Plotting with Seaborn

Seaborn simplifies the creation of fundamental plots. This subunit will guide you through generating scatter plots, line plots, and bar plots using Seaborn's user-friendly API (Application Programing Interface). You'll learn how Seaborn abstracts away much of the complexity, allowing you to create visually appealing plots with minimal code.

Let us learn to create various plots using Seaborn:

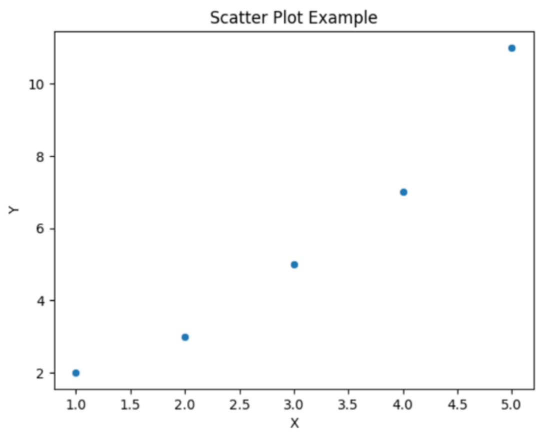
a. Generating Scatter Plots

Scatter plots are effective for visualizing the relationship between two continuous variables. Seaborn simplifies the process of creating scatter plots with the scatterplot() function. Let's consider an example:

```
# Sample data
import pandas as pd
data = pd.DataFrame({'X': [1, 2, 3, 4, 5], 'Y': [2,
3, 5, 7, 11]})

# Scatter plot using Seaborn
sns.scatterplot(x='X', y='Y', data=data)
plt.title('Scatter Plot Example')
plt.show()
```

Output

The Python code uses the **scatterplot()** function to create a scatter plot of the 'X' and 'Y' variables from the dataset.

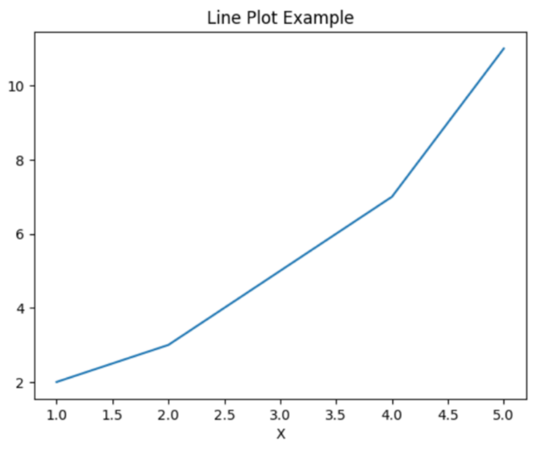


b. Line Plots with Seaborn

Line plots are effective for visualizing trends in data. Seaborn simplifies the creation of line plots using the lineplot() function. Consider the following example:

```
# Line plot using Seaborn
sns.lineplot(x='X', y='Y', data=data)
plt.title('Line Plot Example')
plt.show()
```

Output:



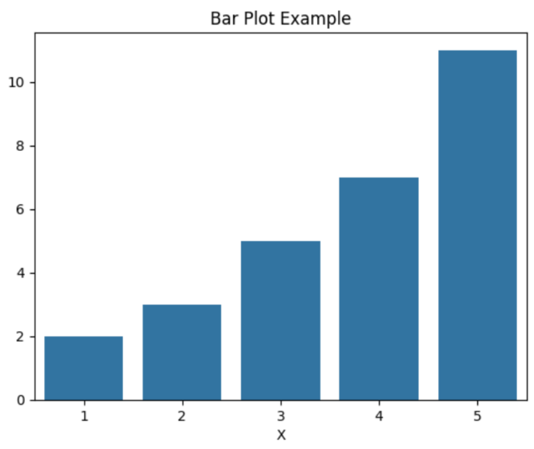
The **lineplot()** function allows us to visualize the trend between two variables over a continuous axis, in this case, 'X' and 'Y'.

c. Creating Bar Plots

Bar plots are useful for comparing categories or groups. Seaborn's barplot() function simplifies the process of creating bar plots. Let's explore an example:

```
# Bar plot using Seaborn
sns.barplot(x='X', y='Y', data=data)
plt.title('Bar Plot Example')
plt.show()
```

Output:



Here, the **barplot()** function helps us compare the values of 'Y' for different categories represented by 'X'.

Utilizing Seaborn's Simplified API (Application Programming Interface)

Seaborn provides a simplified API for common visualizations, making it even more user-friendly. The **sns.relplot**, **sns.lineplot**, and **sns.catplot** functions are part of this simplified API, allowing for easy creation of scatter plots, line plots, and categorical plots, respectively.

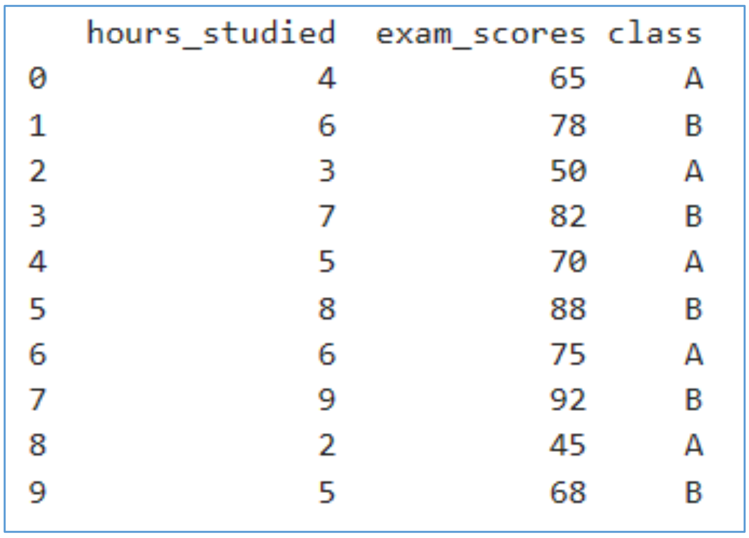
Let's explore these functions with examples:

a. Create Data: Let's first create data using pandas.

```
import pandas as pd

# Creating a hypothetical student_data DataFrame
data = {
    'hours_studied': [4, 6, 3, 7, 5, 8, 6, 9, 2, 5],
    'exam_scores': [65, 78, 50, 82, 70, 88, 75, 92, 45,
68],
    'class': ['A', 'B', 'A', 'B', 'A', 'B', 'A', 'B',
'A', 'B']
}

student_data = pd.DataFrame(data)

# Displaying the student_data DataFrame
print(student_data)
```

Output:

The code uses pandas to create a DataFrame named **student_data** with columns 'hours_studied', 'exam_scores', and 'class'. It then prints the DataFrame, displaying hypothetical student data.

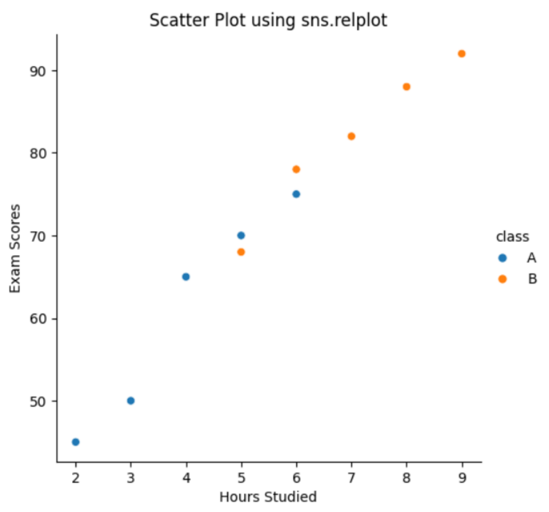


```
   hours_studied  exam_scores class
0              4           65     A
1              6           78     B
2              3           50     A
3              7           82     B
4              5           70     A
5              8           88     B
6              6           75     A
7              9           92     B
8              2           45     A
9              5           68     B
```

b. Scatter Plot using **sns.relplot**: Now let's make a Scatter plot using the data named student_data.

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

# Assume you have a DataFrame named student_data
# with columns 'hours_studied', 'exam_scores', and
'class'

# Create a scatter plot using sns.relplot
sns.relplot(x='hours_studied', y='exam_scores',
hue='class', data=student_data, kind='scatter')

# Add labels and title
plt.xlabel('Hours Studied')
plt.ylabel('Exam Scores')
plt.title('Scatter Plot using sns.relplot')

# Show the plot
plt.show()
```

Output:



This code uses the seaborn (sns) and matplotlib (plt) libraries to create a scatter plot. It assumes there is a DataFrame named student_data with columns 'hours_studied', 'exam_scores', and 'class'. The scatter plot is generated using sns.relplot, where 'hours_studied' is on the x-axis, 'exam_scores' on the y-axis, and points are color-coded based on the 'class' column. Additional labels and a title are added using plt.xlabel, plt.ylabel, and plt.title. Finally, the plot is displayed using plt.show().

In the **sns.relplot** function, the **kind** parameter is used to specify the kind of plot to be drawn. In the provided code, **kind='scatter'** is used, indicating that a scatter plot should be created. You can make other plots like Line Plot and Categorical Plot using **sns.relplot** by changing the value in the kind parameter.

Visualizing data directly from example datasets:

Visualizing data directly from example datasets refers to the process of using pre-existing datasets that come bundled with the Seaborn library for the purpose of data exploration and visualization. Seaborn provides a collection of well-known datasets, often used in statistical analysis and machine learning, which users can easily load and visualize without the need for external data sources. This feature is particularly beneficial for learning and practicing data visualization techniques. Let learn more about it in parts:

a. **Loading Example Datasets:** Seaborn provides a convenient function called **load_dataset()** that allows users to load various example datasets directly into their Python environment. These datasets cover a wide range of scenarios, from simple toy datasets to more complex, real-world datasets. Let us consider an example using iris dataset.

```
import seaborn as sns

# Load the Iris dataset
iris = sns.load_dataset('iris')
```

In this example, the Iris dataset, a well-known dataset in the field of machine learning, is loaded using **load_dataset()**.

b. **Exploring Dataset Characteristics**: Once the dataset is loaded, users can explore its structure, dimensions, and contents. Understanding the dataset's features and their relationships is essential for effective visualization.

```
# Display basic information about the Iris dataset
print(iris.info())
```

The **info()** method provides information about the dataset, including the data types of each column and the presence of missing values.

c. **Visualizing Example Datasets**: Seaborn's plotting functions can be applied directly to the loaded example datasets. This simplifies the process of creating informative visualizations for data analysis and interpretation.

```
# Create a pair plot for visualization
sns.pairplot(iris, hue='species')
```

In this case, a pair plot is generated to visualize relationships between different pairs of features in the Iris dataset. The **hue** parameter adds color differentiation based on the species of iris.

d. **Learning and Practicing**: Example datasets are invaluable for learning and practicing data visualization techniques. Users can experiment with various Seaborn functions and parameters, gaining hands-on experience in creating different types of plots.

```
# Experiment with other Seaborn functions on the Iris
dataset
sns.boxplot(x='species', y='sepal_length', data=iris)
```

Here, a box plot is created to visualize the distribution of sepal lengths across different iris species.