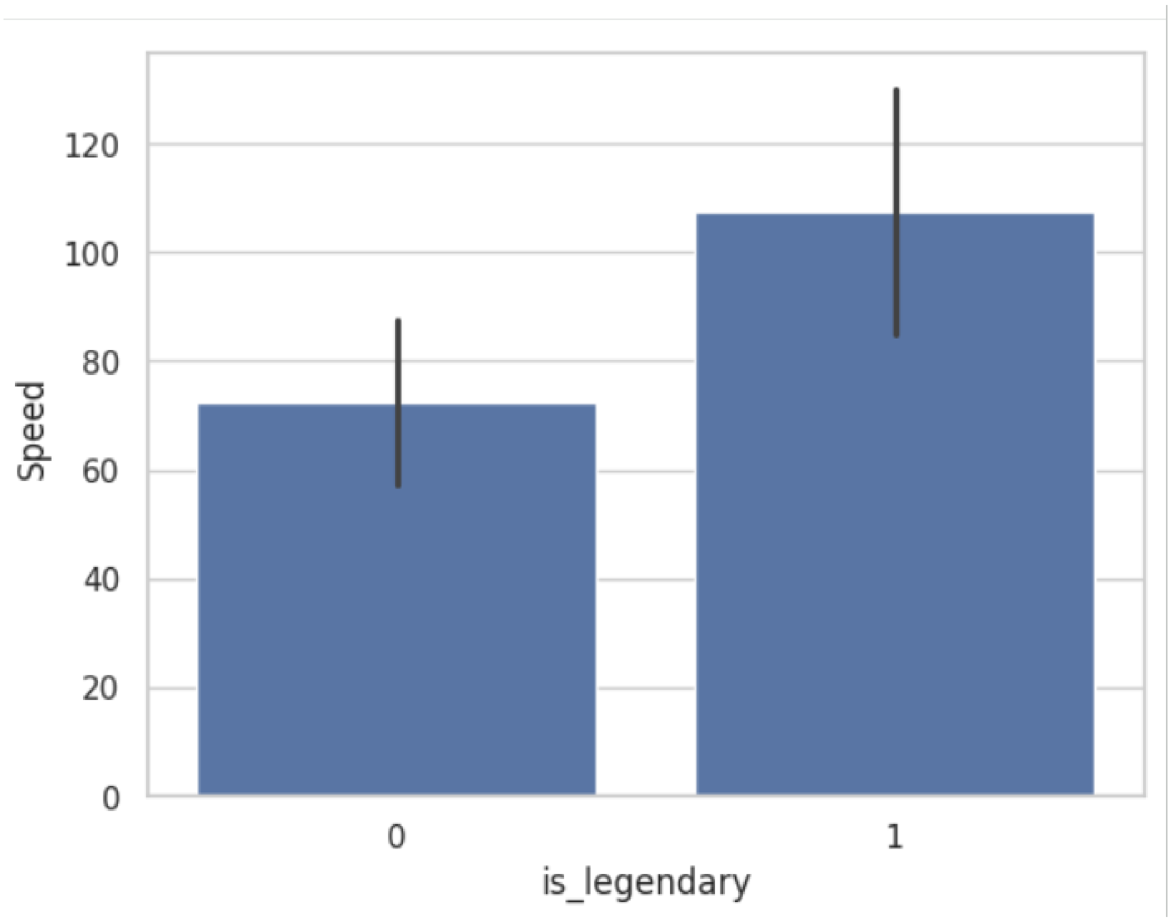
import matplotlib.pyplot as plt

sns.set(style="whitegrid")

pokemon=pd.read_csv("pokemon.csv")sns.barplot(x="is_legendary", y="Speed", data=pokemon)

-plt.show()

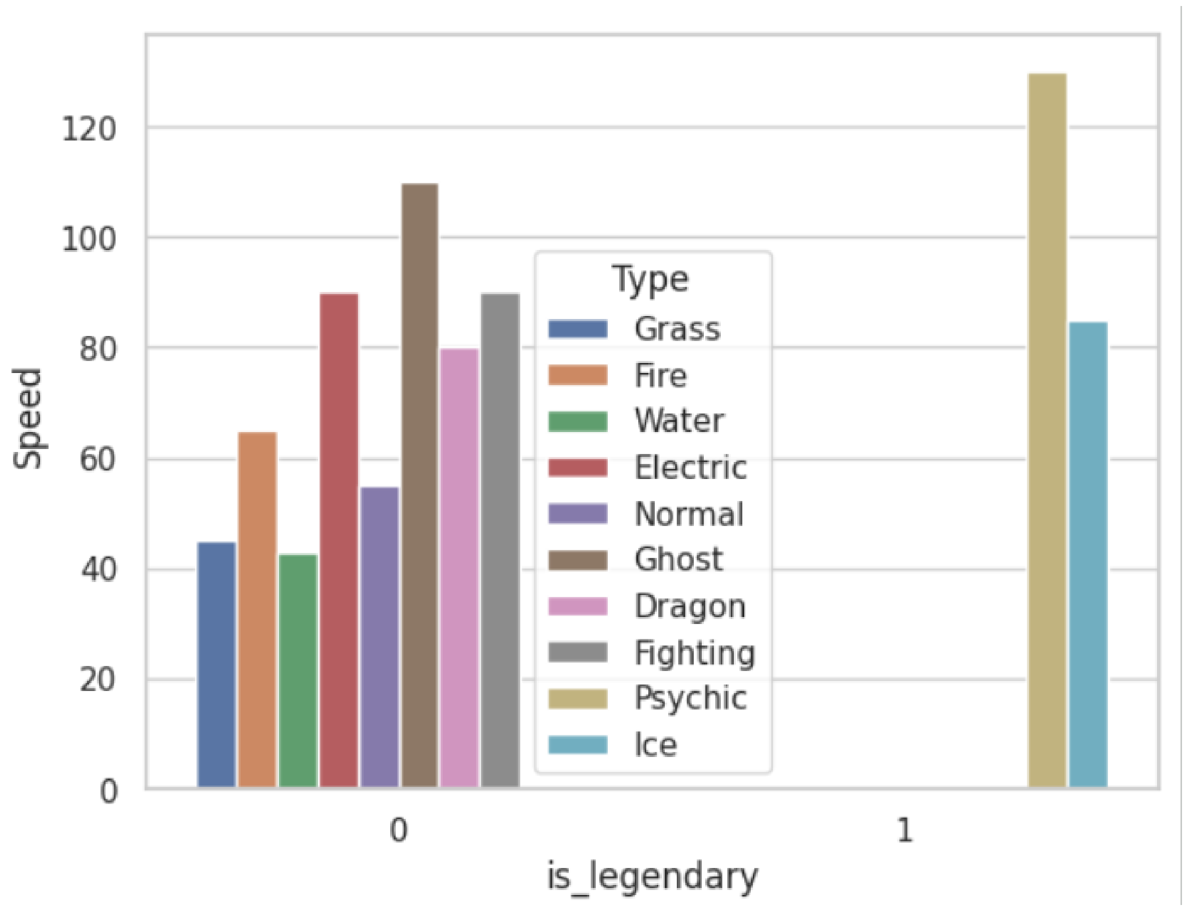
Output:



sns.barplot(x="is_legendary",y="Speed", hue="Type", data=pokemon)

plt.show()

Output:



Relationship Plots

ds=sns.load_dataset('tips')

ds.head()

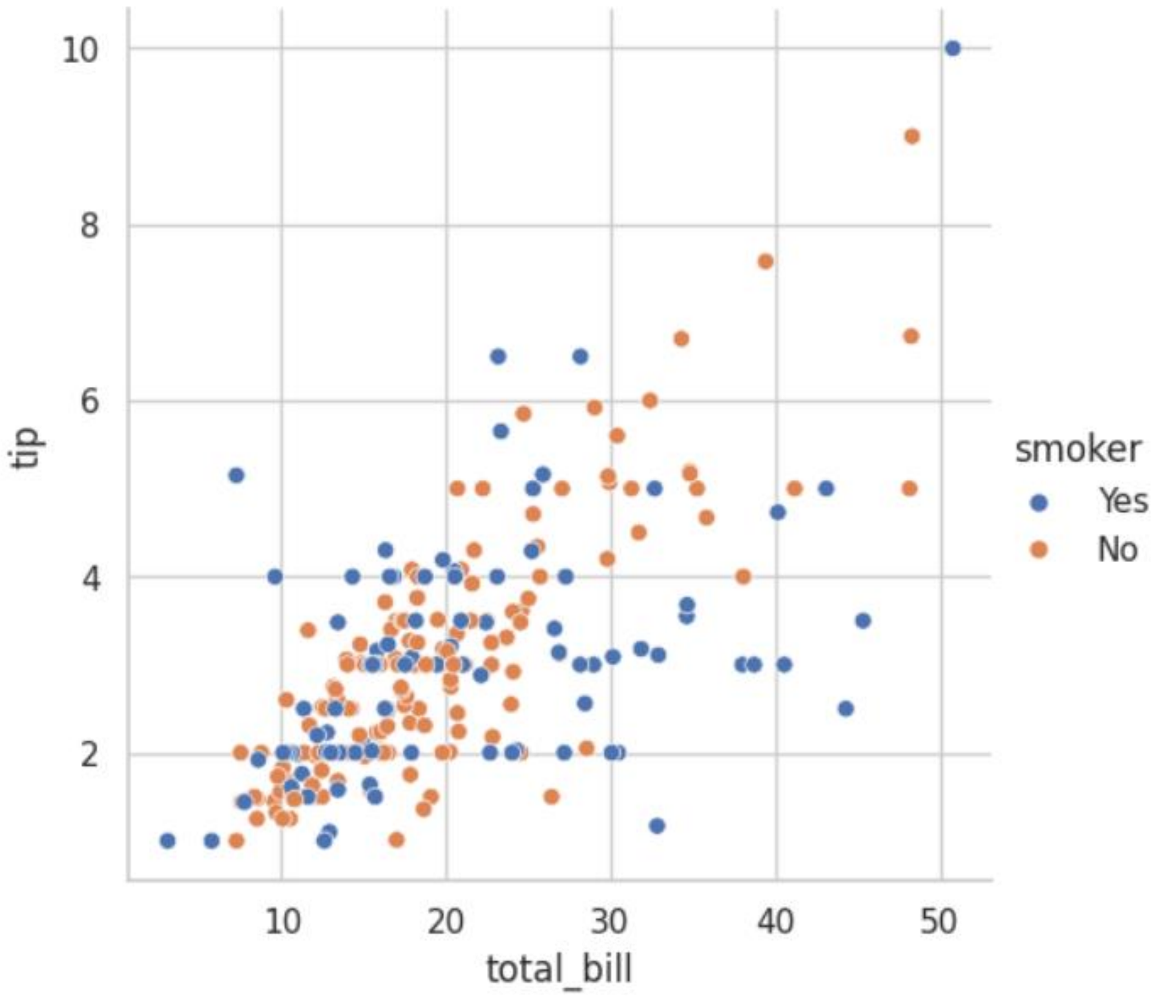
Output:

```
   total_bill   tip     sex  smoker  day    time  size
0       16.99  1.01  Female      No  Sun  Dinner     2
1       10.34  1.66    Male      No  Sun  Dinner     3
2       21.01  3.50    Male      No  Sun  Dinner     3
3       23.68  3.31    Male      No  Sun  Dinner     2
4       24.59  3.61  Female      No  Sun  Dinner     4
```

```
sns.relplot(data=ds, x='total_bill', y='tip', hue='smoker')
plt.show()
```

Output: Scatter plot colored by smoker status.

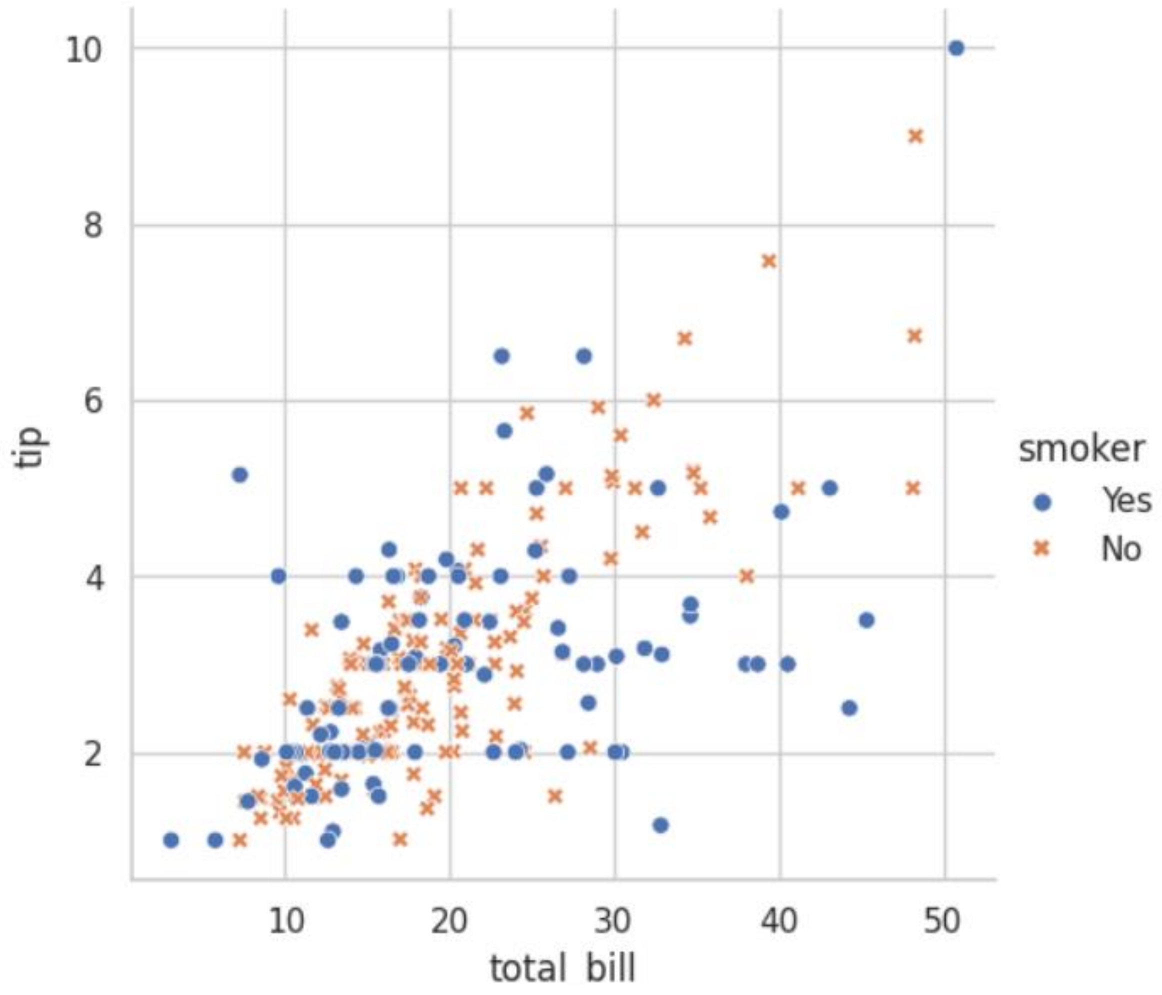
```
<seaborn.axisgrid.FacetGrid at 0x7f52c776f7a0>
```



```
sns.relplot(data=ds, x='total_bill', y='tip', hue='smoker', style='time')
plt.show()
```

Output: Scatter plot showing time (Lunch/Dinner) style markers.

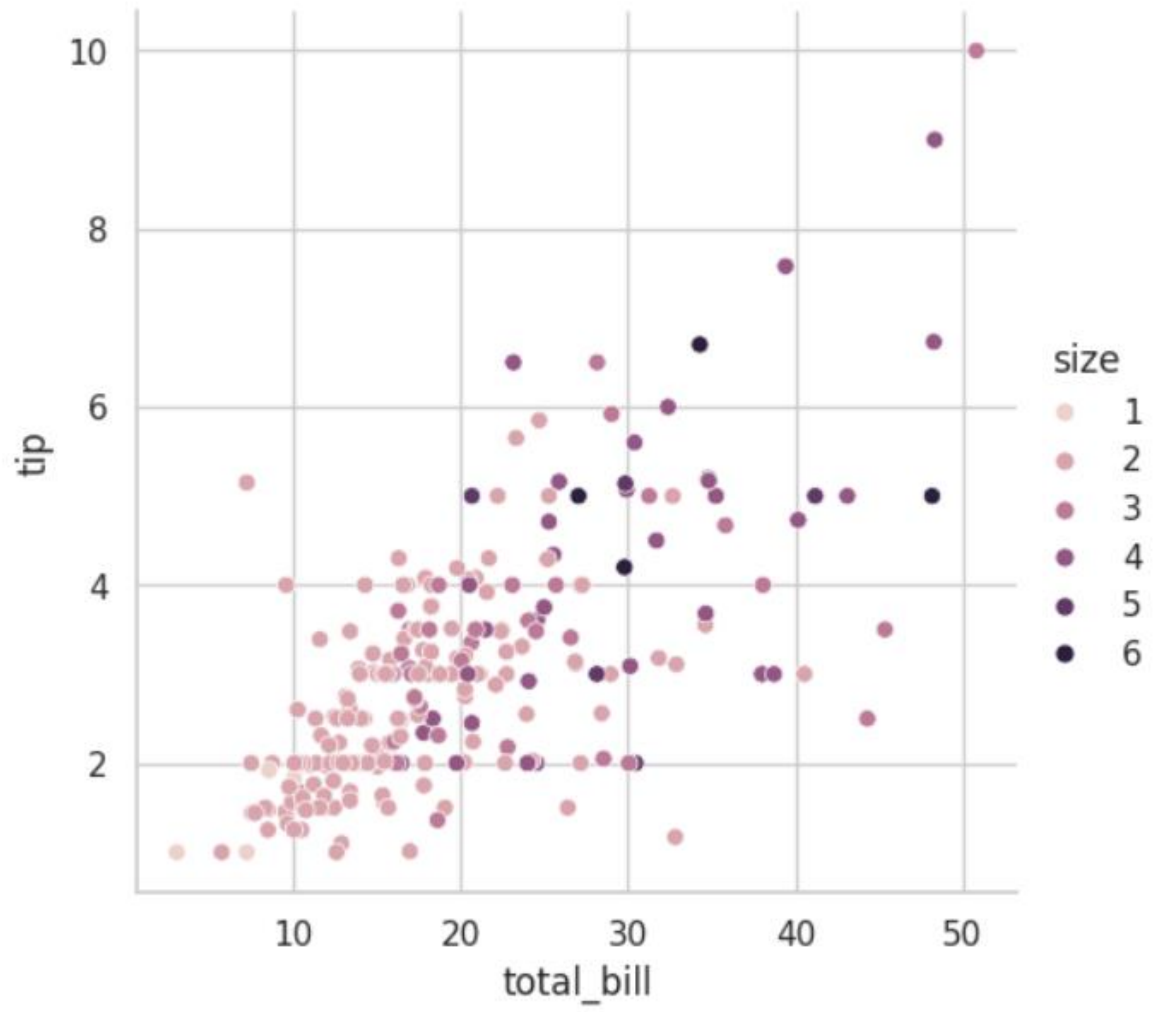
```
<seaborn.axisgrid.FacetGrid at 0x7f52c7a9f7a0>
```



```
sns.relplot(data=ds, x="total_bill", y="tip", hue="size")
plt.show()
```

Output: Color-coded plot by group size.

```
<seaborn.axisgrid.FacetGrid at 0x7f52c4cdf8f0>
```

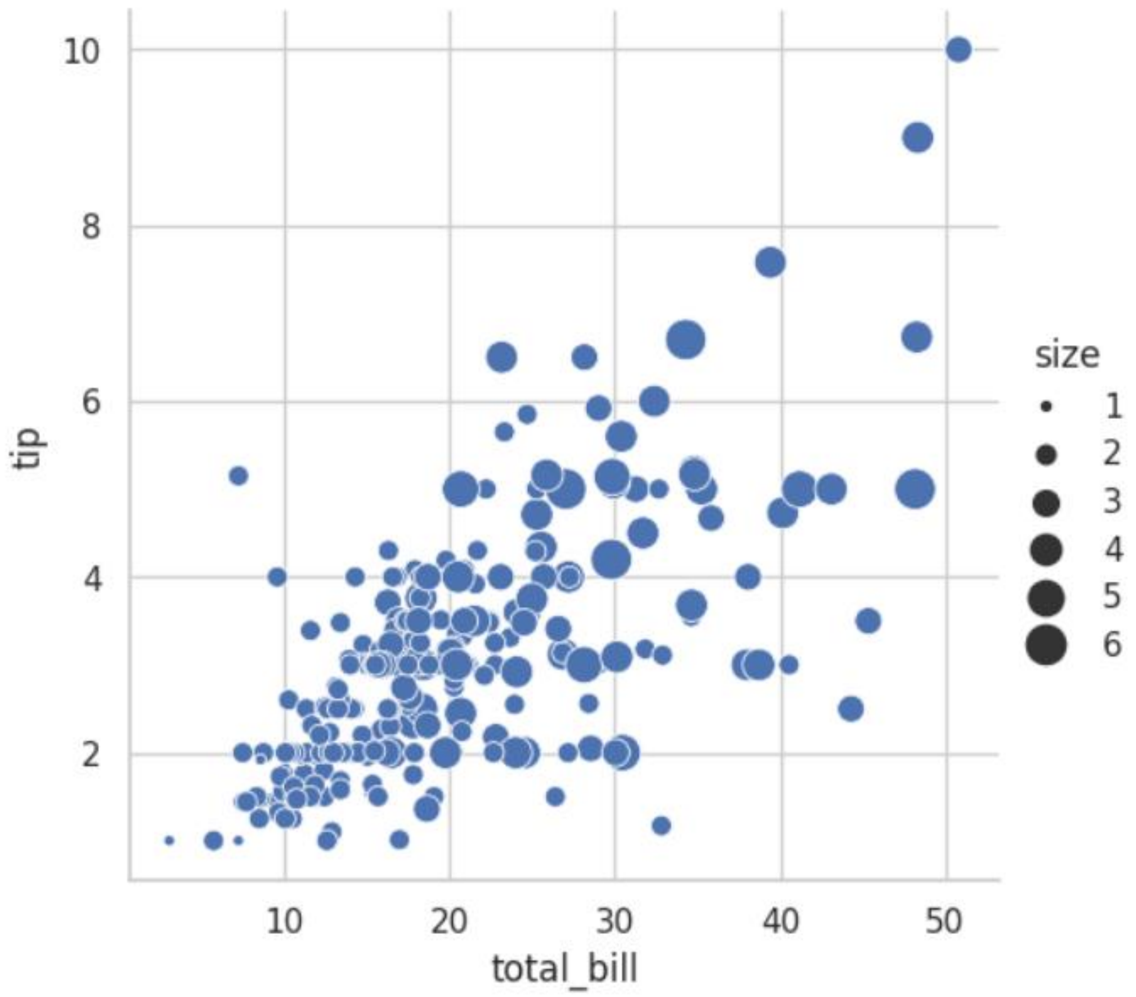


sns.relplot(data=ds, x="total_bill", y="tip", size="size", sizes=(15,200))

plt.show()

Output: **Bubble chart sized by group size.**

```
<seaborn.axisgrid.FacetGrid at 0x7f52c50ff950>
```

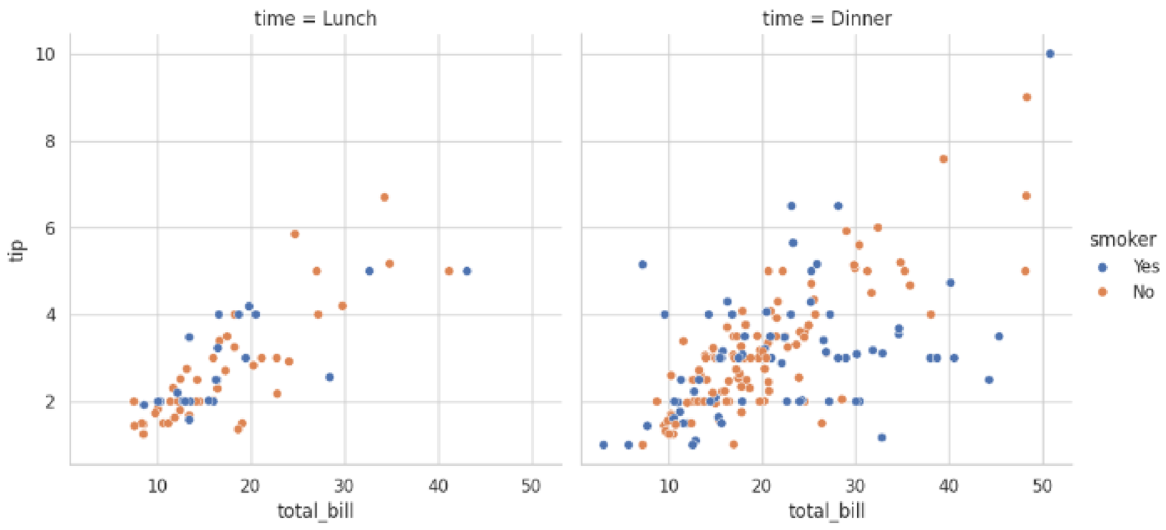


sns.relplot(data=ds, x="total_bill", y="tip", hue="smoker", col="time")

plt.show()

Output: Two scatter plots (Lunch & Dinner) comparing smoker vs non-smoker.

```
<seaborn.axisgrid.FacetGrid at 0x7f52c4b42de0>
```


time = Lunch
time = Dinner
tip
total_bill
total_bill
10
8
6
4
2
10
20
30
40
50
smoker
Yes
No