dv-with-iris-1

September 17, 2024

0.1 Data Visualization with Matplotlib and Seaborn using the Iris Dataset

- Exploring the data visualization techniques using Python's matplotlib and seaborn libraries.

0.1.1 Introduction to Matplotlib :

->Matplotlib is one of the most popular plotting libraries in Python, used for creating static, interactive, and animated visualizations.

->It provides a flexible way to generate various types of plots and charts, making it an essential tool for data analysis and visualization.

->Basic Syntax: To use Matplotlib, first import the pyplot module, which provides a MATLAB-like interface for creating plots:

[62]:
```
import matplotlib.pyplot as plt
```

->Matplotlib can produce a variety of plots such as

1.Line Plot: plot()

2.Scatter Plot: scatter()

3.Bar Chart: bar()

4.Histogram: hist()

5.Heatmap: imshow() etc.

->Matplotlib has extensive customization options like

~Colors & Markers: Set colors, line styles, and markers in plot functions.

~Labels & Titles: Customize with set_xlabel(), set_ylabel(), set_title().

~Legends: Use ax.legend() to add legend.

0.1.2 Introduction to Seaborn:

->Seaborn is a powerful Python library built on top of Matplotlib that simplifies creating attractive and informative statistical graphics.

->It is particularly well-suited for visualizing complex datasets and understanding statistical relationships.

-> Seaborn works seamlessly with pandas data structures and simplifies the process of creating complex visualizations with just a few lines of code.

->Basic Syntax: You can start using Seaborn by importing it as

[63]:
```
import seaborn as sns
```

->Plot Types in Seaborn:

1.Distribution Plots: sns.histplot(), sns.kdeplot()

2.Categorical Plots: sns.boxplot(), sns.violinplot(), sns.barplot()

3.Regression Plots: sns.regplot(), sns.lmplot()

4.Pairwise Plots: sns.pairplot()

5.Heatmaps: sns.heatmap()etc.

->Seaborn has Customization options like:

~Themes: sns.set_theme() for global styles.

~Color Palettes: Customizable with sns.color_palette() and sns.set_palette().

0.1.3 Iris dataset:

->The Iris dataset is a classic dataset used in statistics and machine learning, particularly for classification problems.

->The Iris dataset consists of 150 observations from 3 species of iris flowers (setosa, versicolor, and virginica).

-> Each observation contains the following features:

- sepal length (cm)
- sepal width (cm)
- petal length (cm)
- petal width (cm)

Loading the Dataset: The Iris dataset is available in various libraries like Seaborn and Scikit-learn. Here's how you can load it using these libraries:

->Using Seaborn:

iris=sns.load_dataset('iris')

->Using Scikit-learn:

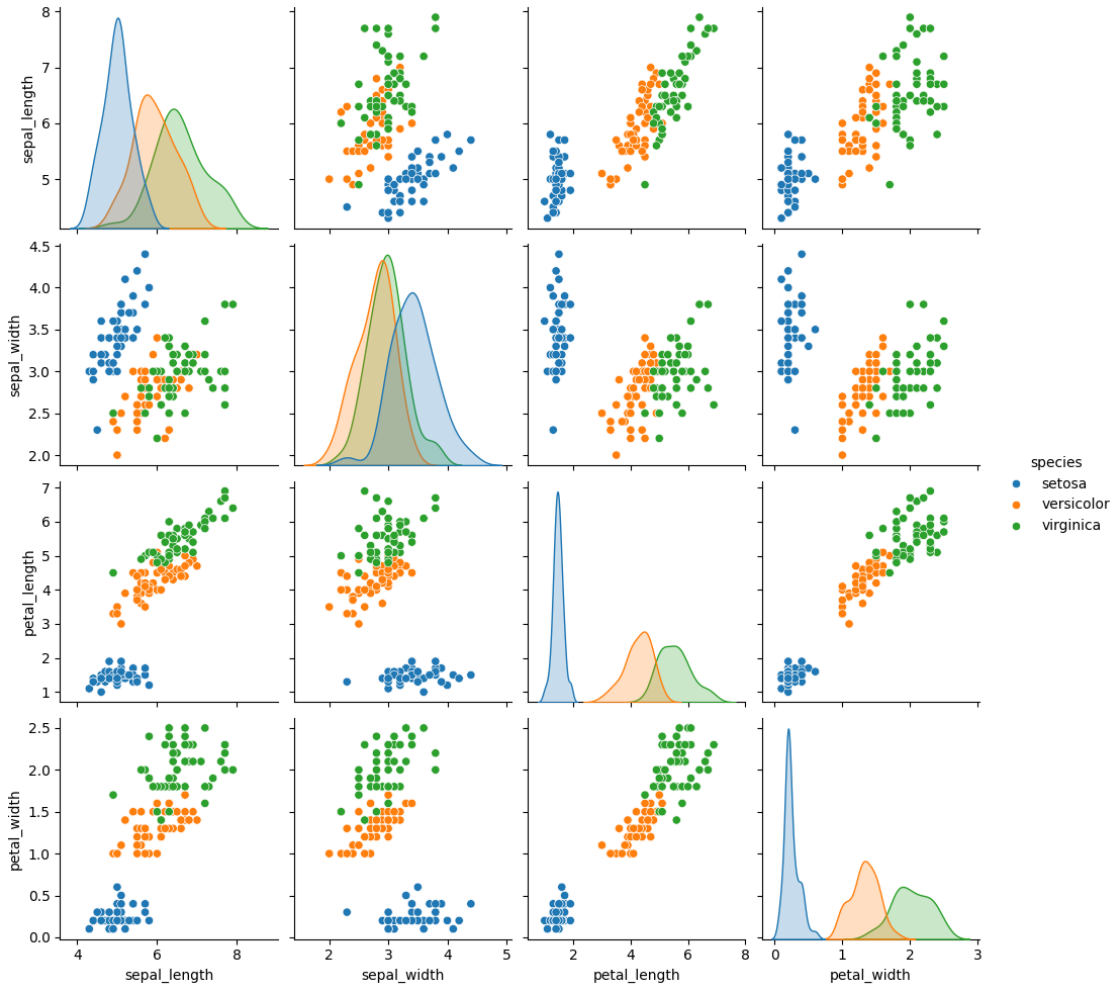
from sklearn.datasets import load_iris

iris = load_iris()

[64]:
```
# Load Iris dataset
iris=sns.load_dataset('iris')
```

1. General Statistics Plot (Matplotlib or Seaborn):

[54]:
```
# Method 1: Creating a plot Using pairplot
sns.pairplot(iris, hue='species', height=2.5)
plt.show()
```



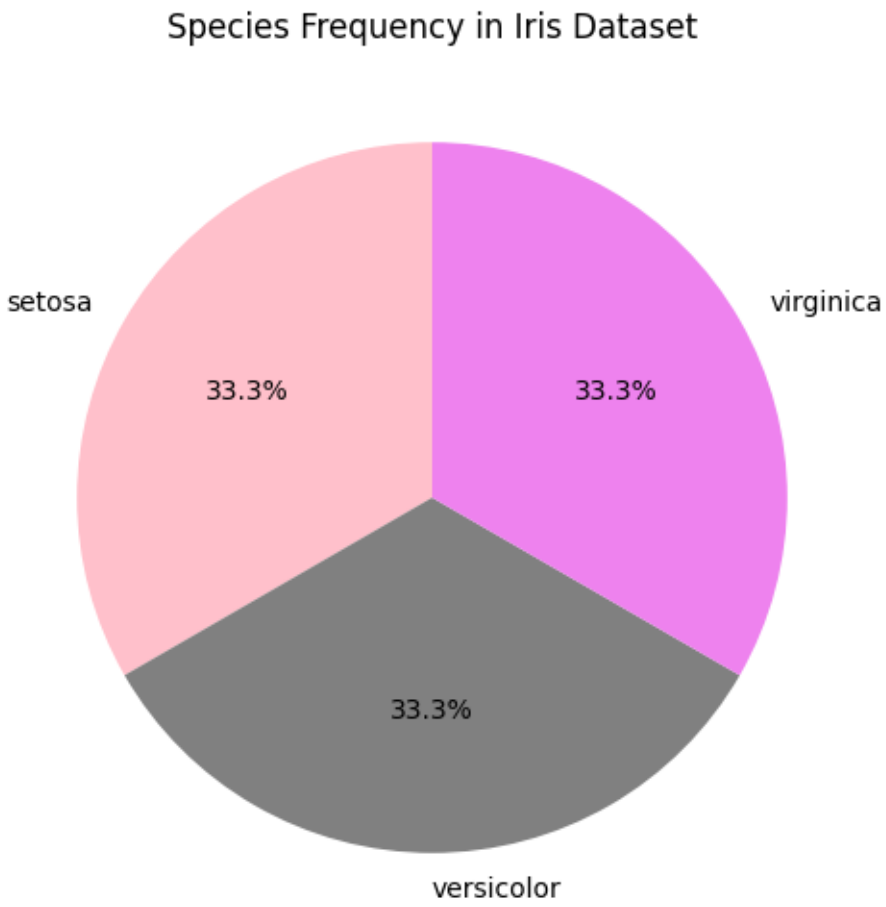
[55]:
```
# Method 2: creating a plot Using pandas'describe()
summary = iris.describe()
print("Statistical Summary of the Iris Dataset:")
print(summary)
```

```
Statistical Summary of the Iris Dataset:
       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

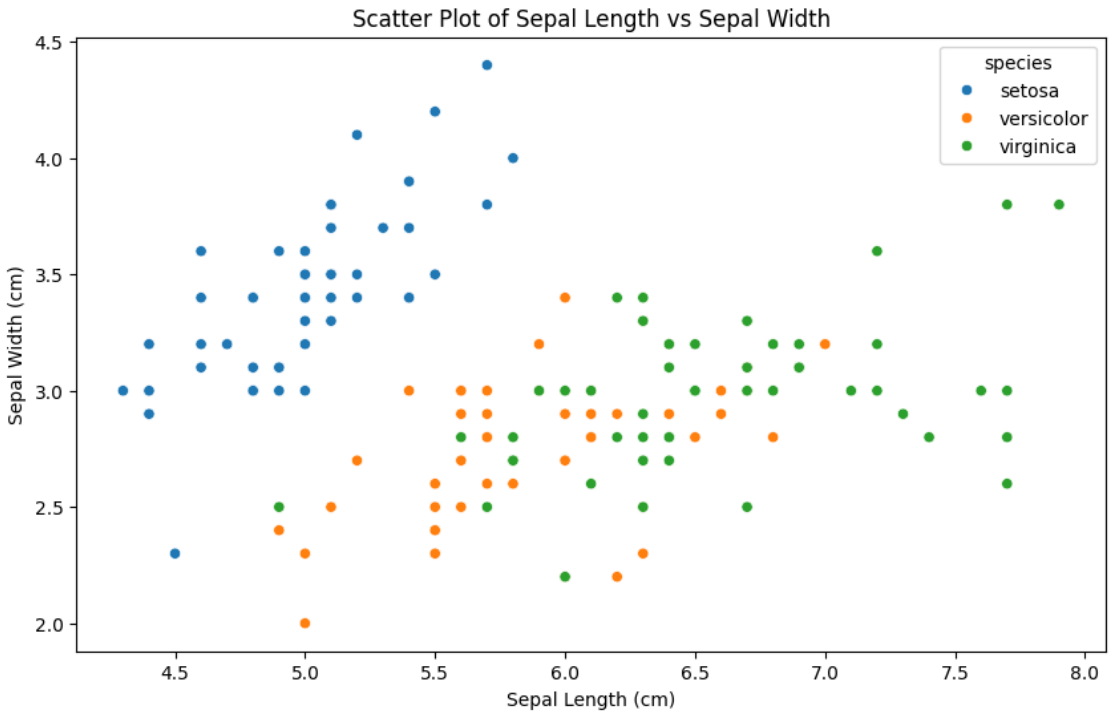
2. Pie Plot for Species Frequency:

[65]:
```
#Species frequency
species_counts = iris['species'].value_counts()
plt.figure(figsize=(6,6))
#plotting a pie plot
plt.pie(species_counts, labels=species_counts.index, autopct='%1.1f%%',␣
 ↪startangle=90,colors=['pink','grey','violet'])
plt.title('Species Frequency in Iris Dataset')#Adding title
plt.show()
```



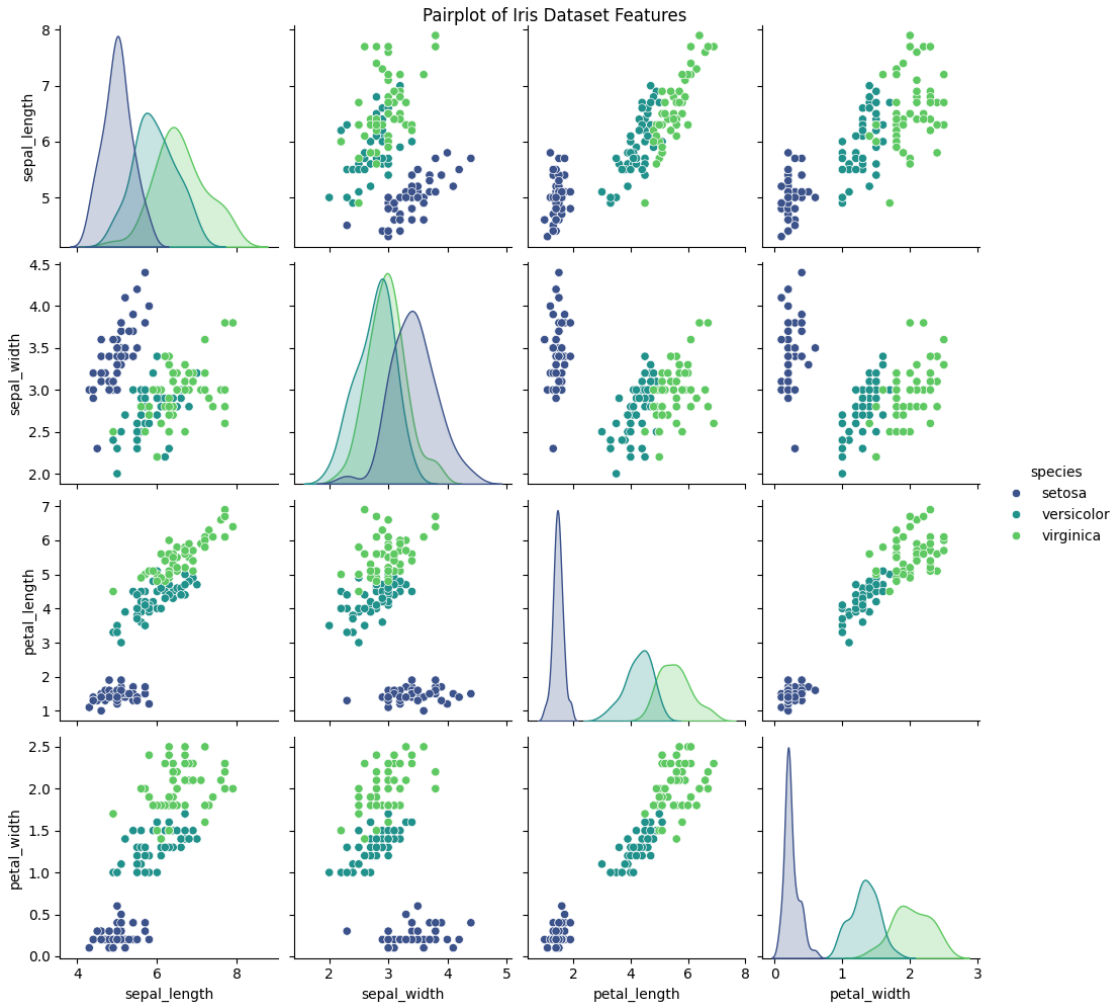
3. Relationship Between Sepal Length and Sepal Width:

[66]:
```
#Creating a scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=iris)
# Adding titles and labels
plt.title('Scatter Plot of Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
# Show the plot as
plt.show()
```



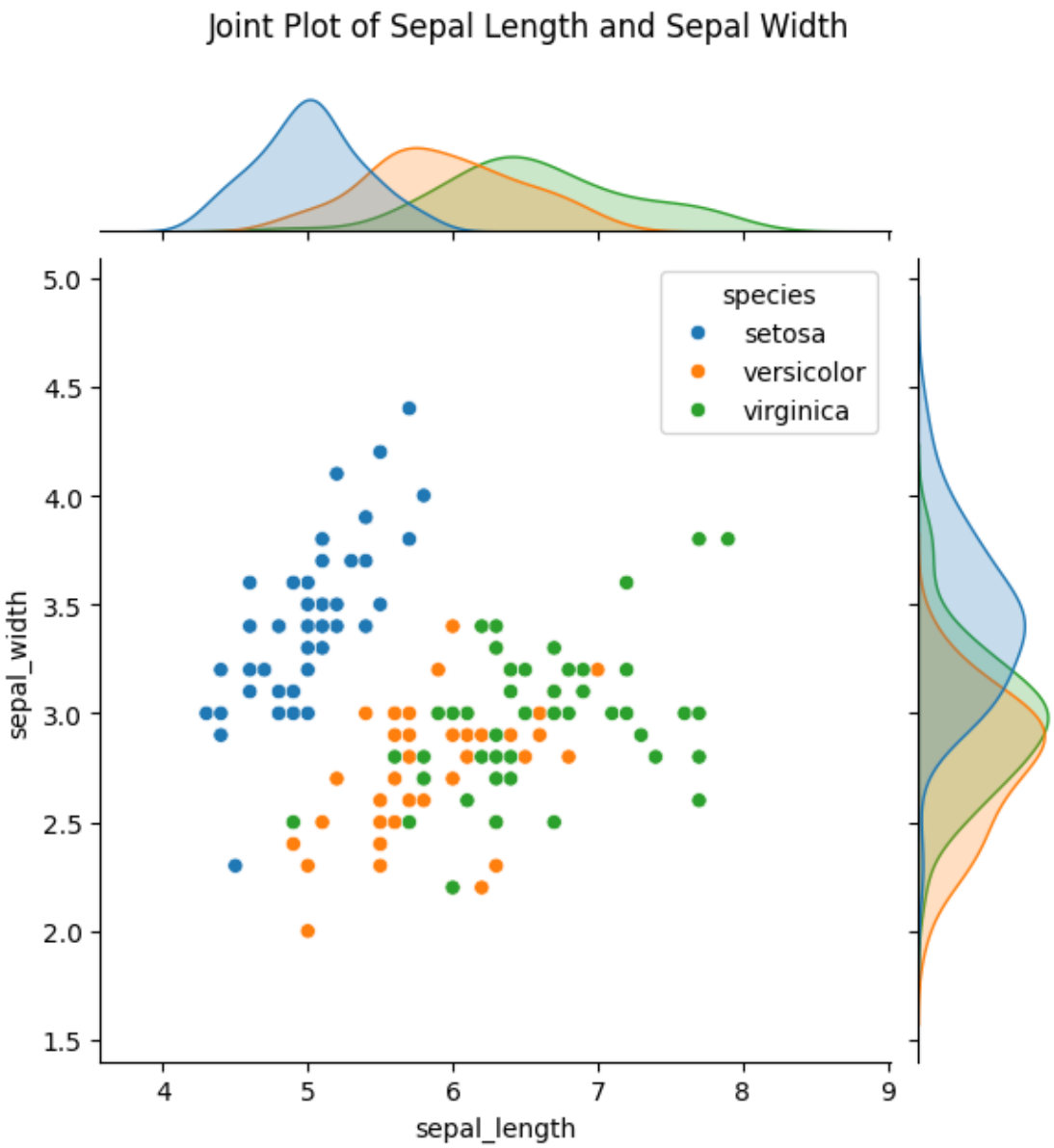
4. Distribution of Sepal and Petal Features:

[67]:
```
#Creating a pairplot
sns.pairplot(iris, hue='species', palette='viridis')#Adding palette to specify␣
↪the color for visualizations
plt.suptitle('Pairplot of Iris Dataset Features', y=1)  # Title with padding
plt.show()
```



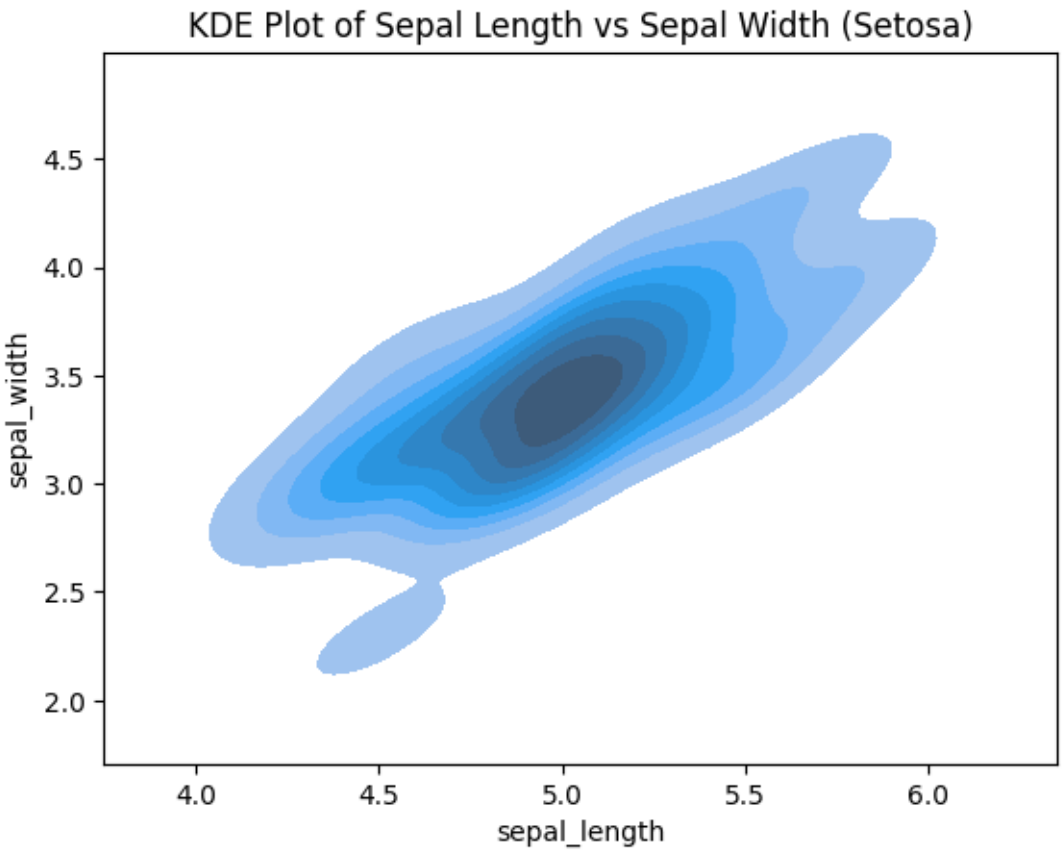
5. Jointplot of Sepal Length vs Sepal Width:

[59]:
```
#Creating a jointplot
jointplot=sns.jointplot(x='sepal_length', y='sepal_width', data=iris,␣
 ↪kind='scatter', hue='species')
# Adding titles to the joint plot
jointplot.fig.suptitle('Joint Plot of Sepal Length and Sepal Width',y=1.05)
plt.show()
```



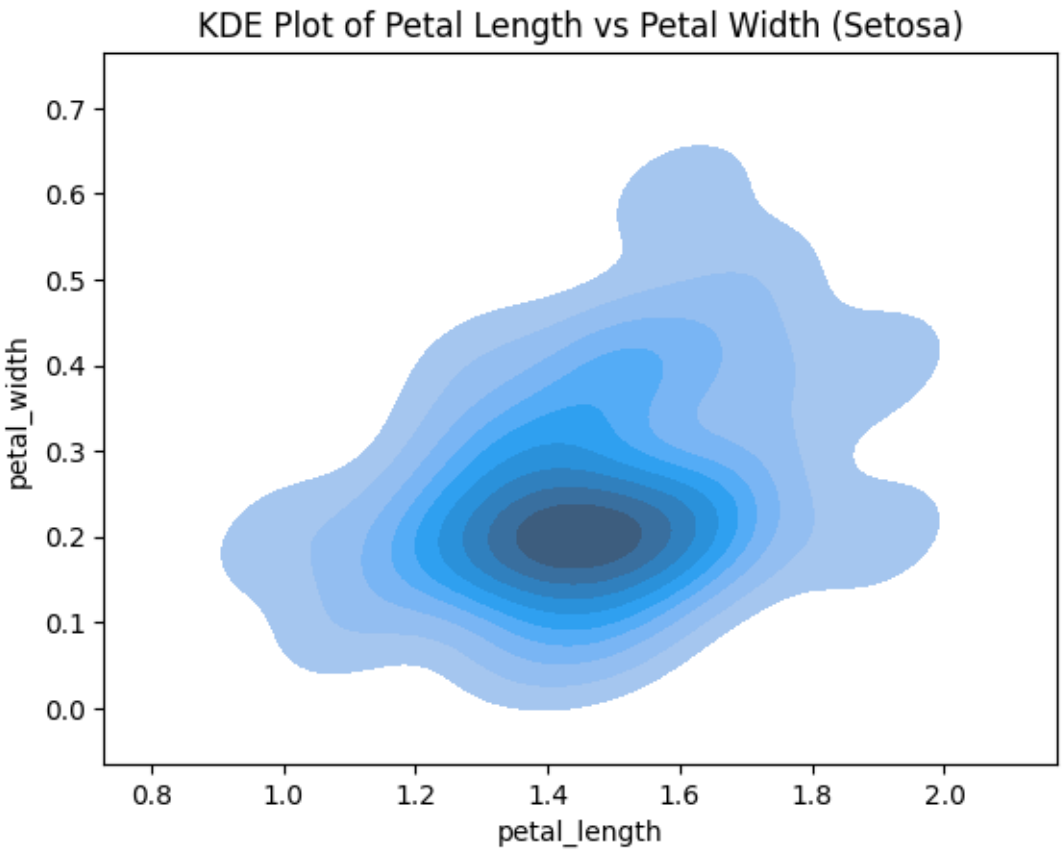
6. KDE Plot for Setosa Species (Sepal Length vs Sepal Width):

[68]:
```
#Creating a KDE plot
setosa = iris[iris['species'] == 'setosa']
sns.kdeplot(x='sepal_length', y='sepal_width', data=setosa,fill='True')
plt.title('KDE Plot of Sepal Length vs Sepal Width (Setosa)')#Adding title
plt.show()
```



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

[69]:
```
#Creating a KDE Plot
sns.kdeplot(x='petal_length', y='petal_width', data=setosa,fill=True)
plt.title('KDE Plot of Petal Length vs Petal Width (Setosa)')#Adding title
plt.show()
```


KDE Plot of Petal Length vs Petal Width (Setosa)
petal_width
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
1.2
1.4
1.6
1.8
2.0
petal_length