week-8-a22126551047

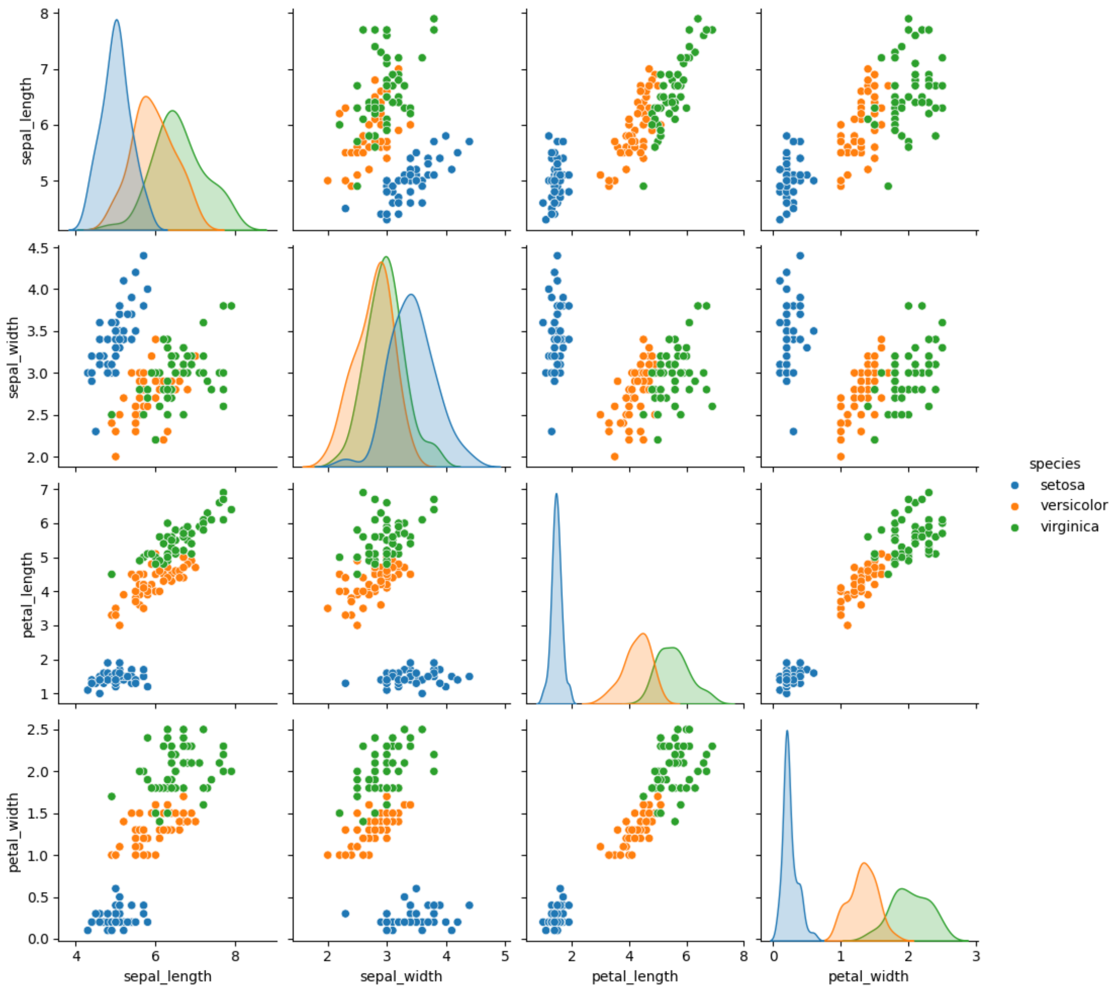
September 17, 2024

1. General Statistics Plot (Matplotlib or Seaborn):

Write a 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.

The first plot is a pairplot generated using the seaborn library. The pairplot displays pairwise relationships between the different numerical features (sepal length, sepal width, petal length, petal width) in the Iris dataset. By using the hue='species' parameter, we can color-code the three different species in the dataset, making it easier to observe how the species differ based on these attributes. This plot is useful for identifying correlations and clusters in the dataset.

[14]:
```python
# Importing the necessary libraries for plotting
import matplotlib.pyplot as plt
import seaborn as sns

# Loading the built-in 'iris' dataset from Seaborn
iris = sns.load_dataset('iris')

# Plotting pairwise relationships in the dataset with different species shown␣
 ↪in different colors
# 'hue' distinguishes the species of flowers, and 'height' sets the size of␣
 ↪each subplot
sns.pairplot(iris, hue='species', height=2.5)

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



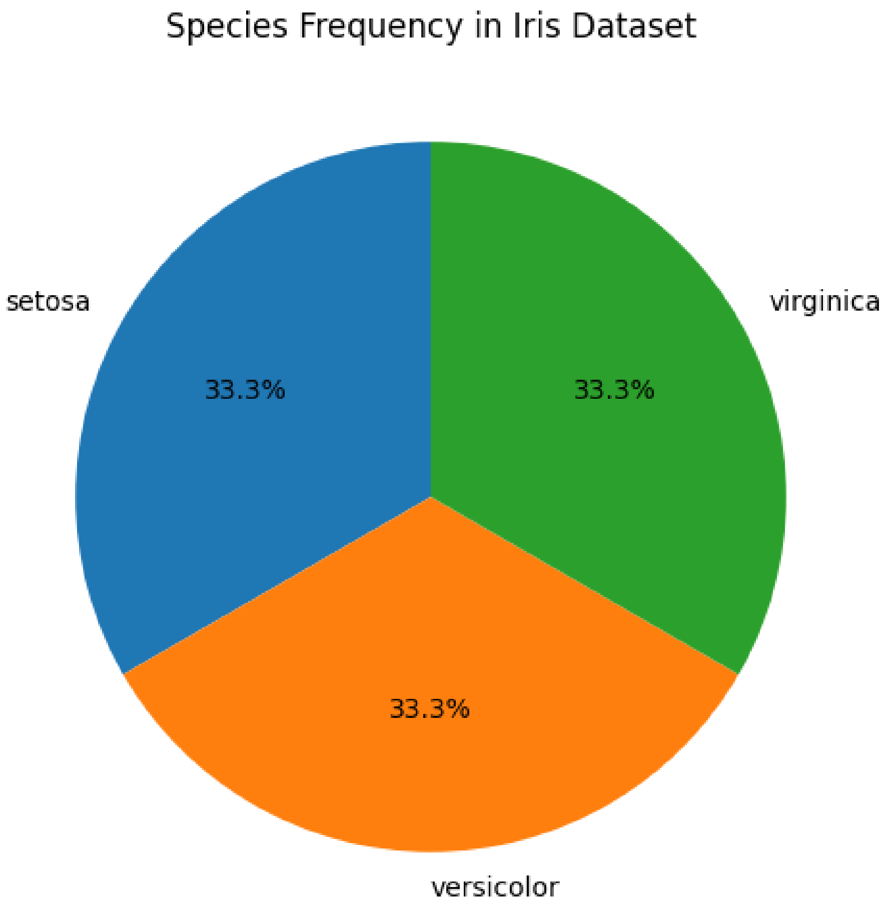
2. Pie Plot for Species Frequency:

Write a Python program to create a pie chart to display the frequency of the three species (setosa, versicolor, virginica) in the Iris dataset.

This code creates a pie chart that represents the frequency distribution of each species in the Iris dataset. By using the value_counts() function, the occurrences of each species are calculated, and the pie chart is plotted with percentage labels for each species. The pie chart visually shows the proportion of each species—Setosa, Versicolor, and Virginica—in the dataset.

[15]:
```
# Pie Plot for Species Frequency:

# Counting the occurrences of each species in the iris dataset
species_counts = iris['species'].value_counts()

# Creating a figure with a specific size (6x6 inches)
plt.figure(figsize=(6,6))
```

```
# Plotting a pie chart:
# - 'species_counts' is the data being plotted
# - 'labels' are set to the species names (from species_counts.index)
# - 'autopct' formats the percentage values on the chart
# - 'startangle' rotates the pie chart to start at 90 degrees (top)
plt.pie(species_counts, labels=species_counts.index, autopct="%1.1f%%",␣
 ↪startangle=90)

# Adding a title to the pie chart
plt.title('Species Frequency in Iris Dataset')

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

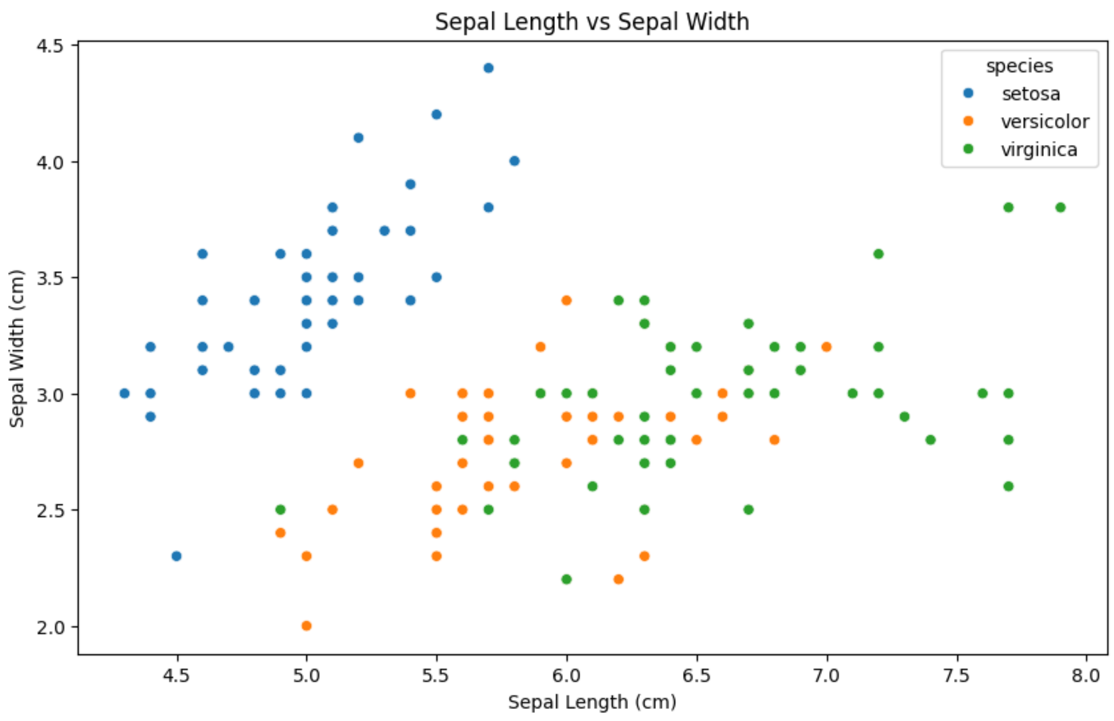


3. Relationship Between Sepal Length and Width:

Write a Python program to create a scatter plot to find the relationship between sepal length and sepal width for the Iris dataset.

A scatter plot is generated to visualize the relationship between sepal length and sepal width for all species in the dataset. Each point represents a sample, and different colors are used to distinguish between species. This plot helps in understanding how sepal measurements vary between the species and whether there are any visible patterns or trends between these two features.

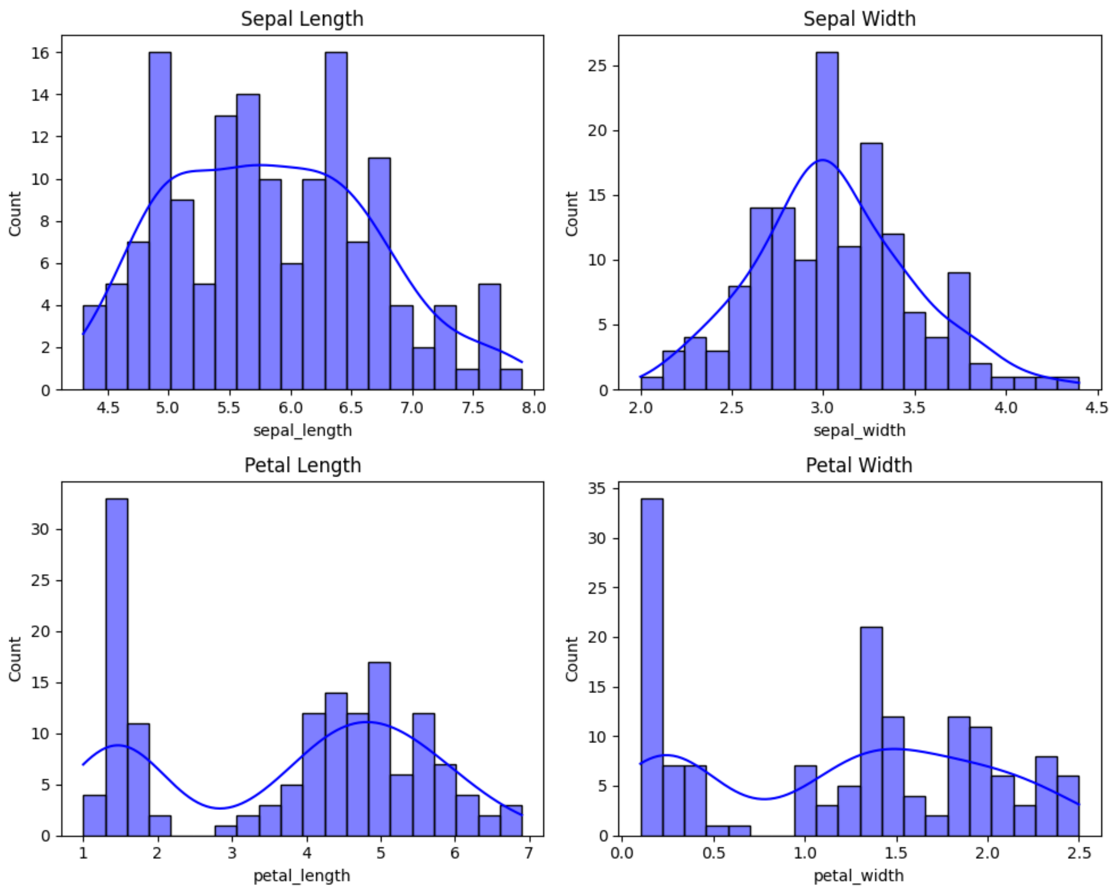
[16]:
```
# Relationship Between Sepal Length and Sepal Width:

# Creating a figure with a specific size (10x6 inches)
plt.figure(figsize=(10, 6))

# Plotting a scatter plot to show the relationship between sepal length and␣
 ↪sepal width:
# - 'x' and 'y' are set to 'sepal_length' and 'sepal_width' columns from the␣
 ↪dataset
# - 'hue' is used to color-code the species, allowing us to see how species␣
 ↪differ in terms of these measurements
# - 'data' refers to the iris dataset being used
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=iris)

# Adding a title to the scatter plot
plt.title('Sepal Length vs Sepal Width')

# Labeling the x-axis as 'Sepal Length (cm)'
plt.xlabel('Sepal Length (cm)')

# Labeling the y-axis as 'Sepal Width (cm)'
plt.ylabel('Sepal Width (cm)')

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



4. Distribution of Sepal and Petal Features: Write a 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.

This code creates a grid of histograms showing the distribution of each feature in the Iris dataset: sepal length, sepal width, petal length, and petal width. By using Kernel Density Estimation (KDE) overlaid on the histograms, the code provides a smooth estimate of the feature distributions. This allows for easier visualization of the shape and spread of the data. The tight_layout() function ensures the subplots are neatly organized with no overlap.

[17]:
```
# Distribution of Sepal and Petal Features:

# Defining the features (sepal and petal measurements) to plot
features = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']

# Corresponding titles for each subplot
titles = ['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width']

# Creating a 2x2 grid of subplots, with a figure size of 10x8 inches
fig, axes = plt.subplots(2, 2, figsize=(10, 8))

# Looping through each subplot, feature, and title to create individual␣
 ↪histograms
for ax, feature, title in zip(axes.flat, features, titles):
    # Plotting a histogram with a Kernel Density Estimate (KDE) for each feature
    # 'bins' sets the number of bins in the histogram
    # 'ax' refers to the current axis (subplot) being plotted
    # 'color' sets the color of the histogram to blue
    sns.histplot(iris[feature], kde=True, bins=20, ax=ax, color='blue')

    # Setting the title for each subplot
    ax.set_title(title)

# Adjusting the layout to prevent overlap between subplots
plt.tight_layout()

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

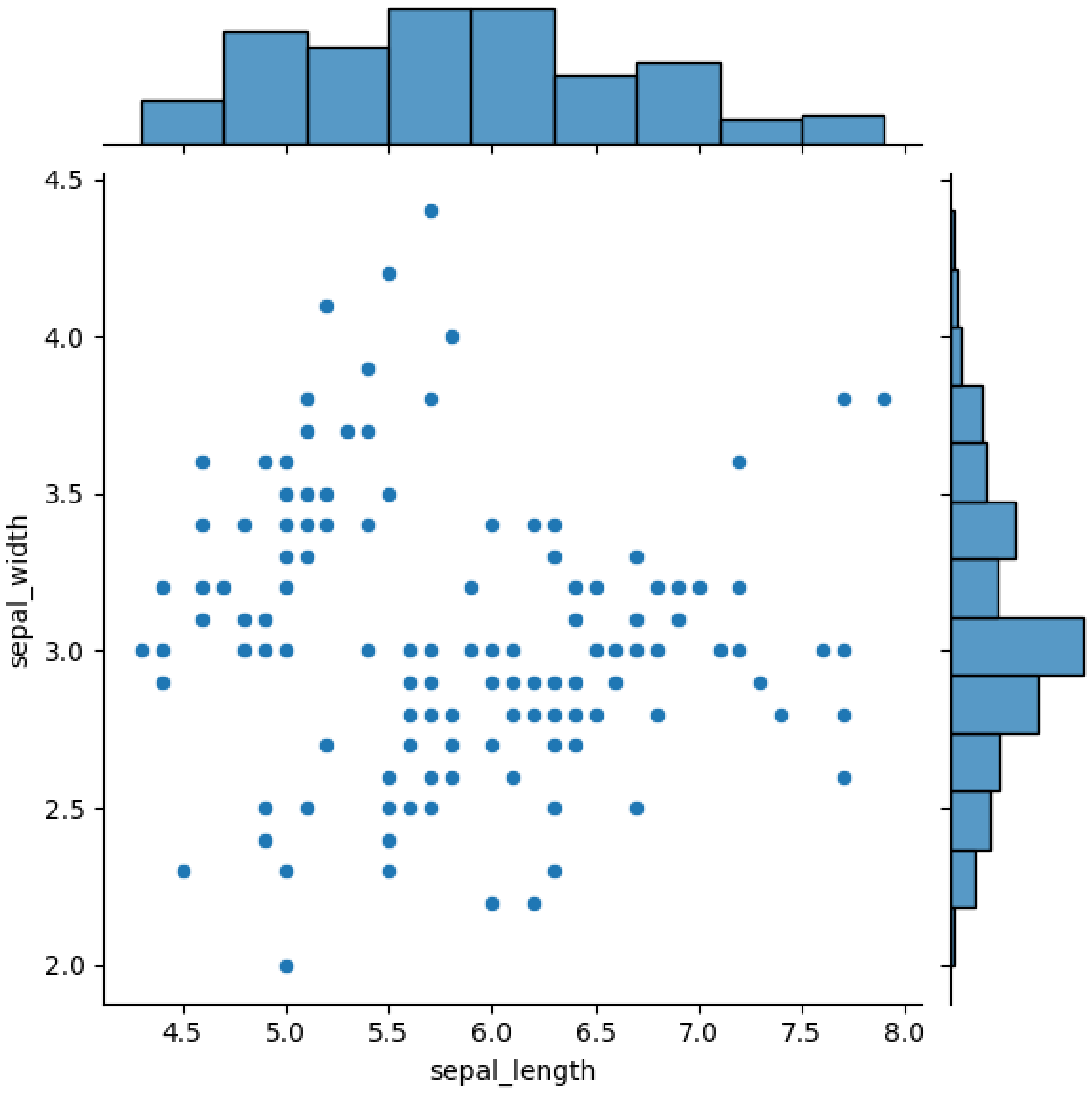


5 . Jointplot of Sepal Length vs Sepal Width:

Write a Python program to create a joint plot to describe the individual distributions on the same plot between sepal length and sepal width.

The jointplot shows the bivariate distribution between sepal length and sepal width, combining both a scatter plot and univariate histograms for each variable on the x and y axes. This helps to visualize both the relationship between the two features and their individual distributions, allowing us to observe potential correlations or trends in the data.

[18]:
```
# Jointplot of Sepal Length vs Sepal Width:

# Creating a joint plot to visualize the relationship between sepal length and␣
 ↪sepal width:
# - 'x' and 'y' are set to 'sepal_length' and 'sepal_width' respectively
# - 'data' refers to the iris dataset being used
# - 'kind' specifies the type of plot, in this case, a scatter plot
sns.jointplot(x='sepal_length', y='sepal_width', data=iris, kind='scatter')

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

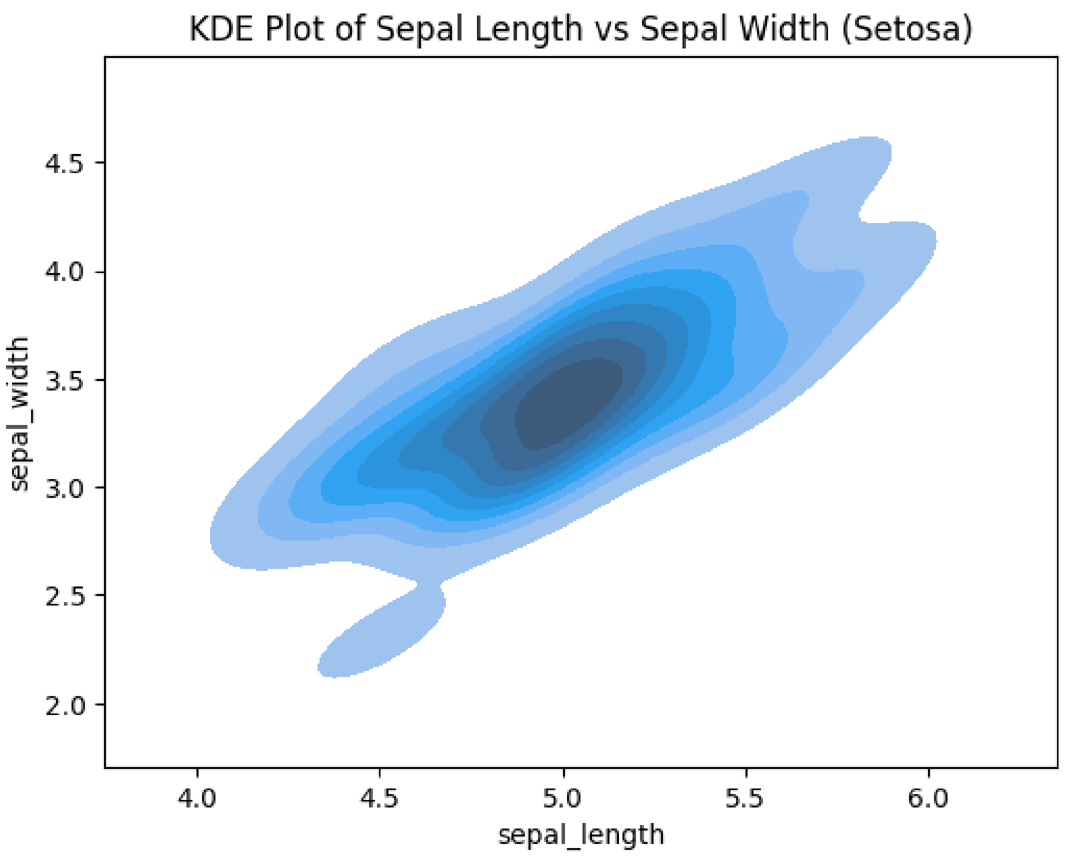


6 . KDE Plot for Setosa Species (Sepal Length vs Sepal Width): Write a Python program using seaborn to create a KDE (Kernel Density Estimate) plot of sepal length versus sepal width for the setosa species of the Iris dataset.

This code generates a Kernel Density Estimate (KDE) plot for the Setosa species, visualizing the density of sepal length versus sepal width. The KDE plot highlights areas of higher density in the feature space, showing where most of the Setosa samples are concentrated. This kind of plot is particularly useful for observing how the features are distributed for a specific species.

[19]:
```
# KDE Plot for Setosa Species (Sepal Length vs Sepal Width):

# Filtering the iris dataset to only include rows where the species is 'setosa'
setosa = iris[iris['species'] == 'setosa']

# Creating a Kernel Density Estimate (KDE) plot to visualize the distribution␣
 ↪of sepal length and sepal width for the Setosa species:
# - 'x' and 'y' are set to 'sepal_length' and 'sepal_width' respectively
# - 'data' refers to the filtered Setosa dataset
# - 'shade=True' fills the KDE plot to make the density areas shaded
sns.kdeplot(x='sepal_length', y='sepal_width', data=setosa, shade=True)

# Adding a title to the KDE plot
plt.title('KDE Plot of Sepal Length vs Sepal Width (Setosa)')

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

```
<ipython-input-19-4fe27e34cba0>:10: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(x='sepal_length', y='sepal_width', data=setosa, shade=True)
```



7 . KDE Plot for Setosa Species (Petal Length vs Petal Width):

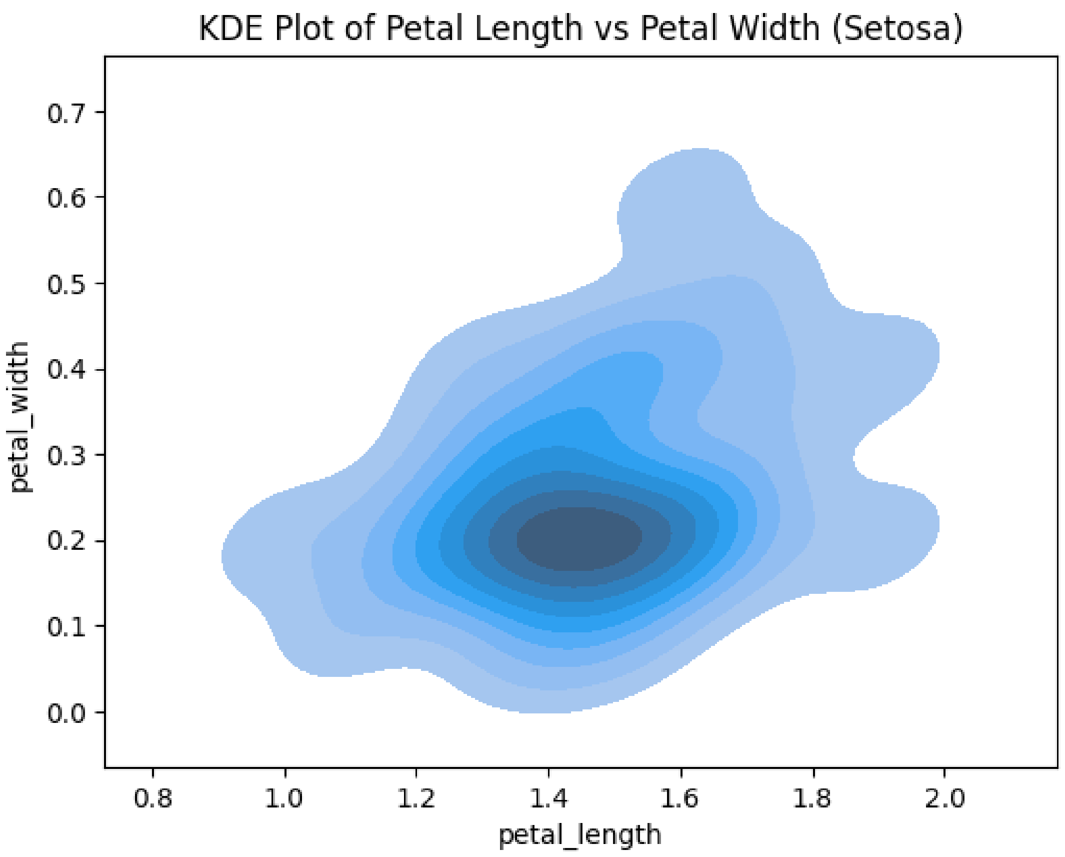
Write a Python program using seaborn to create a KDE plot of petal length versus petal width for the setosa species

Similar to the previous KDE plot, this plot focuses on the petal length and petal width for the Setosa species. It shows the density distribution for these features, helping us to understand the areas where the petal measurements of Setosa are more concentrated. This plot is useful for comparing the feature distributions of different species.

[20]:
```
# KDE Plot for Setosa Species (Petal Length vs Petal Width):

# Creating a Kernel Density Estimate (KDE) plot to visualize the distribution␣
 ↪of petal length and petal width for the Setosa species:
# - 'x' and 'y' are set to 'petal_length' and 'petal_width' respectively
# - 'data' refers to the filtered Setosa dataset
# - 'shade=True' fills the KDE plot to shade the areas of higher density
sns.kdeplot(x='petal_length', y='petal_width', data=setosa, shade=True)

# Adding a title to the KDE plot
plt.title('KDE Plot of Petal Length vs Petal Width (Setosa)')
```

```python
# Displaying the plot
plt.show()
```

```
<ipython-input-20-9802786da1f2>:7: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(x='petal_length', y='petal_width', data=setosa, shade=True)
```


KDE Plot of Petal Length vs Petal Width (Setosa)
petal_width
petal_length