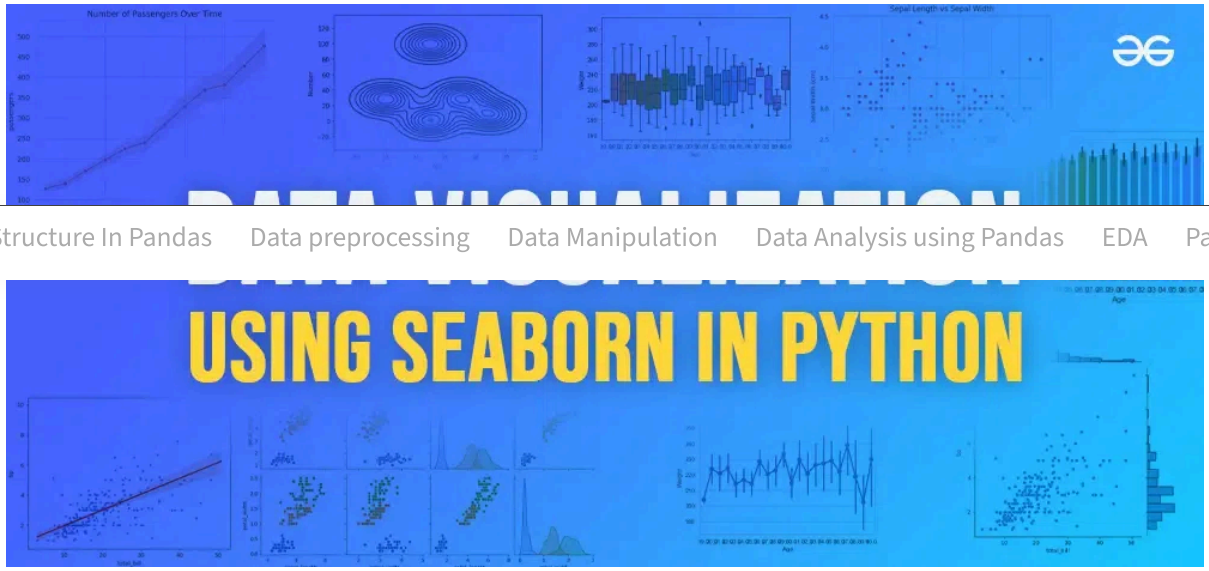
Data Visualization with Seaborn – Python

Last Updated : 29 Aug, 2024

Data Visualization is the presentation of data in pictorial format. It is extremely important for Data Analysis, primarily because of the fantastic ecosystem of data-centric Python packages. And it helps to understand the data, however, complex it is, the significance of data by summarizing and presenting a huge amount of data in a simple and easy-to-understand format and helps communicate information clearly and effectively.



Data Visualization with Seaborn

Data Structure In Pandas Data preprocessing Data Manipulation Data Analysis using Pandas EDA Pandas Exe

Table of Content

Getting Started with Seaborn

Seaborn is a Python data visualization library that simplifies the process of creating complex visualizations. ***It is specifically designed for statistical data visualization, making it easier to understand data distributions and relationships between variables***. Seaborn integrates closely with Pandas data structures, allowing seamless data manipulation and visualization.

It is built on the top of matplotlib library and also closely integrated into the data structures from pandas.

Key Features of Seaborn:

- **High-level interface**: Simplifies the creation of complex visualizations.
- **Integration with Pandas**: Works seamlessly with Pandas DataFrames for data manipulation.
- **Built-in themes**: Offers attractive default themes and color palettes.
- **Statistical plots**: Provides various plot types to visualize statistical relationships and distributions.

Installing Seaborn for Data Visualization

Before using Seaborn, you'll need to install it. The easiest way to install Seaborn is using `pip`, the Python package manager for python environment:

```
pip install seaborn
```

Installing Seaborn

For more, please refer to below links:

- How to Install Seaborn on Windows?
- How To Install Seaborn In Pycharm?
- How to Install Seaborn on Linux?

Creating Basic Plots with Seaborn

Before starting let's have a small intro of bivariate and univariate data:

- **Bivariate data:** This type of data involves **two different variables**. The analysis of this type of data deals with causes and relationships and the analysis is done to find out the relationship between the two variables.
- **Univariate data:** This type of data consists of **only one variable**. The analysis of univariate data is thus the simplest form of analysis since the information deals with only one quantity that changes. It does not deal with causes or relationships and the main purpose of the analysis is to describe the data and find patterns that exist within it.

Seaborn offers a variety of plot types to visualize different aspects of data. ***Seaborn helps to visualize the statistical relationships, to understand how variables in a dataset are related to one another and how that relationship is dependent on other variables, we perform statistical analysis.*** This Statistical analysis helps to visualize the trends and identify various patterns in the dataset. Below are some common plots you can create using Seaborn:

1. Line plot

Lineplot is the most popular plot to draw a relationship between x and y with the possibility of several semantic groupings. It is often used to track changes over intervals.

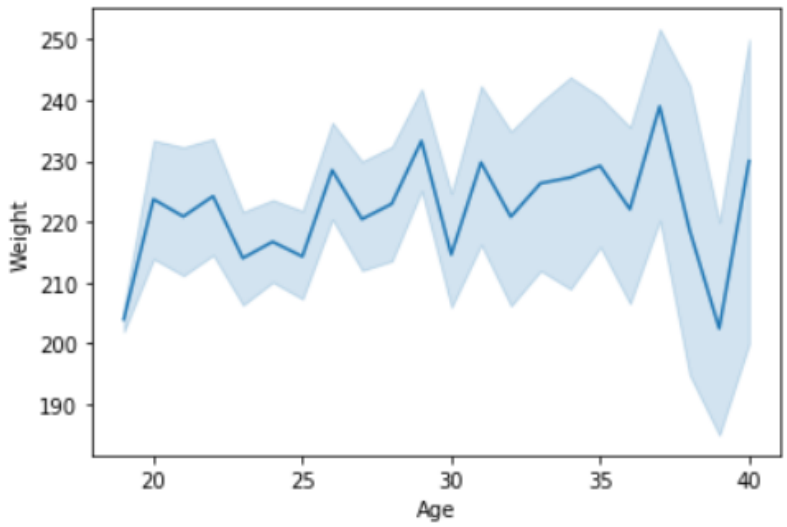
> ***Syntax :*** *sns.lineplot(x=None, y=None)*
>
> ***Parameters:***
>
> ***x, y:*** *Input data variables; must be numeric. Can pass data directly or reference columns in data.*

Let's visualize the data with a line plot and pandas:

Python

```
import pandas as pd
import seaborn as sns

# initialise data of lists
data = {'Name':[ 'Mohe' , 'Karnal' , 'Yrik' , 'jack' ],
        'Age':[ 30 , 21 , 29 , 28 ]}
df = pd.DataFrame( data )

# plotting lineplot
sns.lineplot( data['Age'], data['Weight'])
```

Output:



2. Scatter Plot

Scatter plots are used to visualize the relationship between two numerical variables. They help identify correlations or patterns. It can draw a two-dimensional graph.

Syntax: *seaborn.scatterplot(x=None, y=None)*

Parameters:
x, y: *Input data variables that should be numeric.*

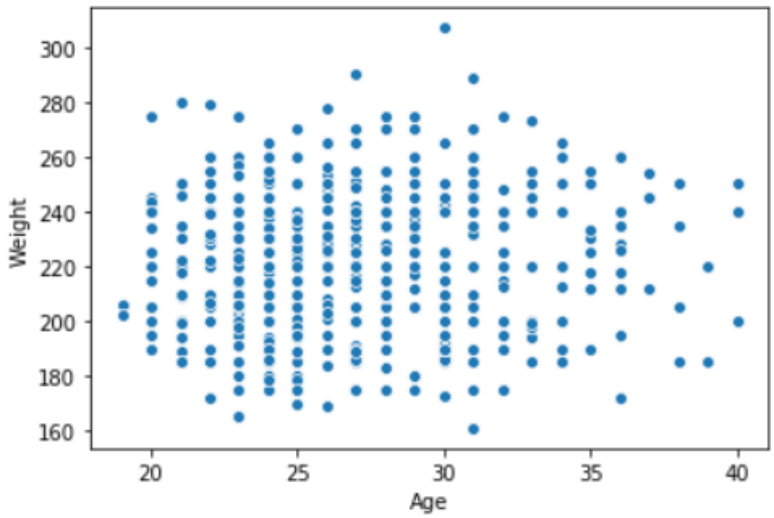
Returns: *This method returns the Axes object with the plot drawn onto it.*

Let's visualize the data with a scatter plot and pandas:

Python

```python
import pandas as pd
import seaborn as sns

# initialise data of lists
data = {'Name':[ 'Mohe' , 'Karnal' , 'Yrik' , 'jack' ],
        'Age':[ 30 , 21 , 29 , 28 ]}
df = pd.DataFrame( data )

seaborn.scatterplot(data['Age'],data['Weight'])
```

Output:



3. Box plot

A box plot (or box-and-whisker plot) s is the visual representation of the depicting groups of numerical data through their quartiles against continuous/categorical data.

A box plot consists of 5 things.

- Minimum
- First Quartile or 25%
- Median (Second Quartile) or 50%
- Third Quartile or 75%

- Maximum

Syntax:

seaborn.boxplot(x=None, y=None, hue=None, data=None)

Parameters:

- ***x, y, hue:*** *Inputs for plotting long-form data.*
- ***data:*** *Dataset for plotting. If x and y are absent, this is interpreted as wide-form.*

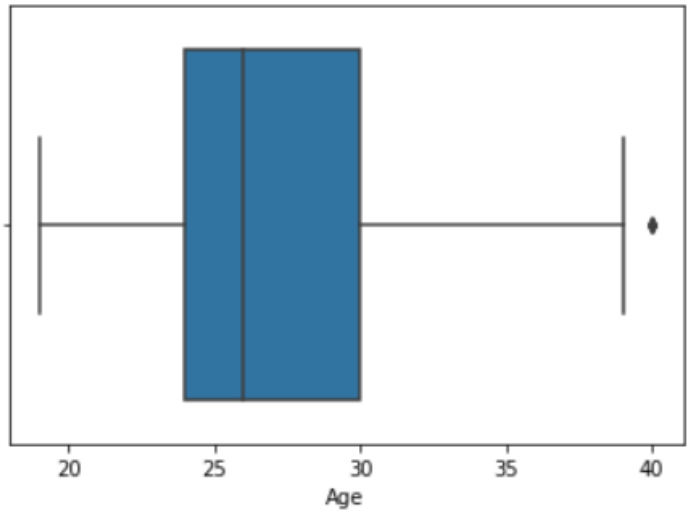
Returns: *It returns the Axes object with the plot drawn onto it.*

Let's create box plot with seaborn with an example.

Python

```python
import pandas as pd
import seaborn as sns

# initialise data of lists
data = {'Name':[ 'Mohe' , 'Karnal' , 'Yrik' , 'jack' ],
        'Age':[ 30 , 21 , 29 , 28 ]}
df = pd.DataFrame( data )
sns.boxplot( data['Age'] )
```

Output:

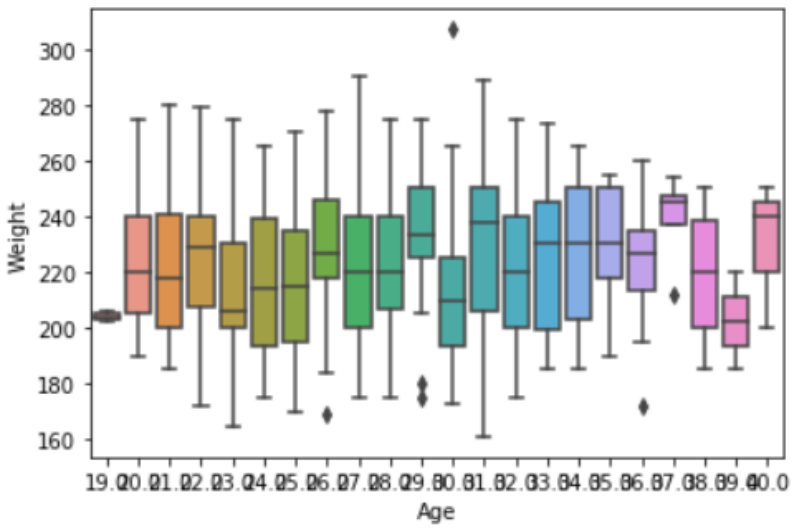


Example 2: Let's see create box plot with more features

Python

```python
# import module
import seaborn as sns
import pandas

# read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'], data['Weight'])
```

Output:



4. Violin Plot

A violin plot is similar to a boxplot. It shows several quantitative data across one or more categorical variables such that those distributions can be compared.

Syntax: *seaborn.violinplot(x=None, y=None, hue=None, data=None)*

Parameters:

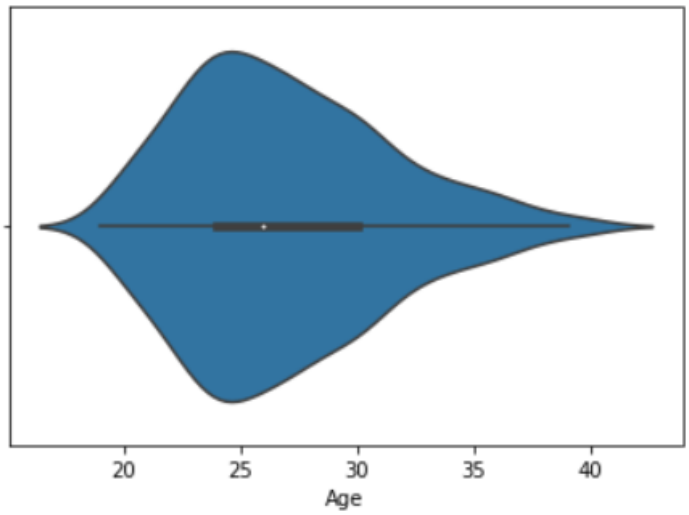
- ***x, y, hue:*** *Inputs for plotting long-form data.*
- ***data:*** *Dataset for plotting.*

Example 1: Draw the violin plot with Pandas

Python

```
import pandas as pd
import seaborn as sns

# initialise data of lists
data = {'Name':[ 'Mohe' , 'Karnal' , 'Yrik' , 'jack' ],
        'Age':[ 30 , 21 , 29 , 28 ]}
df = pd.DataFrame( data )
sns.violinplot(data['Age'])
```

Output:



5. Swarm plot

A swarm plot is similar to a strip plot, We can draw a swarm plot with non-overlapping points against categorical data.

Syntax: *seaborn.swarmplot(x=None, y=None, hue=None, data=None)*

Parameters:

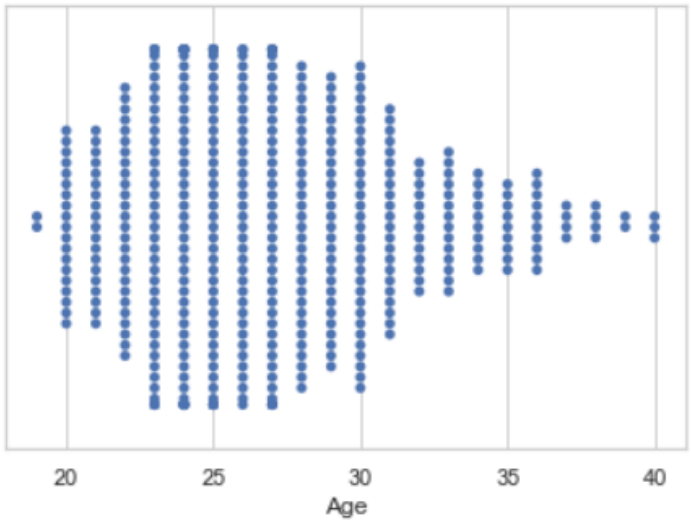
- ***x, y, hue:*** *Inputs for plotting long-form data.*
- ***data:*** *Dataset for plotting.*

Example : Draw the swarm plot with Pandas

Python

```python
# import module
import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv( "nba.csv" )
seaborn.swarmplot(x = data["Age"])
```

Output:



6. Bar plot

Barplot represents an estimate of central tendency for a numeric variable with the height of each rectangle and provides some indication of the uncertainty around that estimate using error bars.

Syntax : *seaborn.barplot(x=None, y=None, hue=None, data=None)*

Parameters :

- ***x, y :*** *This parameter take names of variables in data or vector data, Inputs for plotting long-form data.*
- ***hue :*** *(optional) This parameter take column name for colour encoding.*
- ***data :*** *(optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.*

Returns : *Returns the Axes object with the plot drawn onto it.*

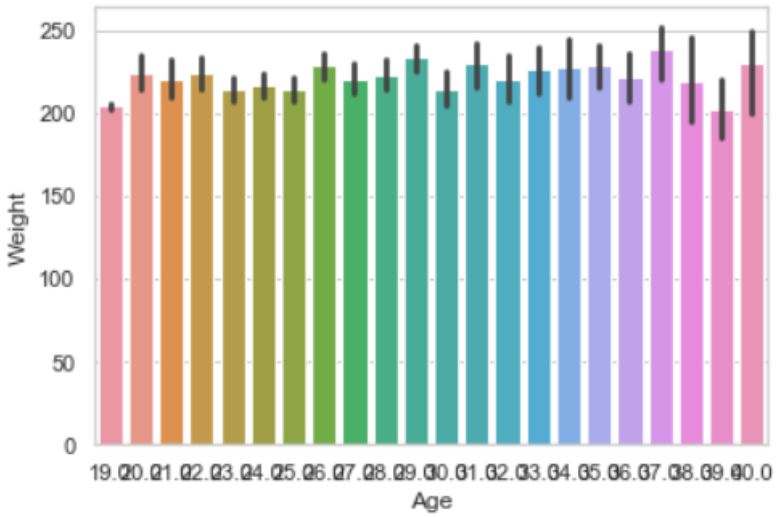
Example : Draw the bar plot with Pandas

Python

```
# import module
import seaborn
```

```
seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.barplot(x ="Age", y ="Weight", data = data)
```

Output:



7. Point plot

Point plot used to show point estimates and confidence intervals using scatter plot glyphs. A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars.

Syntax: *seaborn.pointplot(x=None, y=None, hue=None, data=None)*

Parameters:

- ***x, y:*** *Inputs for plotting long-form data.*
- ***hue:*** *(optional) column name for color encoding.*
- ***data:*** *dataframe as a Dataset for plotting.*

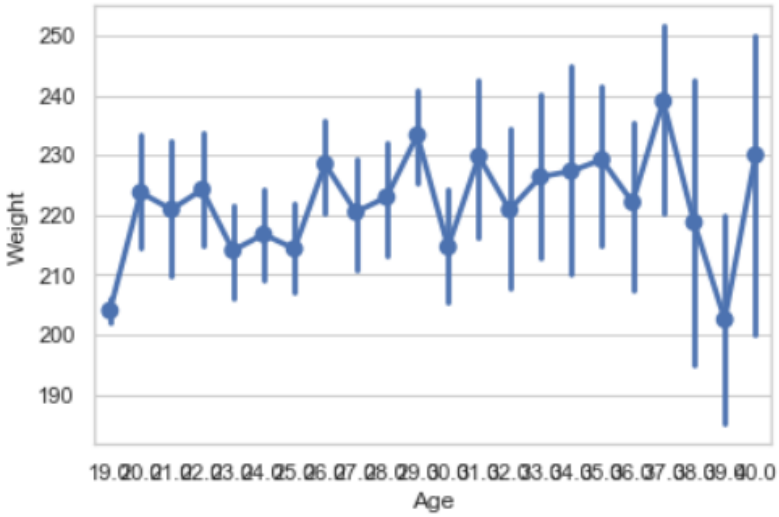
Return: *The Axes object with the plot drawn onto it.*

Example: Draw the point plot with Pandas

Python

```
# import module
import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.pointplot(x = "Age", y = "Weight", data = data)
```

Output:



8. Count plot

Count plot used to Show the counts of observations in each categorical bin using bars.

Syntax : *seaborn.countplot(x=None, y=None, hue=None, data=None)*

Parameters :

- ***x, y:*** *This parameter take names of variables in data or vector data, optional, Inputs for plotting long-form data.*
- ***hue :*** *(optional) This parameter take column name for color encoding.*

- ***data :*** *(optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise, it is expected to be long-form.*

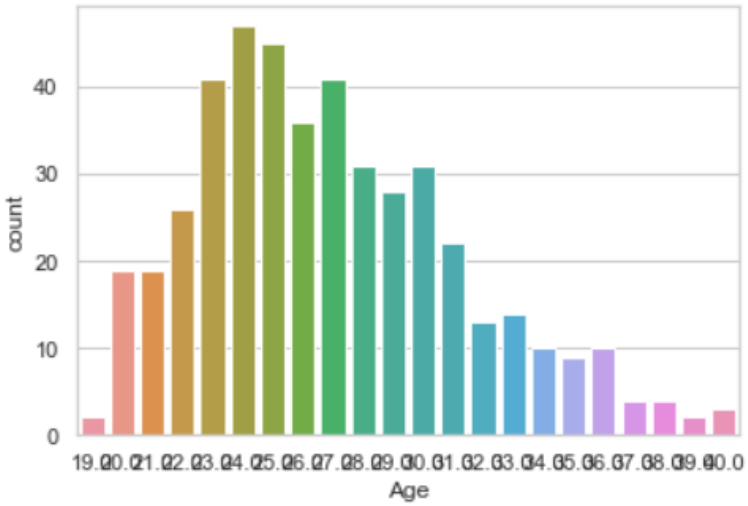
Returns: *Returns the Axes object with the plot drawn onto it.*

Example: Draw the count plot with Pandas

Python

```python
# import module
import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.countplot(data["Age"])
```

Output:



9. KDE Plot

KDE Plot described as **Kernel Density Estimate** is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at

different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.

Syntax: *seaborn.kdeplot(x=None, *, y=None, vertical=False, palette=None, **kwargs)*

Parameters:

x, y : *vectors or keys in data*

vertical : *boolean (True or False)*

data : *pandas.DataFrame, numpy.ndarray, mapping, or sequence*

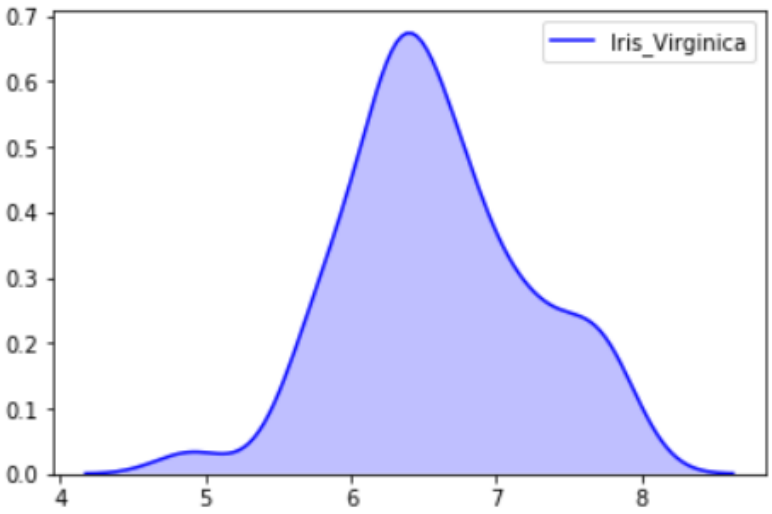
Example : Draw the KDE plot with Pandas

Python

```
# importing the required libraries
from sklearn import datasets
import pandas as pd
import seaborn as sns

# Setting up the Data Frame
iris = datasets.load_iris()

iris_df = pd.DataFrame(iris.data, columns=['Sepal_Length',
                       'Sepal_Width', 'Patal_Length',
'Petal_Width'])

iris_df['Target'] = iris.target

iris_df['Target'].replace([0], 'Iris_Setosa',
inplace=True)
iris_df['Target'].replace([1], 'Iris_Vercicolor',
inplace=True)
iris_df['Target'].replace([2], 'Iris_Virginica',
inplace=True)

# Plotting the KDE Plot
```

```
    sns.kdeplot(iris_df.loc[(iris_df['Target']
    =='Iris_Virginica'),
20              'Sepal_Length'], color = 'b', shade = True,
    Label ='Iris_Virginica')
```

Output:

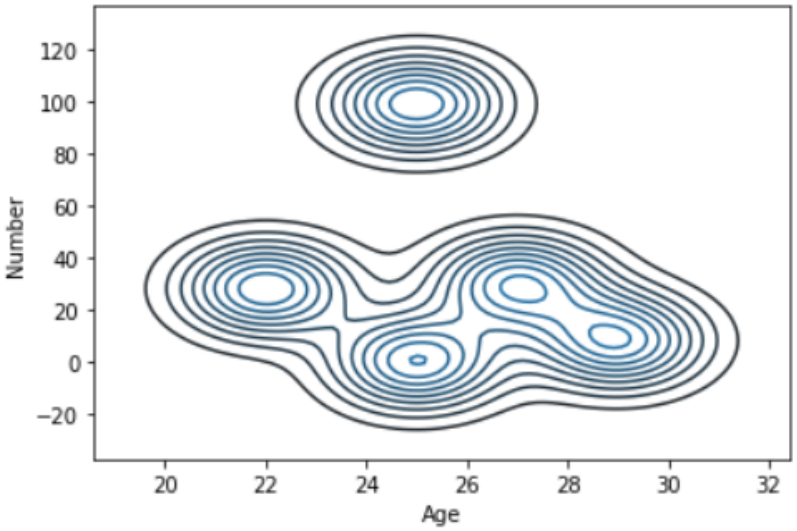


Example 2: KDE plot for Age and Number Feature

Python

```
1  # import module
2  import seaborn as sns
3  import pandas
4
5  # read top 5 column
6  data = pandas.read_csv("nba.csv").head()
7
8  sns.kdeplot( data['Age'], data['Number'])
```

Output:



Bivariate and Univariate data Using Seaborn

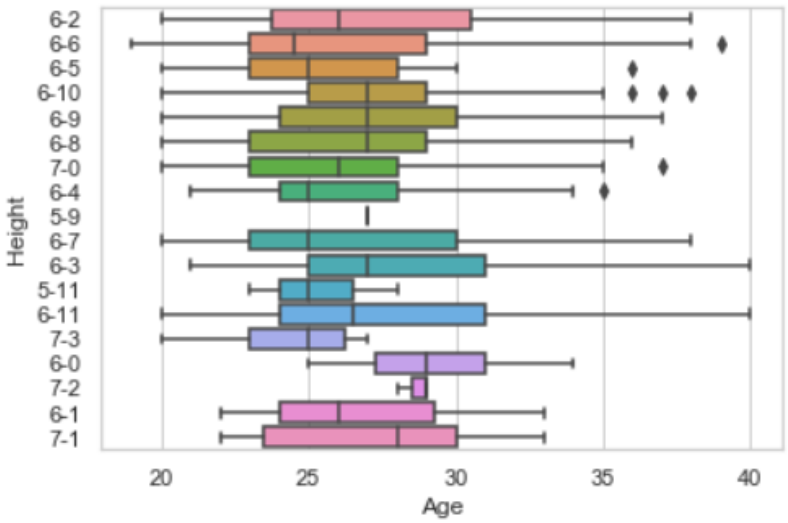
Let's see an example of **Bivariate data :**

Example 1: Using the box plot.

Python

```
# import module
import seaborn as sns
import pandas

# read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'], data['Height'])
```

Output:

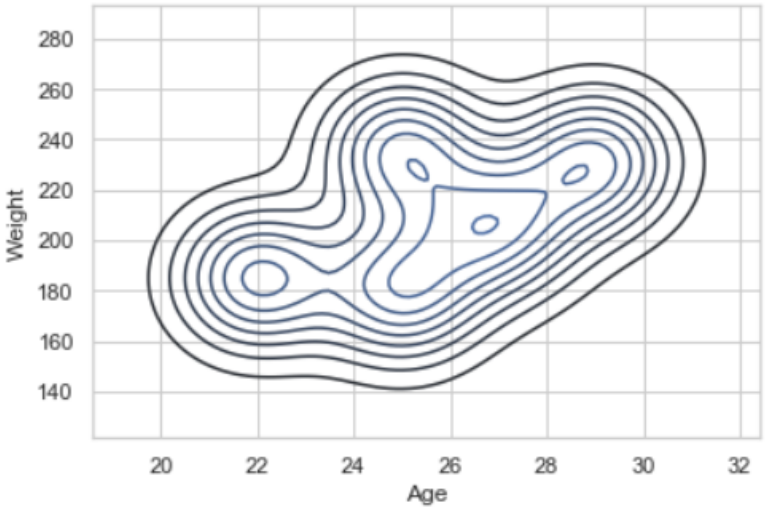


Example 2: Using KDE plot

Python

```python
# import module
import seaborn as sns
import pandas

# read top 5 column
data = pandas.read_csv("nba.csv").head()

sns.kdeplot( data['Age'], data['Weight'])
```

Output:



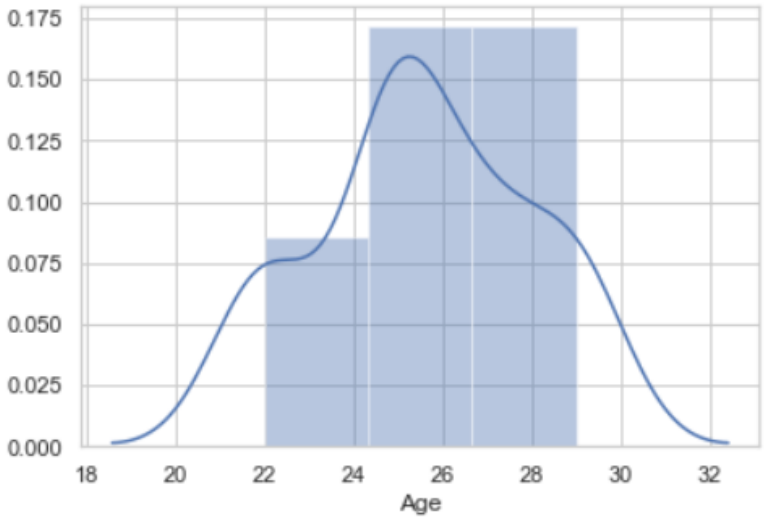
Let's see an example of **univariate data distribution**

Example 1: Using the dist plot

Python

```
# import module
import seaborn as sns
import pandas

# read top 5 column
data = pandas.read_csv("nba.csv").head()

sns.distplot( data['Age'])
```

Output:



Customizing Seaborn Plots with Python

Seaborn plots can be customized extensively to improve their readability and aesthetics.

1. Changing Plot Style and Theme

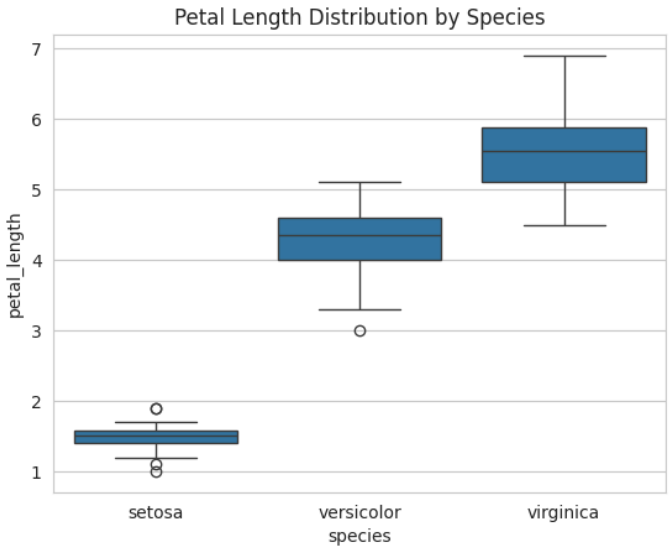
Seaborn offers several built-in themes that can be used to change the overall look of the plots. These themes include `darkgrid`, `whitegrid`, `dark`, `white`, and `ticks`.

Python

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt

# Set the style of the plots
sns.set_style("whitegrid")

# Example plot with the selected style
sns.boxplot(x='species', y='petal_length',
data=sns.load_dataset('iris'))
plt.title('Petal Length Distribution by Species')
plt.show()
```

Output:



Changing Plot Style and Theme

2. Customizing Color Palettes

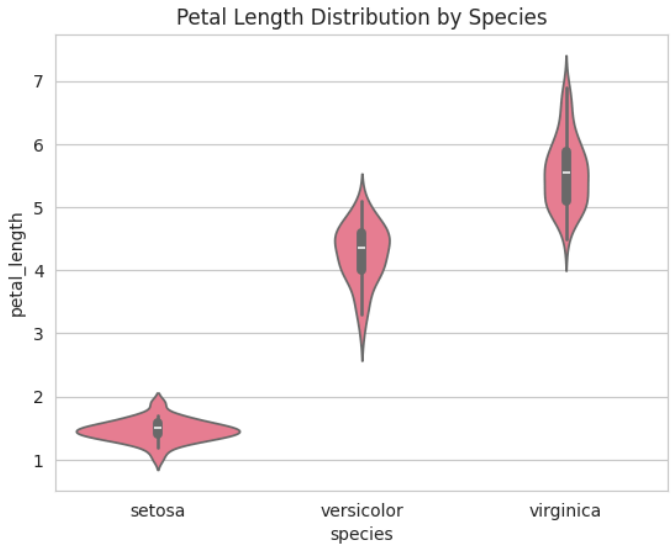
Seaborn allows you to use different color palettes to enhance the visual appeal of your plots. You can use predefined palettes or create custom ones.

Python

```
# Set a custom color palette
custom_palette = sns.color_palette("husl", 8)

# Apply the custom palette
sns.set_palette(custom_palette)

# Example plot with the custom palette
```

```
sns.violinplot(x='species', y='petal_length',
data=sns.load_dataset('iris'))
plt.title('Petal Length Distribution by Species')
plt.show()
```

Output:



Customizing Color Palettes

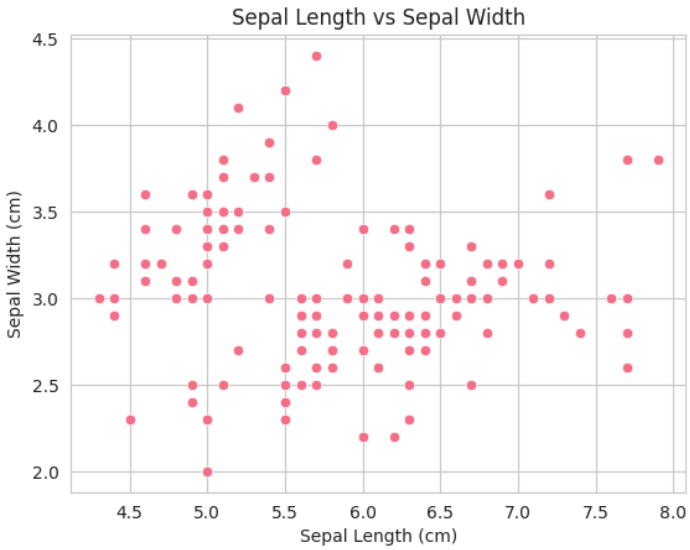
3. Adding Titles and Axis Labels

Adding descriptive titles and labels to your plots can make them more informative.

Python

```
# Adding title and labels
sns.scatterplot(x='sepal_length', y='sepal_width',
data=sns.load_dataset('iris'))
plt.title('Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```

Output:



Adding Titles and Axis Labels

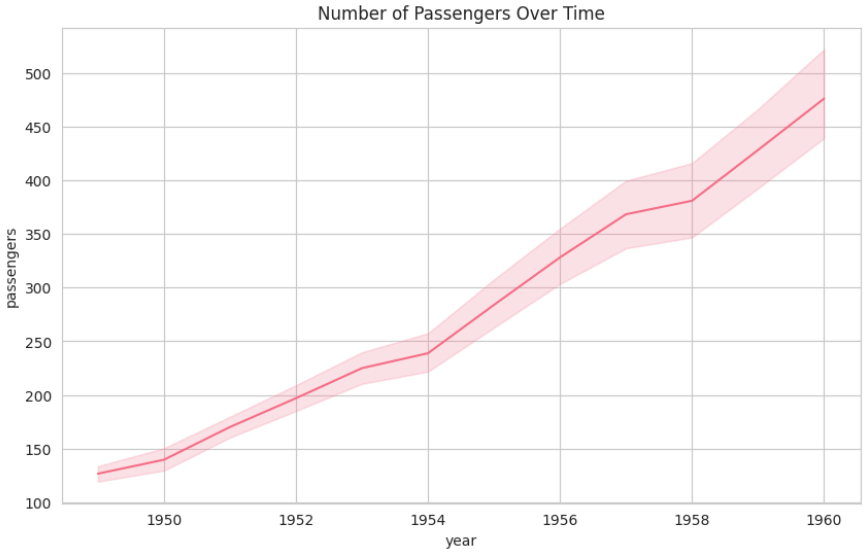
4. Adjusting Figure Size and Aspect Ratio

You can adjust the size of the figure to make it fit better in your presentations or reports.

Python

```python
# Adjust figure size
plt.figure(figsize=(10, 6))

# Example plot with adjusted figure size
sns.lineplot(x='year', y='passengers', data=sns.load_dataset('flights'))
plt.title('Number of Passengers Over Time')
plt.show()
```

Output:



Adjusting Figure Size and Aspect Ratio

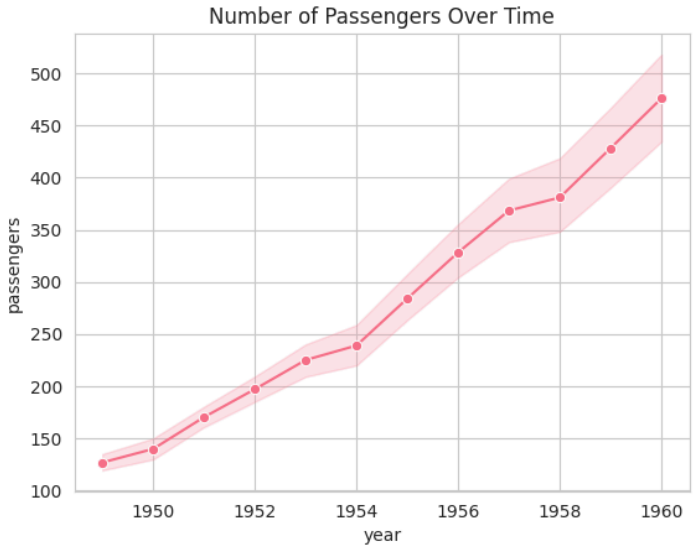
5. Adding Markers to Line Plots

Markers can be added to line plots to highlight data points.

Python

```
# Adding markers to a line plot
sns.lineplot(x='year', y='passengers', data=sns.load_dataset('flights'), marker='o')
plt.title('Number of Passengers Over Time')
plt.show()
```

Output:



Adding Markers to Line Plots

Visualizing Pairwise Relationships with Seaborn: Pair Plots

Pair plots visualize relationships between variables in a dataset. They plot pairwise scatter plots for all combinations of variables, along with univariate distributions on the diagonal. **This is useful for exploring datasets with multiple variables and seeing potential correlations.** Great for exploring patterns, correlations, and distributions in datasets with multiple numeric variables.

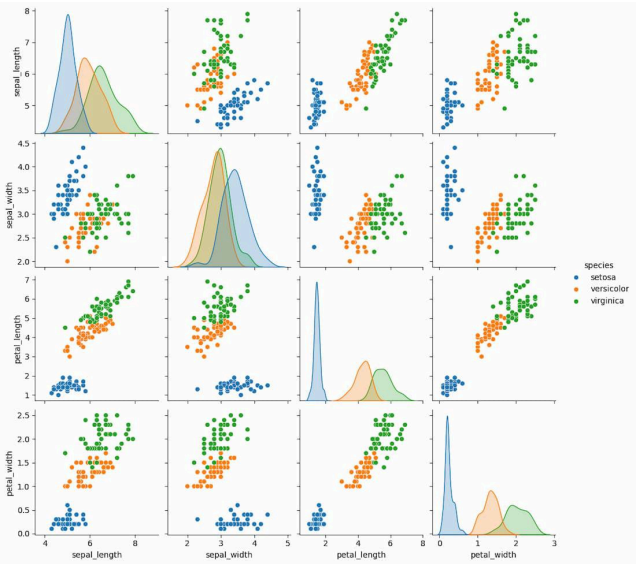
Syntax: `sns.pairplot(data, hue=None)`

Example:

Python

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset("iris")
sns.pairplot(data, hue="species")
plt.show()
```

Output:



Pair plot

The `pairplot` automatically generates a grid of scatter plots showing relationships between each pair of features. The `hue` parameter adds a color

code based on categorical variables like species in the Iris dataset.

Joint Distributions with Seaborn : Joint Plots

Joint plots combine a scatter plot with the distributions of the individual variables. This allows for a quick visual representation of how the variables are distributed individually and how they relate to one another.

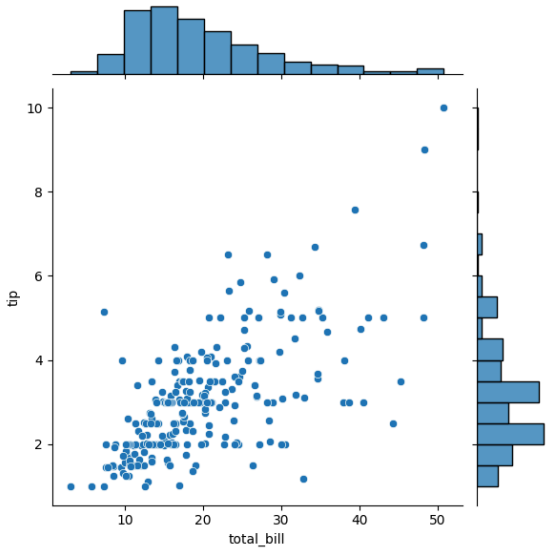
Syntax: `sns.jointplot(x, y, data, kind='scatter')`

Example:

Python

```python
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset("tips")
sns.jointplot(x="total_bill", y="tip", data=data, kind="scatter")
plt.show()
```

Output:



Joint Plot

This creates a scatter plot between `total_bill` and `tip`, with histograms of the individual distributions along the margins. The `kind` parameter can be set to `'kde'` for kernel density estimates or `'reg'` for regression plots.

Understanding Grid Plot Using Seaborn

Grid plots in Seaborn are a powerful way to visualize data across multiple dimensions.

- They allow you to create a grid of plots based on subsets of your data, making it easier to compare different groups or conditions.
- This is particularly useful in exploratory data analysis when you want to understand how different variables interact with each other across different categories.

A grid plot, specifically using Seaborn's **FacetGrid** , is a multi-plot grid that allows you to map a function (such as a plot) onto a grid of subplots.

Creating Multi-Plot Grids with Seaborn's FacetGrid

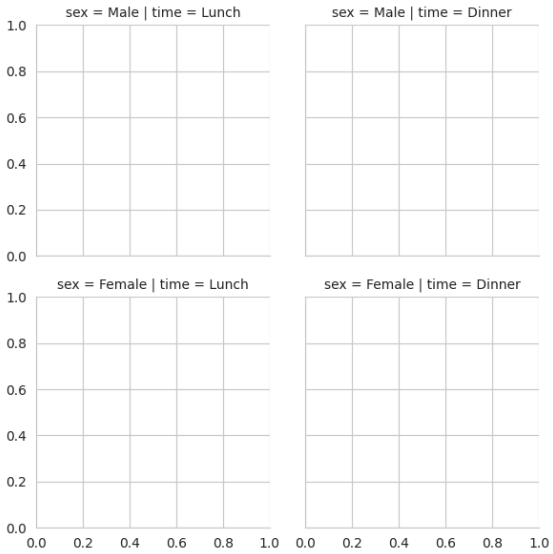
Seaborn's **FacetGrid** is a powerful tool for visualizing data by creating a grid of plots based on subsets of your dataset. It is particularly useful for exploring complex datasets with multiple categorical variables. Here's an in-depth look at what FacetGrid is and how it can be used effectively.

Example: To use FacetGrid, you first need to initialize it with a dataset and specify the variables that will form the row, column, or hue dimensions of the grid. Here is an example using the `tips` dataset:

Python

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

# Load the example dataset
tips = sns.load_dataset("tips")

# Initialize the FacetGrid object
g = sns.FacetGrid(tips, col="time", row="sex")
```

Output:



Multi-Plot Grids with Seaborn's FacetGrid

Illustrating Regression Relationships With Seaborn

Seaborn simplifies the process of performing and visualizing regressions, specifically linear regressions, which is crucial for identifying relationships between variables, detecting trends, and making predictions.

- Seaborn provides various functions that allow you to visualize the results of regressions, along with confidence intervals and residuals.
- This feature is particularly useful in statistical data exploration, enabling users to quickly understand the linear relationship between variables.

Seaborn supports two primary functions for regression visualization:

- **`regplot()`**: This function plots a scatter plot along with a linear regression model fit.
- **`lmplot()`**: This function also plots linear models but provides more flexibility in handling multiple facets and datasets.

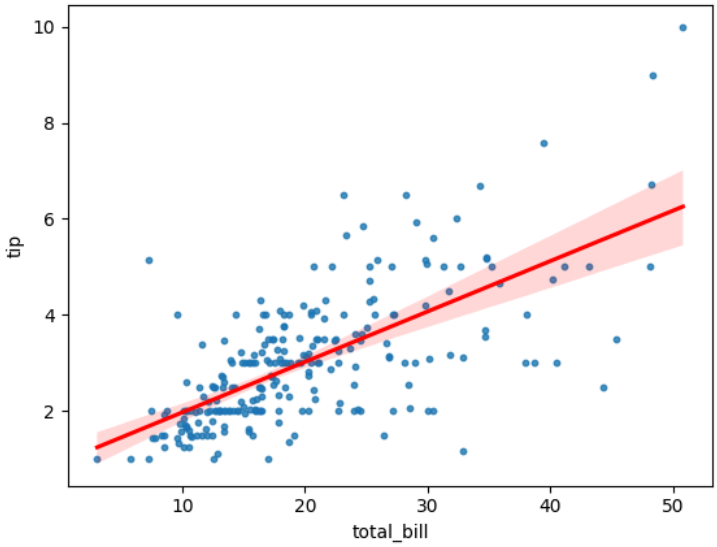
Example: Let's use a simple dataset to visualize a linear regression between two variables: `x` (independent variable) and `y` (dependent variable).

Python

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

# Sample dataset
```

```
tips = sns.load_dataset('tips')

# Plot regression line
sns.regplot(x='total_bill', y='tip', data=tips, scatter_kws={'s':10}, line_kws={'color':'red'})
plt.show()
```

Output:



Reg Plot

Conclusion

In conclusion, Seaborn is an invaluable tool for visualizing data, providing both simplicity and depth for exploratory data analysis and statistical visualization. By leveraging Seaborn's powerful high-level interface, data scientists can create a wide variety of plots to uncover patterns, trends, and relationships within their data. As you explore Seaborn further, experiment with different plot types, customizations, and datasets to gain a deeper understanding of how to communicate your findings visually. With practice, Seaborn can become a key part of your data analysis toolkit.

Data Visualization with Seaborn – FAQs

How does Seaborn handle categorical data visualization?

Seaborn provides several plot types for visualizing categorical data, such as bar plots, box plots, and violin plots. These plots help in

understanding the distribution and relationships between categorical variables

How does Seaborn differ from Matplotlib in data visualization?

Seaborn is built on top of Matplotlib and offers a high-level interface, making it easier to create complex statistical plots with less code. While Matplotlib provides more control over plot customization, Seaborn simplifies the process of creating aesthetically pleasing visualizations with built-in themes and color palettes.

What are the advantages of using Seaborn for data visualization?

Seaborn offers several advantages, including ease of use, integration with Pandas for data manipulation, a variety of built-in statistical functions, and the ability to create complex multi-plot visualizations. It is particularly useful for exploratory data analysis and statistical visualization.

Can Seaborn handle large datasets efficiently?

Seaborn can handle large datasets efficiently, especially when combined with Pandas for data manipulation. However, for extremely large datasets, performance may vary, and it's advisable to preprocess data to optimize visualization performance.

How does Seaborn integrate with Pandas for data visualization?

Seaborn integrates seamlessly with Pandas, allowing users to directly visualize data stored in Pandas DataFrames. This integration simplifies the process of data manipulation and visualization, making it easier to explore and analyze data.

kuma...

16

Previous Article

Python - Data visualization tutorial

Next Article

Python - Data visualization tutorial

Similar Reads

Data Visualization with Seaborn Line Plot

Prerequisite: SeabornMatplotlib Presenting data graphically to emit some information is known as data visualization. It basically is an image to help a...

4 min read

Logarithmic Scaling in Data Visualization with Seaborn

A wide range of libraires like Seaborn built on top of Matplotlib offers informative and attractive statistical graphics. However, the ability to scale axes is considere...

4 min read

Data visualization with Seaborn Pairplot

Data Visualization is the presentation of data in pictorial format. It is extremely important for Data Analysis, primarily because of the fantastic ecosystem of dat...

6 min read

Seaborn Plots in a Loop: Efficient Data Visualization Techniques

Visualizing data is an important aspect of data analysis and exploration. Seaborn is a popular data visualization library which is built on top of matplotlib library. I...

5 min read

Box plot visualization with Pandas and Seaborn

Box Plot is the visual representation of the depicting groups of numerical data through their quartiles. Boxplot is also used for detect the outlier in data set. It...

2 min read

KDE Plot Visualization with Pandas and Seaborn

Kernel Density Estimate (KDE) plot, a visualization technique that offers a detailed view of the probability density of continuous variables. In this article, w...

4 min read

Alternative to Seaborn Pairplot for DataFrame Visualization

Data visualization is key for understanding patterns and relationships within data. While Seaborn's pairplot is widely used for visualizing pairwise relationships in...

3 min read

Why Data Visualization Matters in Data Analytics?

What if you wanted to know the number of movies produced in the world per year in different countries? You could always read this data in the form of a black...

7 min read

Difference Between Data Mining and Data Visualization

Data mining: Data mining is the method of analyzing expansive sums of data in an exertion to discover relationships, designs, and insights. These designs,...

2 min read

Difference Between Data Visualization and Data Analytics

Data Visualization: Data visualization is the graphical representation of information and data in a pictorial or graphical format(Example: charts, graphs,...

3 min read

Article Tags : AI-ML-DS Data Visualization Technical Scripter AI-ML-DS With Python

+4 More