ta-visualization-with-iris-dataset

September 17, 2024

1 Data Visualization With Matplotlib And Seaborn Using Iris Dataset:

1.1 Matplotlib:

Matplotlib is a powerful and popular Python library used for data visualization. It provides a wide range of plotting capabilities and is commonly used to create static, animated, and interactive visualizations. Its flexibility allows users to generate publication-quality plots with just a few lines of code.

Key Features:

2D Plotting: Matplotlib excels in 2D graphics and can create a wide variety of charts, including line plots, bar charts, scatter plots, histograms, and more.

Customizability: Almost every aspect of the plots (titles, labels, legends, colors, grid lines, etc.) can be customized.

Integration: Works well with other Python libraries like NumPy, Pandas, and SciPy for handling and visualizing data.

Subplots and Figures: Users can create complex layouts with multiple subplots and figures in one window.

Interactive Plots: It supports interactive backends, allowing zooming and panning, making it great for exploration.

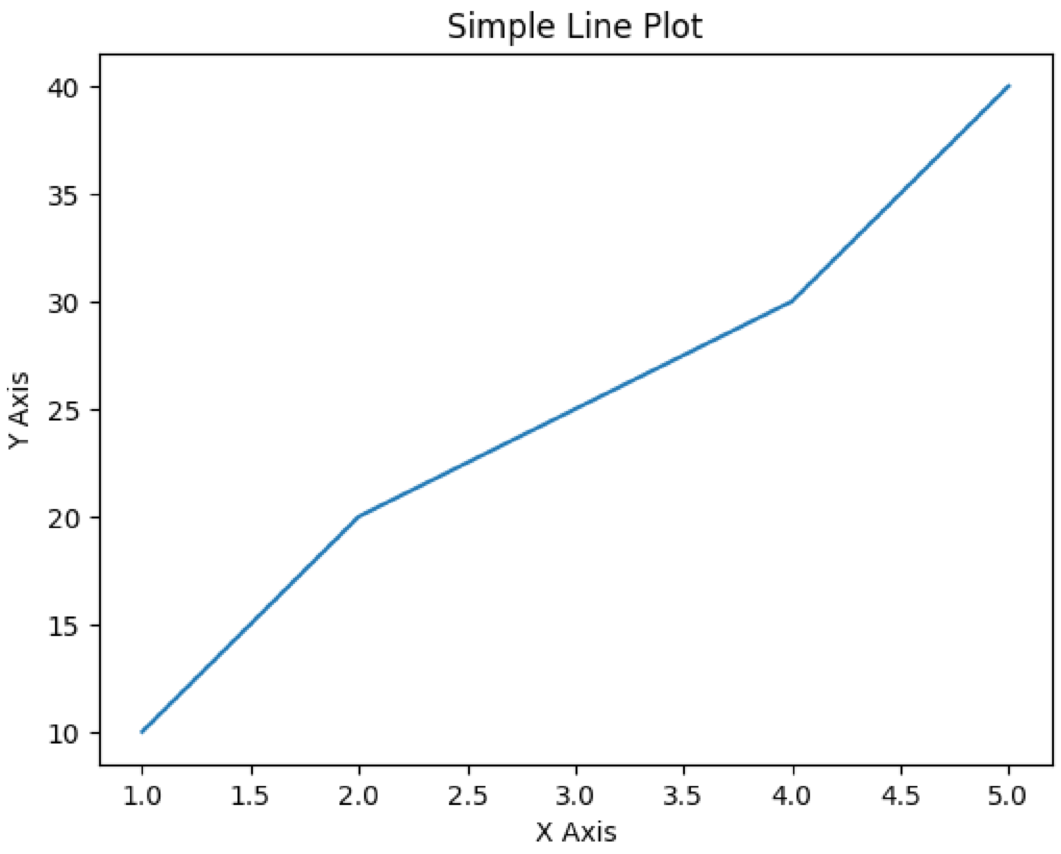
To use matplotlib firstly we have to import pyplot module,which provides a MATLAB-like interface for creating plots:

[ ]:
```
import matplotlib.pyplot as plt
```

Example:

[ ]:
```
# Sample data
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]

# Create a simple line plot
plt.plot(x, y)
#giving titles to lables
plt.title('Simple Line Plot')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
#display the plot
plt.show()
```



1.2 Seaborn

Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating visually appealing and informative statistical graphics with less effort. Seaborn is well-known for making complex plots easier to create and for its focus on visualizing statistical relationships between variables.

Key Features:

High-Level Interface: Seaborn simplifies the process of creating common plots, such as histograms, bar plots, box plots, and scatter plots, with less code compared to Matplotlib.

Built-in Themes: Seaborn comes with aesthetically pleasing default themes and color palettes, making your plots look better with minimal adjustments.

Statistical Plotting: It is designed specifically for statistical plotting, making it easy to show data distributions, relationships, and trends.

Works Well with Pandas: Seaborn integrates smoothly with Pandas DataFrames, allowing you to plot data directly from structured datasets.

Multi-Plot Grids: It has features for creating grids of plots based on data categories, such as pairplot() and facetgrid(), which help visualize relationships across multiple variables.

Common Plot Types:

Relational plots: For visualizing relationships between variables (e.g., scatterplot, lineplot).

Distribution plots: For displaying the distribution of a dataset (e.g., histplot, kdeplot, boxplot).

Categorical plots: For comparing categories (e.g., barplot, countplot, violinplot).

To use seaborn ,we have to import it as:

```
[ ]: import seaborn as sns
```

Example:

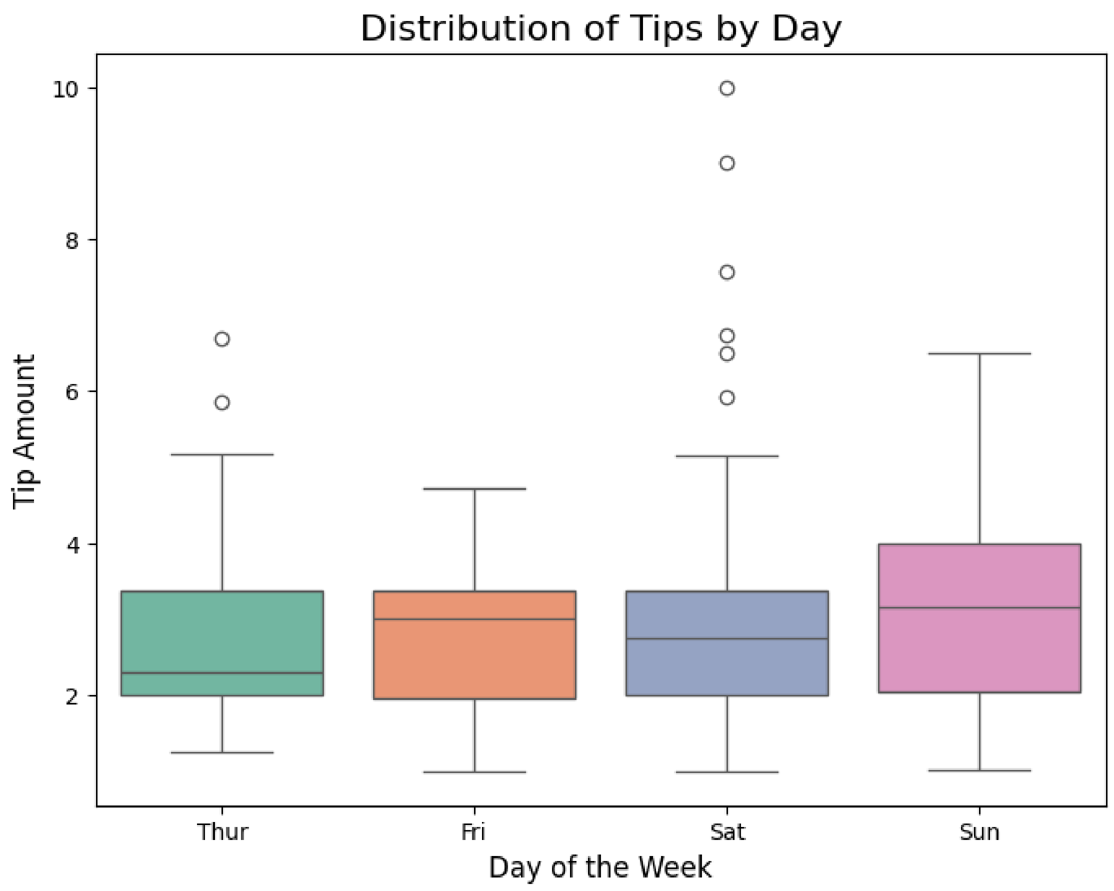
```
[ ]: # Load the built-in 'tips' dataset
     tips = sns.load_dataset('tips')
```

```
[ ]: # Create a box plot to visualize the distribution of tips based on the day
     plt.figure(figsize=(8, 6))
     sns.boxplot(x='day', y='tip', data=tips, palette='Set2')
      #Add title and labels
     plt.title('Distribution of Tips by Day', fontsize=16)
     plt.xlabel('Day of the Week', fontsize=12)
     plt.ylabel('Tip Amount', fontsize=12)
     # Display the plot
     plt.show()
```

```
<ipython-input-13-400bfcddbe1d>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='day', y='tip', data=tips, palette='Set2')
```



Iris Dataset: The Iris dataset is a well-known dataset in machine learning and statistics, often used for classification tasks. It was introduced by the British biologist and statistician Ronald A. Fisher in 1936. The dataset contains measurements of four features from three different species of the Iris flower. These features are:

1.Sepal length (in cm)

2.Sepal width (in cm)

3.Petal length (in cm)

4.Petal width (in cm)

The dataset consists of 150 samples:

50 samples each from three different Iris species:

1.Iris setosa

2.Iris versicolor

3.Iris virginica

Each row in the dataset represents the measurements of one flower, along with the species to which it belongs.

The dataset is popular for testing and implementing classification algorithms, as the task is to predict the species of an Iris flower based on its features. It is readily available in many machine learning libraries, such as scikit-learn in Python.

1.2.1 Sample Exercises :

1. General Statistics Plot (Matplotlib or Seaborn): Python program to create a plot that gives a general statistical summary of the Iris data. You can use seaborn's pairplot or pandas' describe() for guidance

Firstly,we can load the dataset using the following code:

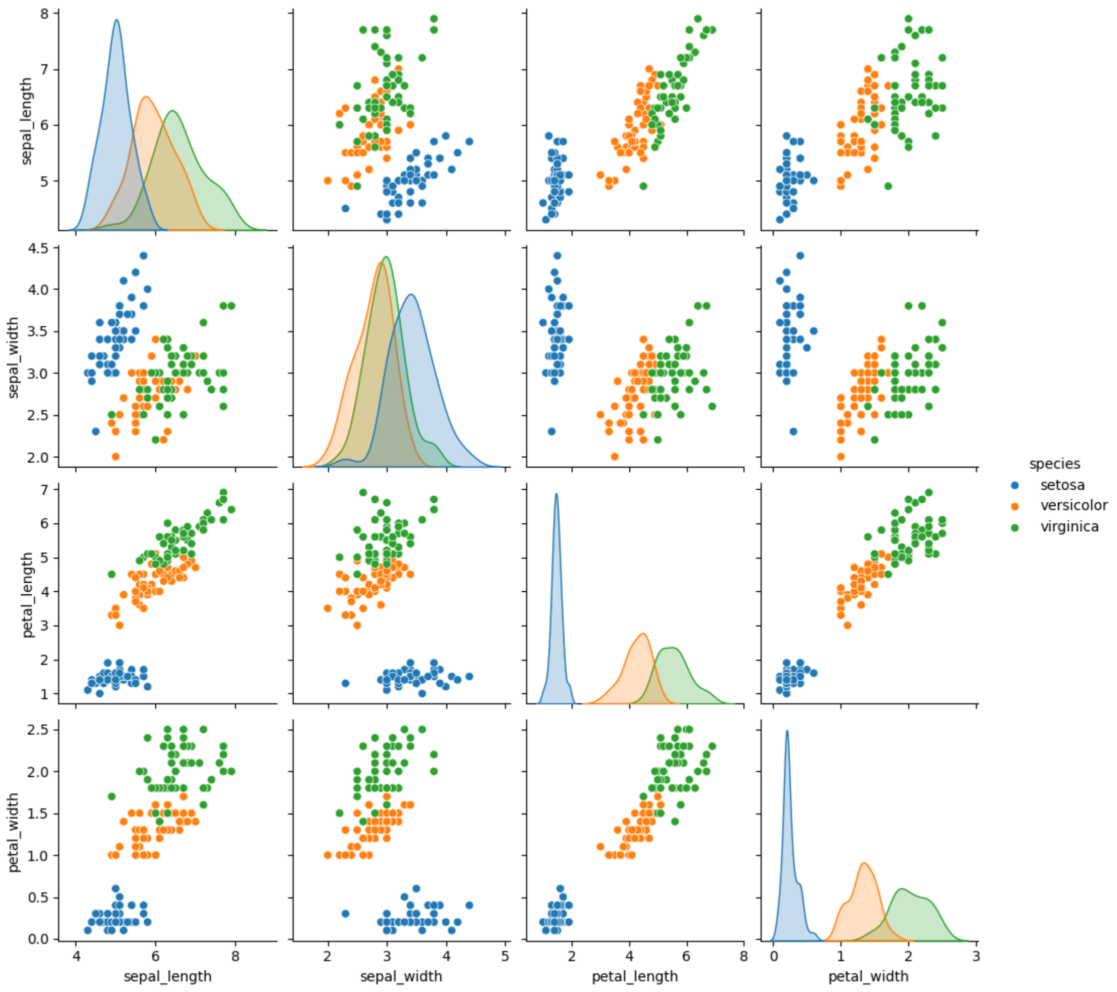
[34]:
```
import seaborn as sns
import matplotlib.pyplot as plt
# Load the Iris dataset
iris=sns.load_dataset('iris')
```

General statistics plot using Seaborn:

[35]:
```
# General statistical summary using pandas describe()
summary = iris.describe()
print("Statistical Summary of Iris Data:\n", summary)
```

```
Statistical Summary of Iris Data:
        sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

[36]:
```
# Pairplot to visualize relationships between features
sns.pairplot(iris, hue='species', height=2.5)
#display the graph
plt.show()
```



1.2.2 Explanation:

Statistical Summary: iris.describe() provides descriptive statistics, including the count, mean, standard deviation, min, and max values for each numerical feature.

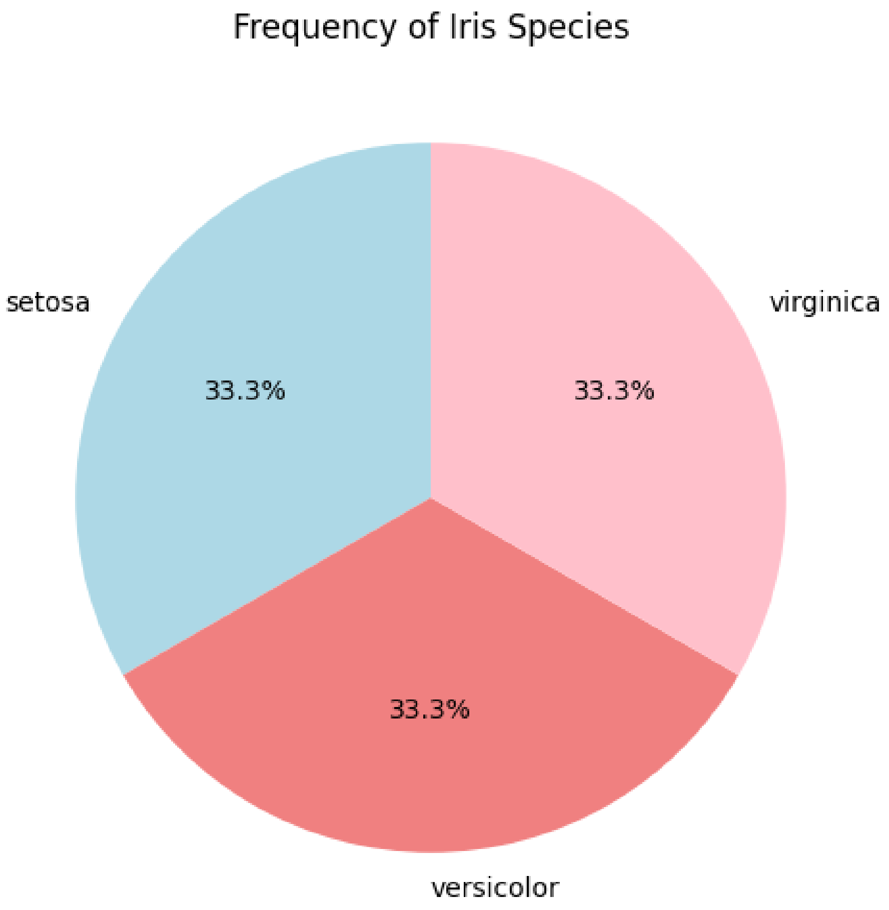
Pairplot: Seaborn's pairplot() visualizes pairwise relationships in the dataset, colored by species.

2. Pie Plot for Species Frequency: Python program to create a pie chart to display the frequency of the three species (setosa, versicolor, virginica) in the Iris dataset.

[37]:
```
# Calculate the frequency of each species
species_count = iris['species'].value_counts()
```

[39]:
```
# Creating a pie chart
plt.figure(figsize=(6, 6))
plt.pie(species_count, labels=species_count.index, autopct='%1.1f%%',␣
 ↪startangle=90, colors=['lightblue','lightcoral','pink'])
```

```
plt.title("Frequency of Iris Species")
plt.show()
```



1.2.3 Explanation:

species_count: This computes the frequency of each species in the Iris dataset using value_counts().

plt.pie(): Creates the pie chart with species labels and percentages displayed.

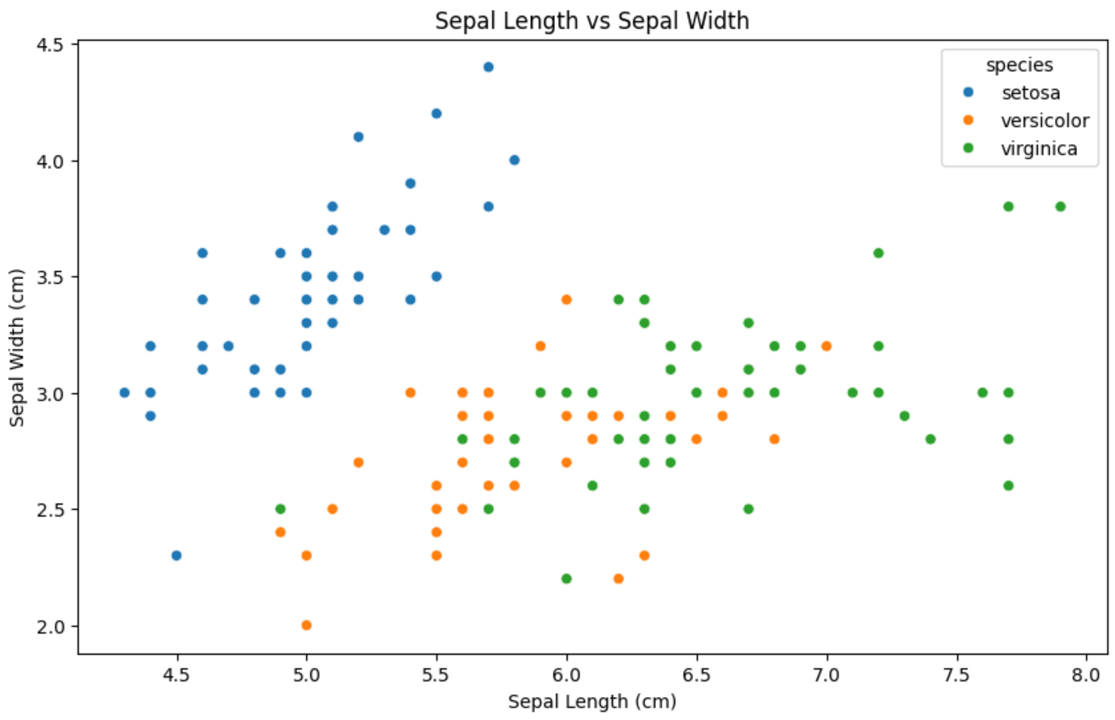
autopct='%1.1f%%':This parameter adds percentages

startangle=90:This rotates the pie for better visualization.

Colors: Custom colors are used for each slice of the pie.

3.Relationship Between Sepal Length and Width: Python program to create a scatter plot to find the relationship between sepal length and sepal width for the Iris dataset.

```
[ ]: #creating a figure of size(10,6)
     plt.figure(figsize=(10, 6))
     #plotting grapg between Sepal Length vs Sepal Width
     sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=iris)
     #giving titles to the axis and labels
     plt.title('Sepal Length vs Sepal Width')
     plt.xlabel('Sepal Length (cm)')
     plt.ylabel('Sepal Width (cm)')
     plt.show()
```



This scatter plot visualizes the relationship between sepal length and sepal width across different species in the Iris dataset. ### Explanation: sns.scatterplot(): This function creates a scatter plot with sepal_length on the x-axis and sepal_width on the y-axis. The hue='species' argument colors the points based on species, and palette='viridis' specifies the color palette. The edgecolor=None removes borders around the scatter points.

Labels and Title: Axis labels and a plot title are added for clarity.

4. Distribution of Sepal and Petal Features: Python program to create a plot that shows how the length and width of sepal length, sepal width, petal length, and petal width are distributed.

```
[38]: # Create a figure with subplots
      plt.figure(figsize=(10, 8))

      # Plot for Sepal Length
```

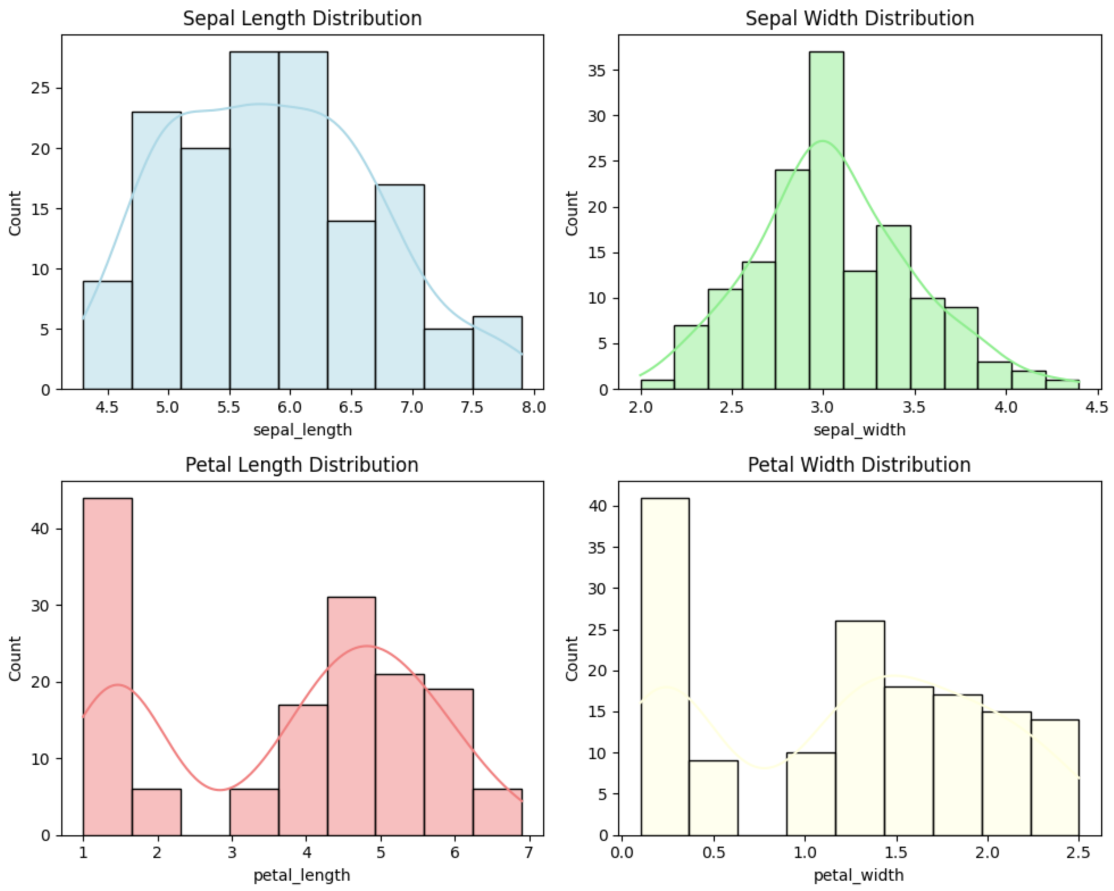
```
plt.subplot(2, 2, 1)
sns.histplot(iris['sepal_length'], kde=True, color='lightblue')
plt.title('Sepal Length Distribution')

# Plot for Sepal Width
plt.subplot(2, 2, 2)
sns.histplot(iris['sepal_width'], kde=True, color='lightgreen')
plt.title('Sepal Width Distribution')

# Plot for Petal Length
plt.subplot(2, 2, 3)
sns.histplot(iris['petal_length'], kde=True, color='lightcoral')
plt.title('Petal Length Distribution')

# Plot for Petal Width
plt.subplot(2, 2, 4)
sns.histplot(iris['petal_width'], kde=True, color='lightyellow')
plt.title('Petal Width Distribution')

# Adjust layout
plt.tight_layout()
plt.show()
```



1.2.4 Explanation:

Sepal and Petal Distributions: Four subplots are created to display the distribution of the sepal length, sepal width, petal length, and petal width.

sns.histplot(): This function plots histograms of the data with optional kernel density estimation (kde=True).

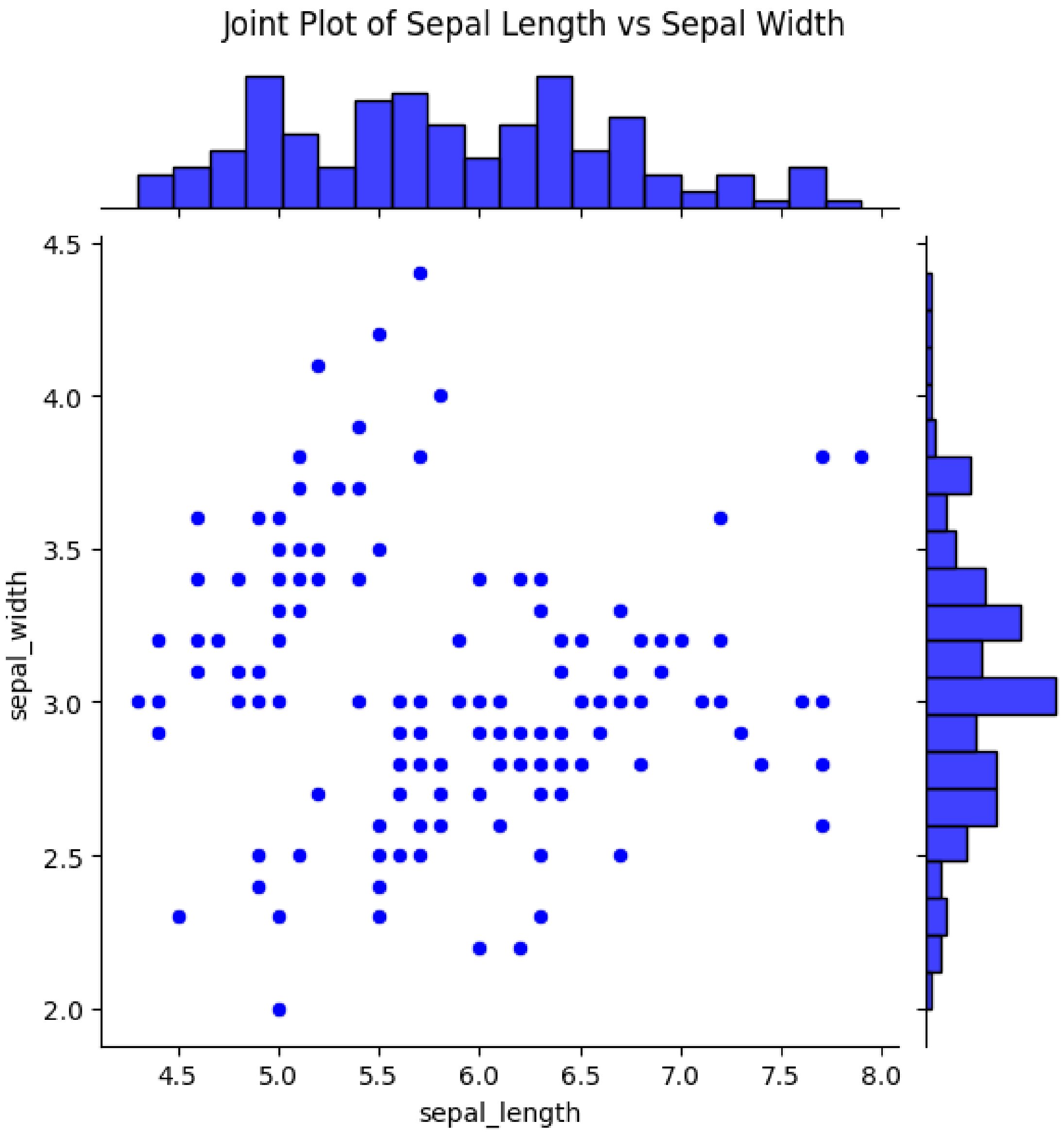
Colors: Each subplot uses a distinct color for clarity.

plt.subplot(): This arranges the subplots in a 2x2 grid.

5. Jointplot of Sepal Length vs Sepal Width: Python program to create a joint plot to describe the individual distributions on the same plot between sepal length and sepal width.

[27]:
```
# Create a joint plot for sepal length and sepal width
sns.jointplot(x='sepal_length', y='sepal_width', data=iris, kind='scatter',␣
↪color='blue', marginal_kws=dict(bins=20, fill=True))
# Add a title to the plot
plt.suptitle("Joint Plot of Sepal Length vs Sepal Width", y=1.03)
# Show the plot
```

```
plt.show()
```



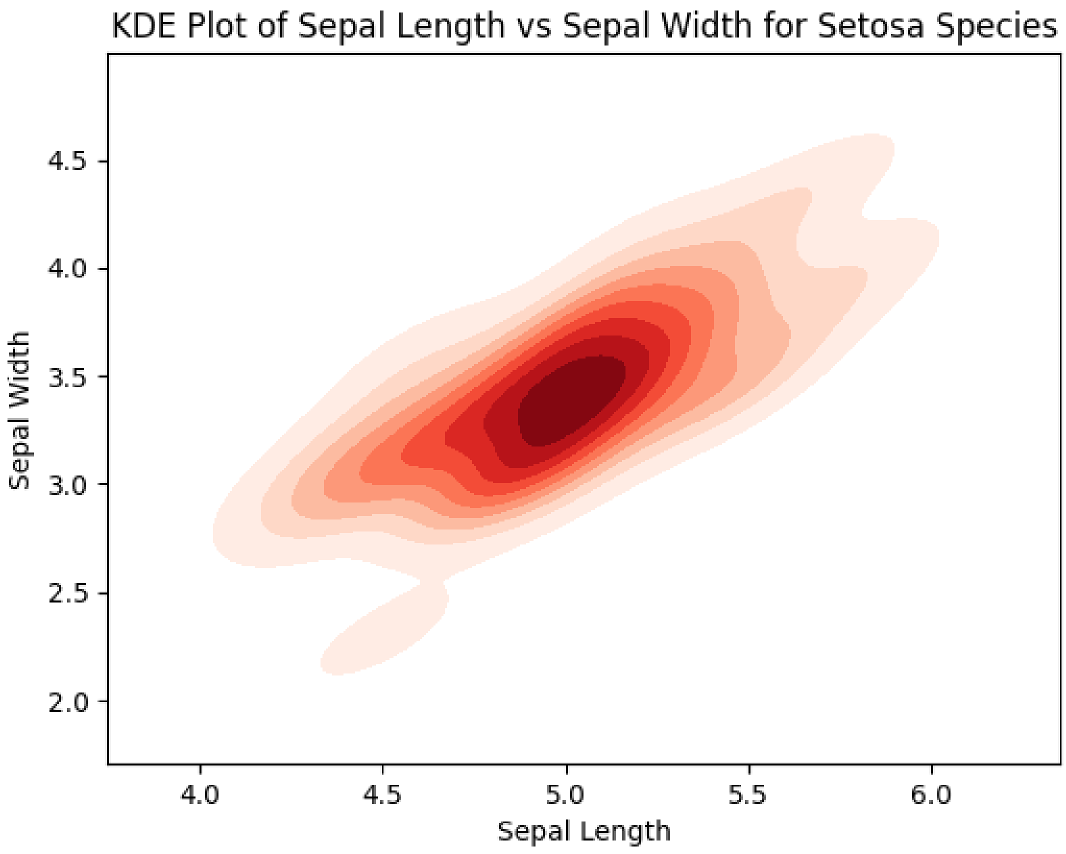
This plot provides a comprehensive view of how sepal length and sepal width are distributed and how they relate to each other in the dataset. ### Explanation: sns.jointplot(): This function creates a scatter plot to visualize the relationship between sepal length and sepal width. Marginal histograms display the individual distributions of each variable.

kind='scatter': Specifies that the central plot should be a scatter plot. You can change this to kind='kde' for a density plot or kind='hex' for a hexagonal bin plot.

marginal_kws: Additional keyword arguments for customizing the appearance of the marginal distributions (histograms). In this case, bins and filled histograms are added.

6. KDE Plot for Setosa Species (Sepal Length vs Sepal Width): Python program using seaborn to create a KDE (Kernel Density Estimate) plot of sepal length versus sepal width for the setosa species of the Iris dataset.

[32]:
```
# Filter the dataset for the Setosa species
setosa = iris[iris['species'] == 'setosa']
# Create a KDE plot for sepal length vs sepal width
sns.kdeplot(x='sepal_length', y='sepal_width', data=setosa, fill=True,␣
 ↪cmap="Reds", thresh=0.05)
# Add titles and labels
plt.title("KDE Plot of Sepal Length vs Sepal Width for Setosa Species")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
# Show the plot
plt.show()
```



This KDE plot provides insight into the density and distribution of sepal length and sepal width for the Setosa species. ### Explanation: Data Filtering: The dataset is filtered to include only the rows where the species is setosa.

sns.kdeplot(): This function creates a 2D KDE plot to visualize the density of points (sepal length vs sepal width) for the Setosa species. The fill=True argument fills the contour levels, and cmap='Reds' specifies the color map for shading.

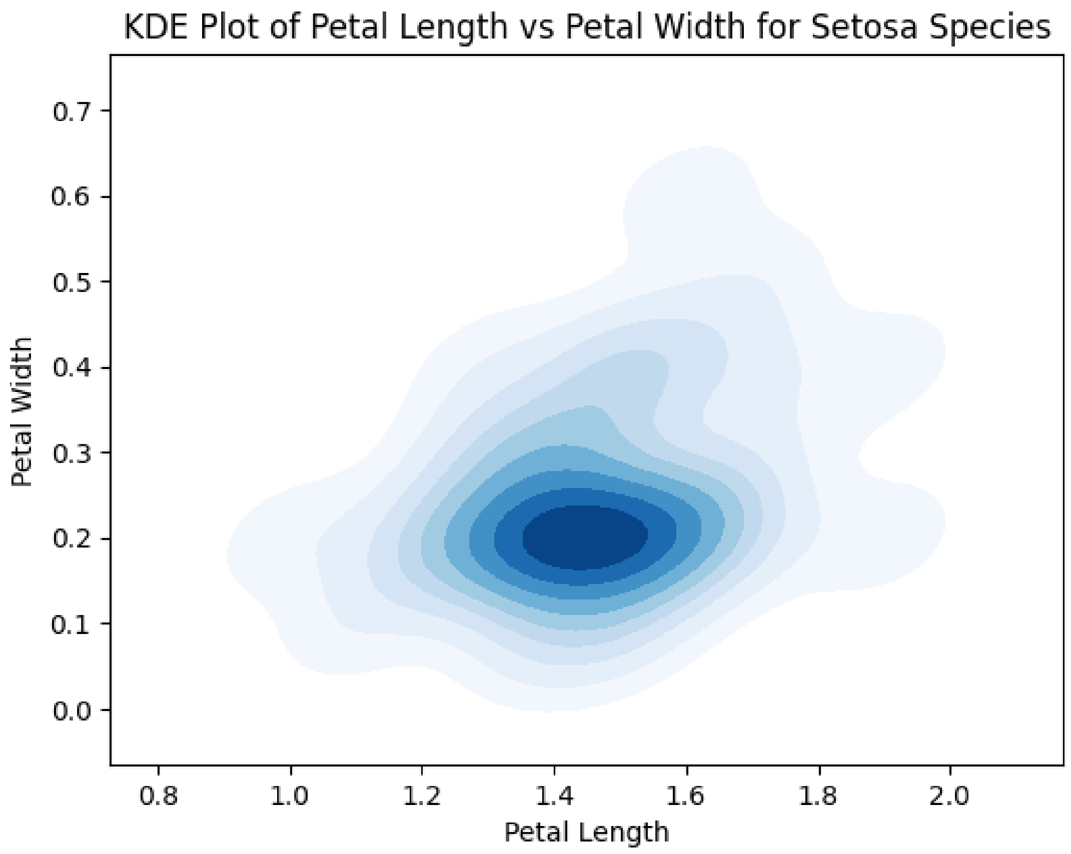
Labels: Added axis labels and a title for clarity.

7. KDE Plot for Setosa Species (Petal Length vs Petal Width): Python program using seaborn to create a KDE plot of petal length versus petal width for the setosa species.

[33]:
```
# Create a KDE plot for petal length vs petal width for Setosa species
sns.kdeplot(x='petal_length', y='petal_width', data=setosa, fill=True,␣
 ↪cmap='Blues')

# Add labels and title
plt.title("KDE Plot of Petal Length vs Petal Width for Setosa Species")
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")

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



This plot visually represents the density distribution of petal length and petal width for the Setosa species. ### Explanation: sns.kdeplot(): The kdeplot function generates a 2D KDE plot for petal length vs petal width for the Setosa species. The fill=True argument fills the contours, and cmap='Blues' uses shades of blue for the contour levels.

Labels and Title: Appropriate labels and a title are added for clarity.