irisdata

September 17, 2024

Matplotlib is a comprehensive plotting library for creating static, interactive, and animated visualizations in Python. It provides a wide range of plotting options including line plots, scatter plots, bar charts, histograms, and more.

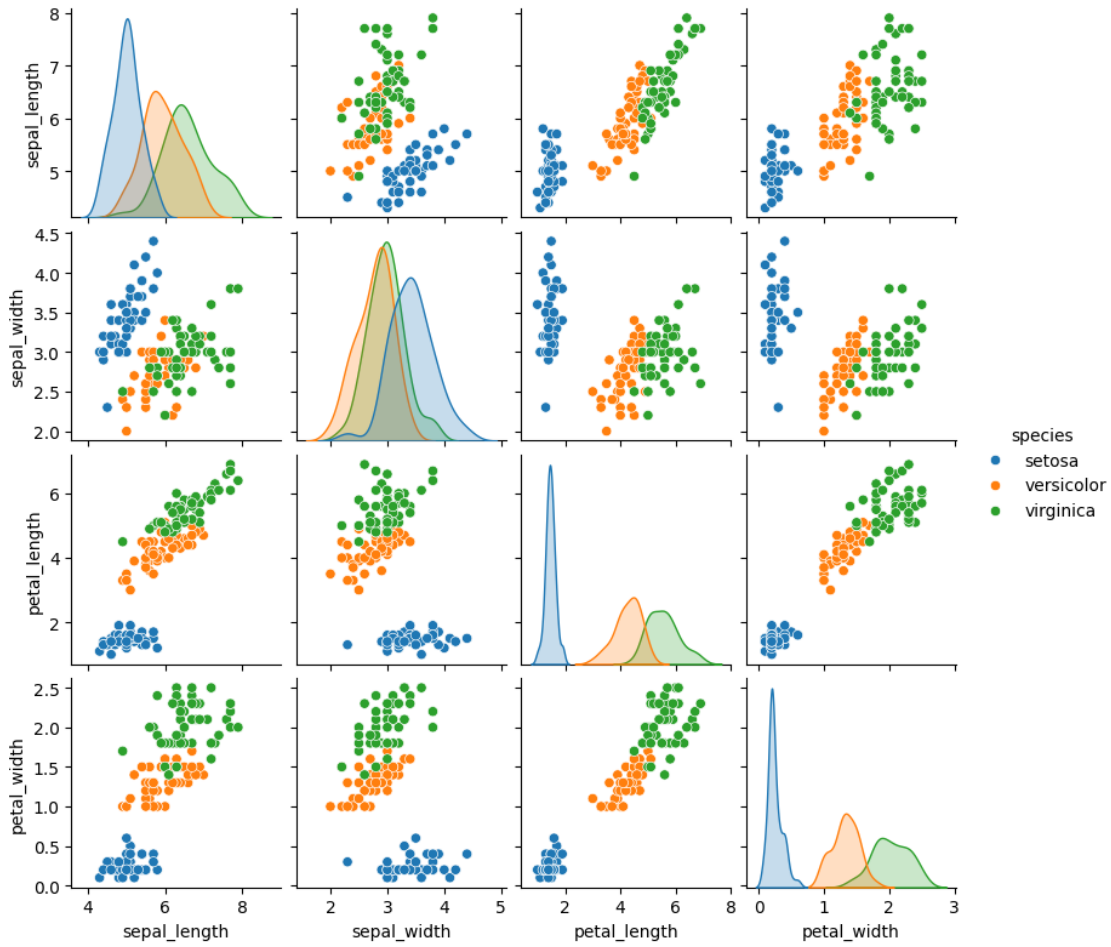
Seaborn is a Python library for creating attractive, informative statistical graphics with ease. Built on Matplotlib, it simplifies the creation of complex plots like scatter plots, histograms, and heatmaps, and integrates seamlessly with Pandas DataFrames.

[2]:
```
import seaborn as sns
import matplotlib.pyplot as plt
#load the data set
iris=sns.load_dataset('iris')
```

General Statistics Plot (Matplotlib or Seaborn):

A pair plot is a great way to visualize relationships between multiple variables in a dataset. It creates a matrix of scatter plots for each pair of variables, along with histograms or density plots for each variable on the diagonal.

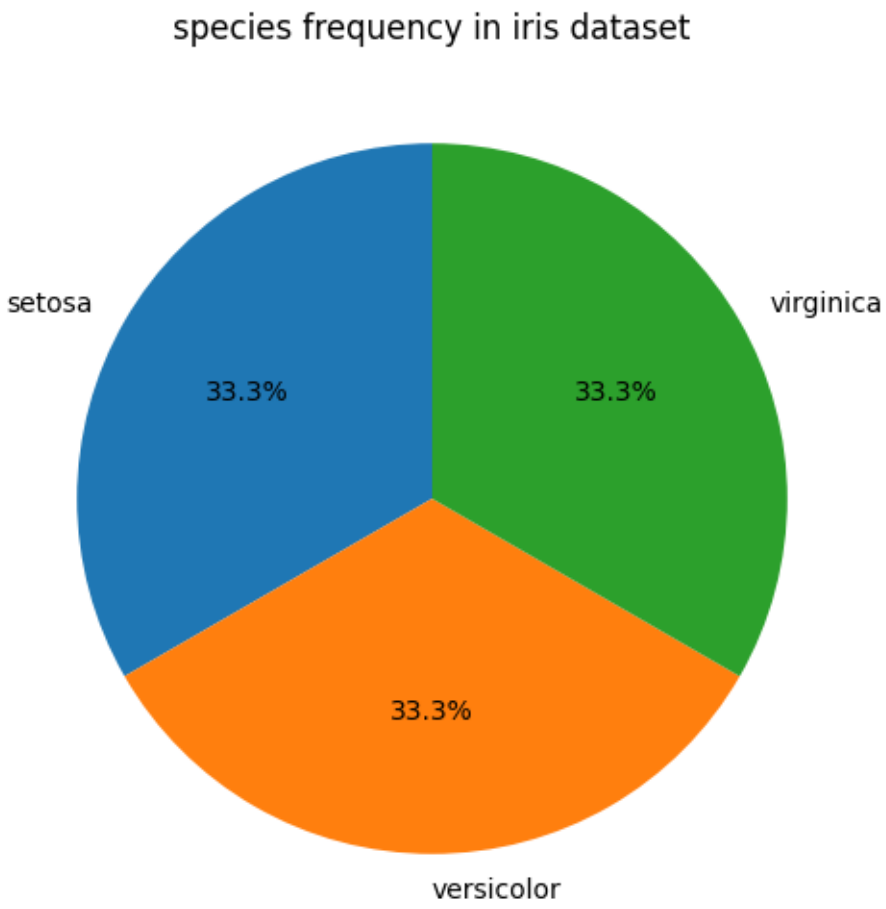
[ ]:
```
# Creates a pairplot to visualize relationships between variables in the iris␣
 ↪dataset.
sns.pairplot(iris,hue='species',height=2)# hue='species' colors the points␣
 ↪based on the species.
# height=2 sets the height of each subplot
plt.show()
```



Pie Plot for Species Frequency:

A pie plot is a circular chart that shows the proportions of different categories as slices of a pie. Using matplotlib, you can create a pie plot by calling plt.pie(), where you pass in the data to be plotted and optionally specify labels, colors, and other parameters. Each slice of the pie represents a category's share of the total, making it easy to visualize the percentage distribution of the data.

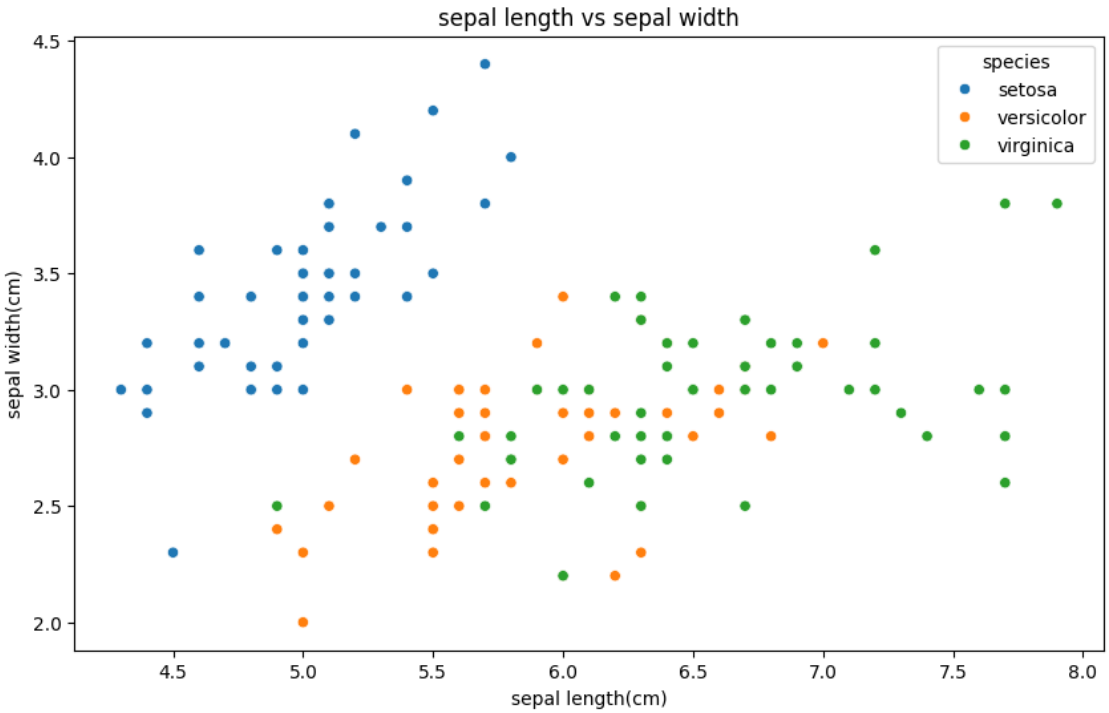
[ ]:
```
# Calculate the frequency of each species in the 'species' column.
species_counts=iris['species'].value_counts()
plt.figure(figsize=(6,6))
# species_counts provides the data.
# labels are set to the index of species_counts (species names).
# autopct formats the percentage displayed on each slice.
# startangle sets the starting angle of the first slice.
plt.pie(species_counts,labels=species_counts.index,autopct='%1.
  ↪1f%%',startangle=90)
plt.title('species frequency in iris dataset')
plt.show()
```



Relationship Between Sepal Length and Width:

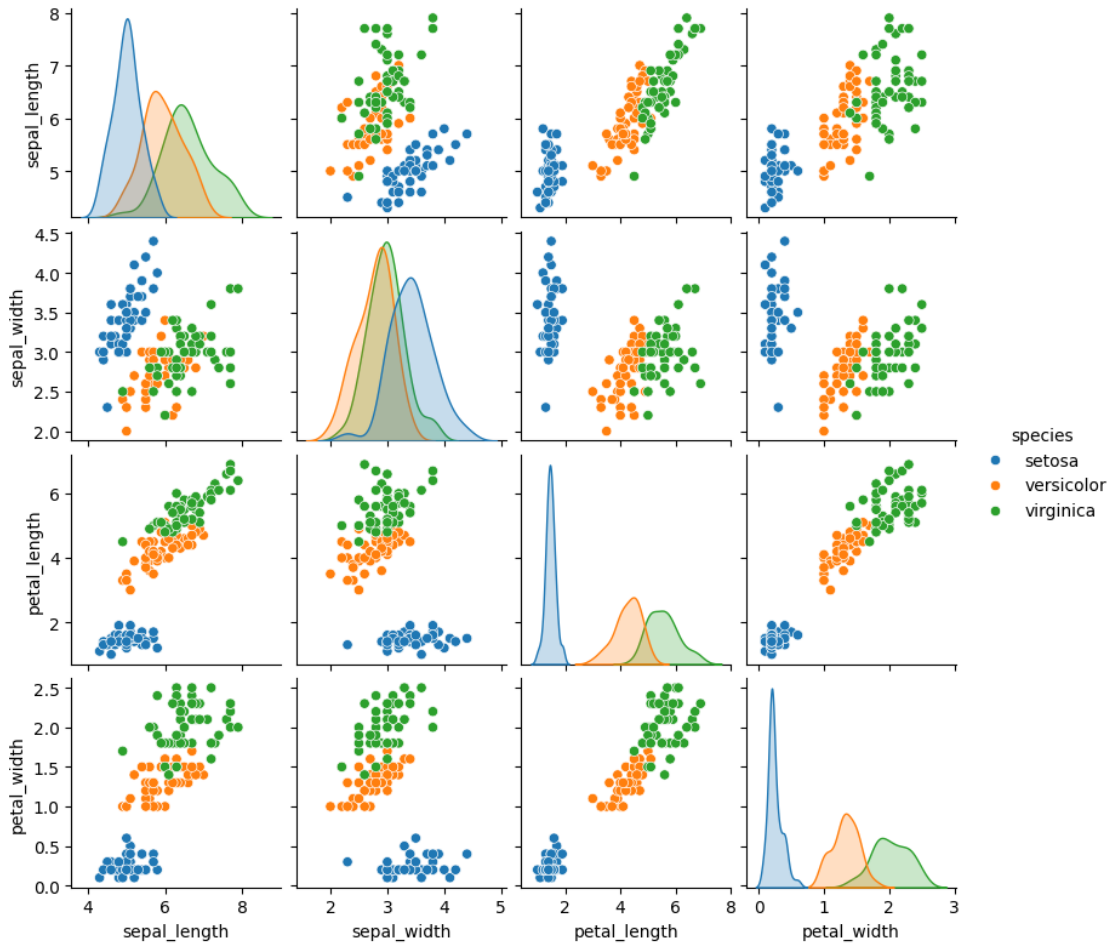
A scatter plot in Python is used to visualize the relationship between two numerical variables by plotting individual data points on a Cartesian plane. Using the matplotlib library, you can create a scatter plot with plt.scatter(), where you provide the x and y coordinates of the data points. Each point represents an observation in the dataset, allowing you to see trends, correlations, and outliers.

[ ]:
```
plt.figure(figsize=(10,6))
# Create a scatter plot using Seaborn.
# x and y specify the columns for the x and y axes.
# hue colors the points based on the 'species' column.
# data specifies the dataset to use
sns.scatterplot(x='sepal_length',y='sepal_width',hue='species',data=iris)
plt.title('sepal length vs sepal width')
plt.xlabel('sepal length(cm)')
plt.ylabel('sepal width(cm)')
plt.show()
```



Distribution of Sepal and Petal Features:

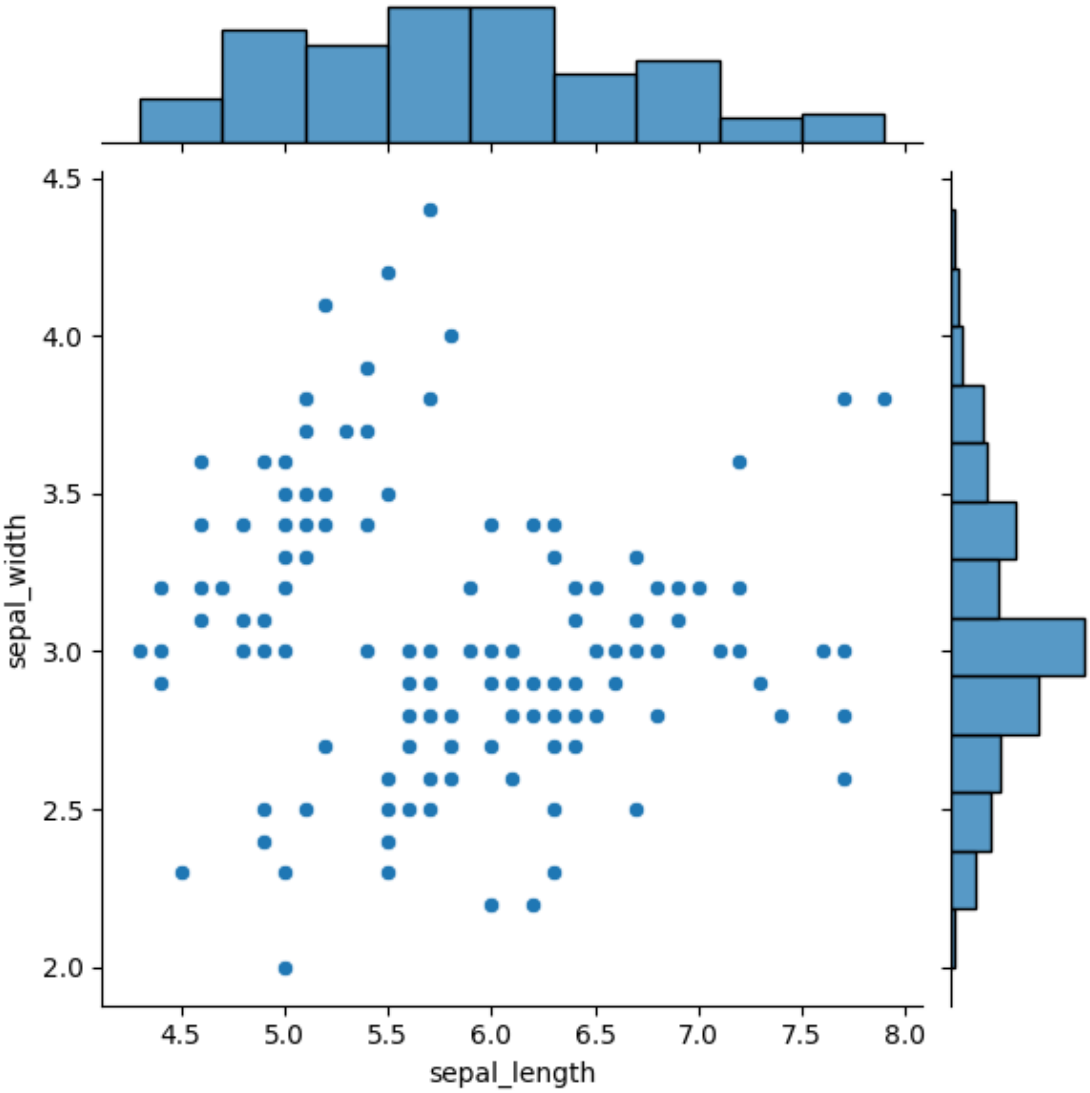
[ ]:
```
sns.pairplot(iris,hue='species',height=2)
plt.show()
```



Jointplot of Sepal Length vs Sepal Width:

A joint plot in Python combines a scatter plot and histograms (or KDE plots) to show the relationship between two numerical variables and their distributions. Using seaborn, you can create a joint plot with sns.jointplot(), which provides a comprehensive view of how two variables are correlated and how each variable is distributed.

[3]:
```
# Creates a jointplot to display the relationship between sepal length and␣
 ↪sepal width.
# x and y specify the columns for the x and y axes.
# data specifies the dataset to use.
# kind='scatter' indicates a scatter plot.
sns.jointplot(x='sepal_length',y='sepal_width',data=iris,kind='scatter')
plt.show()
```



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

A Kernel Density Estimate (KDE) plot in Python is used to visualize the probability density of a continuous variable. It provides a smooth estimate of the data's distribution, which can be more informative than histograms, especially for understanding the underlying distribution shape. Using seaborn, you can create a KDE plot with sns.kdeplot().

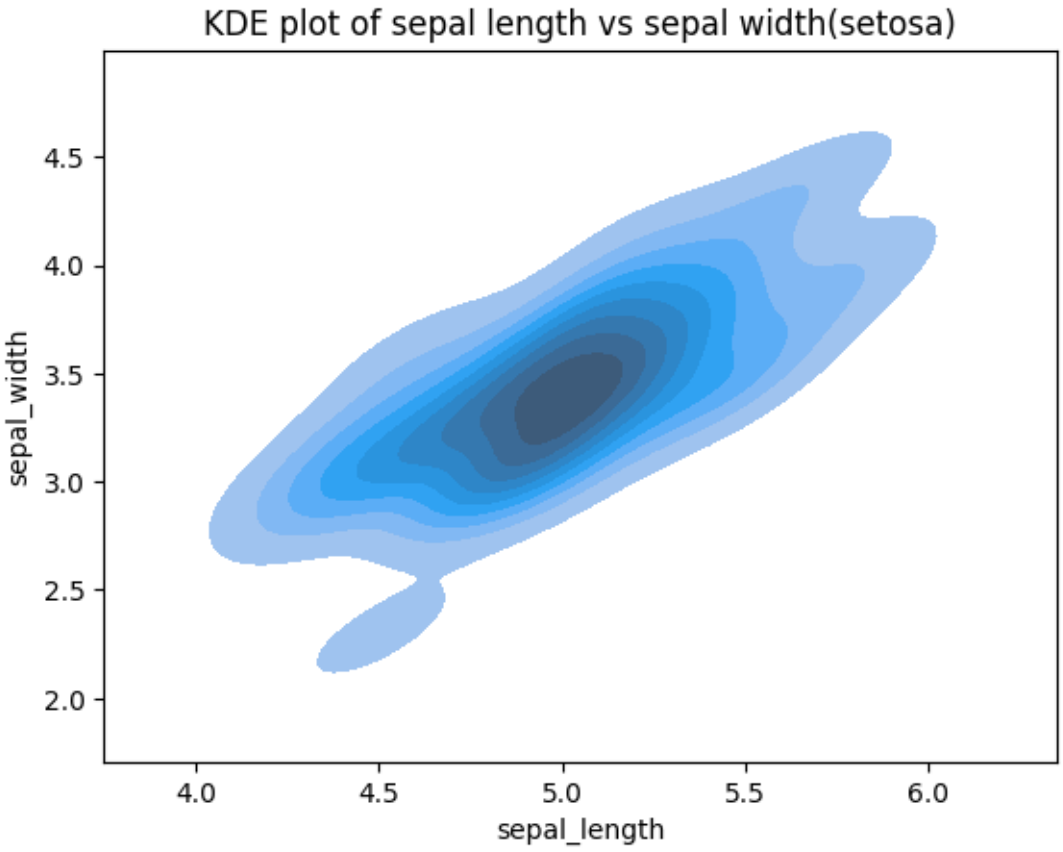
[5]:
```
setosa=iris[iris['species']=='setosa']
# Create a KDE plot using Seaborn.
# x and y specify the columns for the x and y axes.
# data specifies the dataset to use.
# shade=True fills the area under the KDE curve.
# Note: shade=true should be shade=True
sns.kdeplot(x='sepal_length',y='sepal_width',data=setosa,shade=True)
```

```
plt.title('KDE plot of sepal length vs sepal width(setosa)')
plt.show()
```

```
<ipython-input-5-24c162584b07>: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='sepal_length',y='sepal_width',data=setosa,shade=True)
```



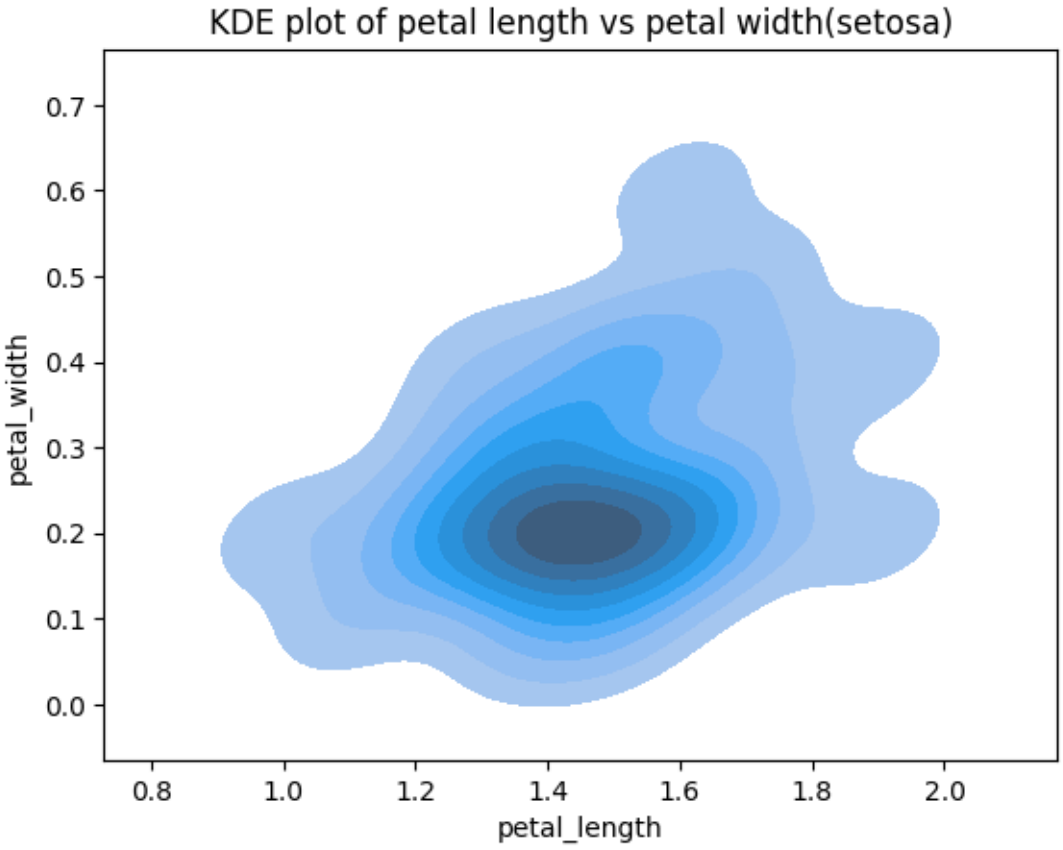
KDE Plot for Setosa Species (Petal Length vs Petal Width):

[6]:
```
# x and y specify the columns for the x and y axes.
# data specifies the dataset to use.
# shade=True fills the area under the KDE curve.
# Note: shade=true should be shade=True
sns.kdeplot(x='petal_length',y='petal_width',data=setosa,shade=True)
plt.title('KDE plot of petal length vs petal width(setosa)')
plt.show()
```

```
<ipython-input-6-2681c6ab530c>:5: 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)
```



[ ]: