DATA VISUALIZATION OF IRIS DATA IN PYTHON

Matplotlib:

Matplotlib is a widely-used library that provides a range of functionalities to create static, animated, and interactive plots. It's very flexible and customizable, making it suitable for creating a variety of different plots.

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

Syntax: import matplotlib.pyplot as plt

Seaborn:

Seaborn is built on top of Matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics. It integrates well with Pandas DataFrames and is often used for visualizing complex datasets.

You can start using Seaborn by importing it as:

Syntax: import seaborn as sns

The following shows the installation of matplotlib and seaborn package

In [2]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [3]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
```

In [4]:
```
#importing pyplot module from matplotlib library
import matplotlib.pyplot as plt
```

In [5]:
```
#inporting seaborn package
import seaborn as sns
```

In [6]:
```
import numpy as np
```

In [86]:
```
import pandas as pd
```

In [7]:
```
#loading iris dataset using seaboarn library into iris_data variable
iris_data=sns.load_dataset('iris')
```

In [8]:
```
iris_data
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

1. General stastics plot using matplotlib and seaborn

The pandas.DataFrame.describe() function in Python provides descriptive statistics of the numerical columns in a DataFrame by default. It can be a quick way to understand the central tendency, variability, and shape of the data distribution.

In [90]:
```
#summary stastics of iris data using pandas
iris=pd.DataFrame(iris_data)
print('Stastical Summary of Iris Data:')
print(iris.describe())
```

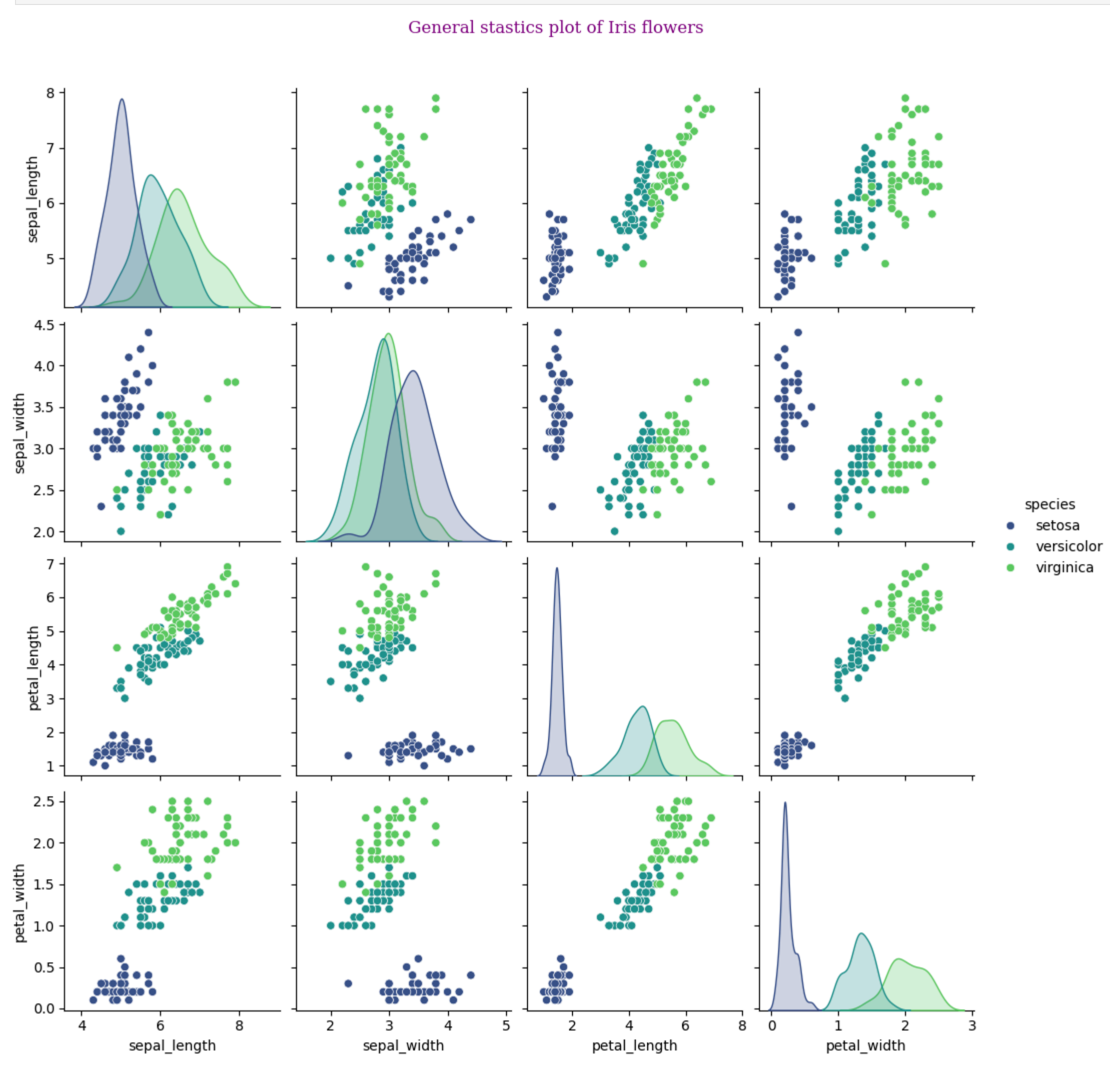
```
Stastical Summary of Iris Data:
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      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
```

Description: This code generates a descriptive statistical summary of the numerical features in the Iris dataset using Pandas' describe() function. It provides useful statistics such as the mean, standard deviation, minimum, maximum, and quartile values for each feature (sepal length, sepal width, petal length, and petal width).

Paiplot

A pairplot is a type of visualization in Seaborn that creates pairwise relationships between variables in a dataset. It's an excellent tool for exploratory data analysis (EDA), allowing you to quickly visualize the relationships between different features and detect patterns, correlations, or clusters.

In [75]:
```
#visualization of stastical summary of iris dataset
sns.pairplot(iris_data,hue='species',palette='viridis')
plt.suptitle(' General stastics plot of Iris flowers',x=0.5,y=1.05,fontdict={'size':16,'weight':'bold','family':'serif'},color='purple')
plt.show()
```



Description: This code generates a pair plot that visualizes the relationships between various features (sepal length, sepal width, petal length, petal width) of Iris flowers, with color-coding based on the species.

2. Pie plot for sepcies frequency

Pie chart

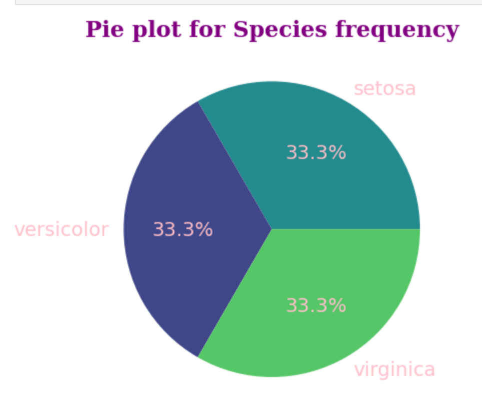
A pie chart is a circular statistical graphic that is divided into slices to illustrate numerical proportions. Each slice represents a category, and the size of each slice is proportional to the value or percentage that it represents in relation to the whole dataset. Pie charts are widely used to show how different categories contribute to a whole, making them great for displaying simple, relative comparisons.

In [10]:
```
species_count=iris_data['species'].value_counts()
```

In [11]:
```
plt.figure(figsize=(6,6))
```

Out[11]:
```
<Figure size 600x600 with 0 Axes>
<Figure size 600x600 with 0 Axes>
```

In [85]:
```
plt.pie(species_count,labels=species_count.index,autopct='%1.1f%%',colors={'#404788FF','#238A8DFF','#55C667FF'},textprops={'color':'pink','fontsize':14})
plt.title('Pie plot for Species frequency',fontdict={'size':16,'weight':'bold','family':'serif'},color='purple')
plt.show()
```



Description: This code generates a pie chart to visualize the distribution of different species in the Iris dataset. Each slice of the pie represents a species, and the size of the slice corresponds to the frequency of that species in the dataset.

Key elements of the pie chart:

Slices: Each species (Setosa, Versicolor, Virginica) is represented by a slice of the pie chart. The size of the slice reflects the proportion of that species in the dataset.

Labels: The labels (species names) are displayed on each slice, along with the percentage of the total that each species represents (formatted to 1 decimal place).

Colors: Each species is distinguished by a unique color, using a set of hex codes for blue, teal, and green shades.

Text Customization: The text (species names and percentages) is colored pink and set to a font size of 14 for readability.

Title: The chart has a bold, purple title that clearly describes the purpose of the plot.

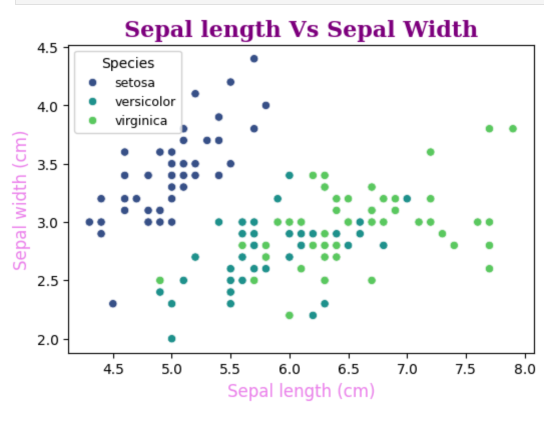
3. Relation between Sepal Length and Width

Scatter Plot

A scatter plot is a type of data visualization used to display the relationship between two continuous variables. Each point on the scatter plot represents an observation in the dataset, with one variable plotted along the x-axis and the other variable plotted along the y-axis. It's particularly useful for identifying patterns, correlations, and potential outliers in the data.

In [71]:
```
plt.figure(figsize=(6,4))
sns.scatterplot(x='sepal_length',y='sepal_width',hue='species',data=iris_data,palette='viridis')

#adding lables and legends
plt.title('Sepal length Vs Sepal Width',fontsize=16,color='purple',fontdict={'size':16,'weight':'bold','family':'serif'})
plt.xlabel('Sepal length (cm)',fontsize=12,color='violet')
plt.ylabel('Sepal width (cm)',fontsize=12,color='violet')
plt.legend(title='Species',fontsize=9)
plt.show()
```



Description: This code generates a scatter plot that visualizes the relationship between sepal length and sepal width for different species in the Iris dataset. Key features include:

Points:Represent individual flowers, with sepal length on the x-axis and sepal width on the y-axis.

Color coding: Points are colored based on the species (Setosa, Versicolor, and Virginica), using the 'viridis' palette, which provides distinct colors.

Title: The plot is titled 'Sepal length Vs Sepal Width' in bold, purple text, making it clear what relationship is being visualized.

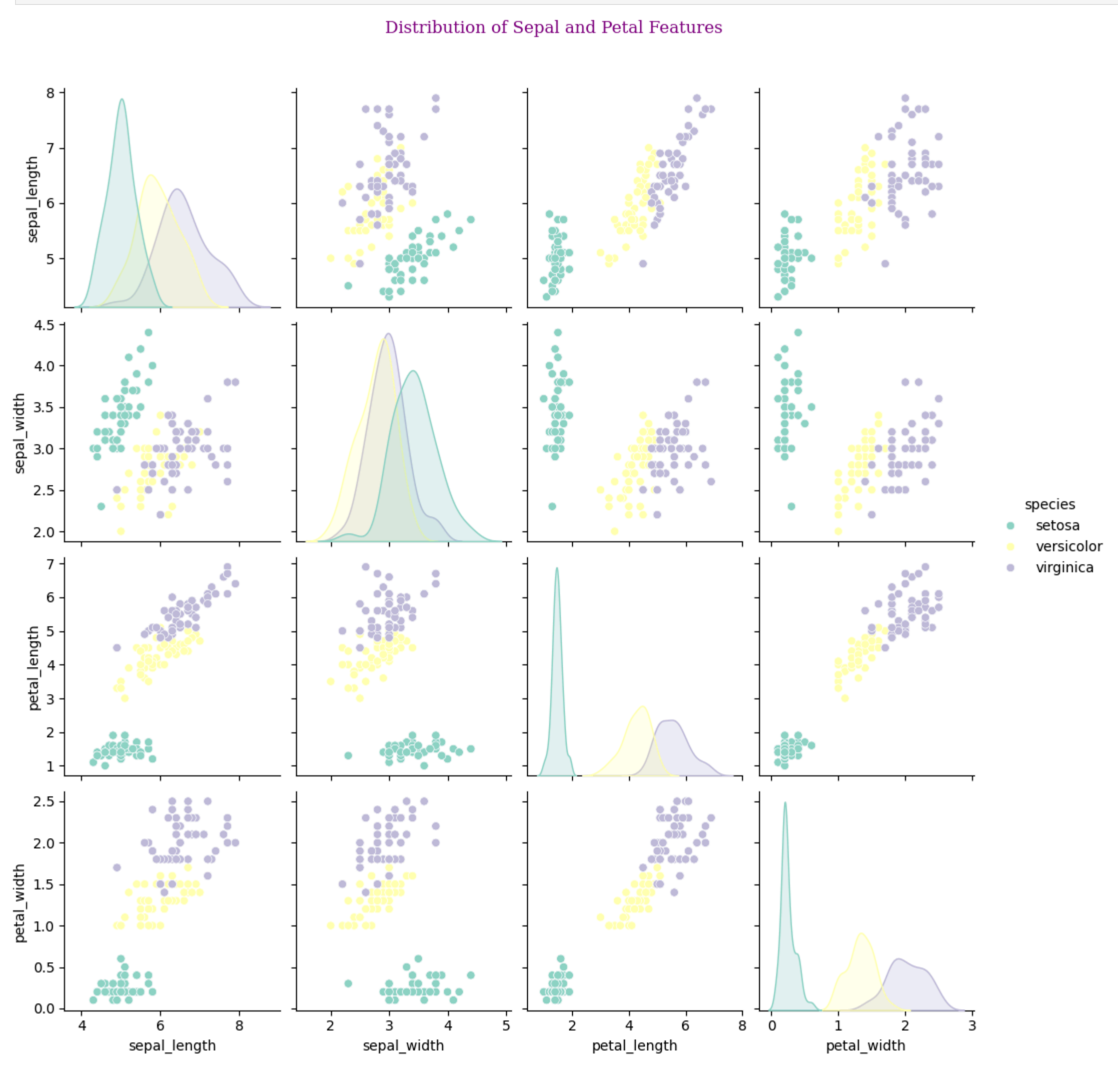
Axis labels: Sepal length is labeled on the x-axis, and sepal width is labeled on the y-axis, both in violet for clarity.

Legend: A legend is included to indicate which species corresponds to which color, making it easy to distinguish the species in the plot.

4. Distribution of Sepal and Petal Features

Pair plot

In [77]:
```
sns.pairplot(iris_data,hue='species',height=2.5,palette='Set3')
plt.suptitle('Distribution of Sepal and Petal Features',x=0.5,y=1.05,ha='center',color='purple',fontdict={'size':16,'weight':'bold','family':'serif'})
plt.show()
```



Description: This code generates a pair plot to visualize the relationships between the four main features of the Iris dataset:

1. Sepal length,
2. Sepal width,
3. Petal length,
4. Petal width

Key elements of the plot:

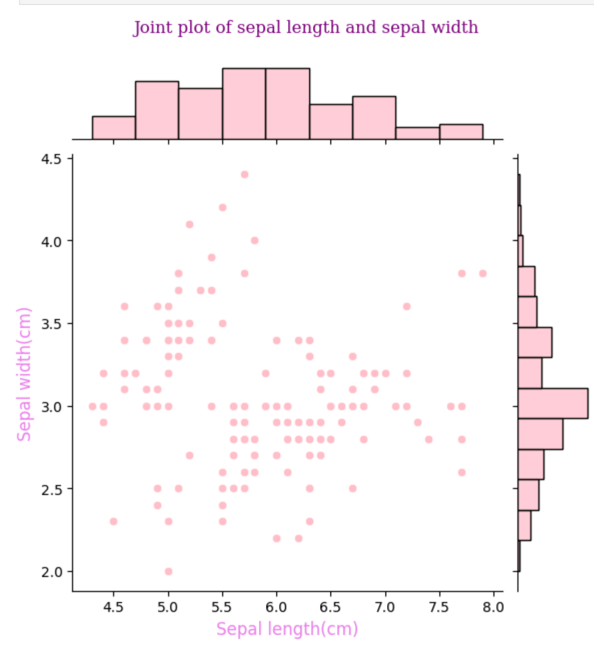
Scatter plots are displayed for each pair of features to illustrate how they relate to one another. Histograms (or diagonal density plots) show the distribution of each feature individually. The points in the scatter plots are color-coded by species (Setosa, Versicolor, Virginica), as indicated by the hue='species' parameter, with the color palette 'Set3' used to assign distinct colors to each species. The overall title describes the plot as the "Distribution of Sepal and Petal Features," making it clear that the plot showcases how these features vary among the three species.

5. Joint plot of sepal length and sepal width

Joint plot

A joint plot is a powerful visualization provided by Seaborn that simultaneously displays the relationship between two variables using both a scatter plot (or other types of bivariate plots) and the marginal distributions of each variable along the axes. It's a great tool for bivariate analysis and gives you a comprehensive view of how two variables relate to each other, while also showing their individual distributions.

In [60]:
```
joint_plot=sns.jointplot(x='sepal_length',y='sepal_width',data=iris_data,kind='scatter',color='pink')
plt.suptitle('Joint plot of sepal length and sepal width',x=0.5,y=1.05,ha='center',color='purple',fontdict={'size':16,'weight':'bold','family':'serif'})
joint_plot.set_axis_labels('Sepal length(cm)','Sepal width(cm)',fontsize=12,color='violet')
plt.show()
```



Description: This joint plot is particularly useful for analyzing both the relationship (scatter plot) and individual distributions (histograms) of sepal length and width. The scatter plot shows how these variables correlate, while the histograms help to understand the frequency distribution of each variable independently.

6. KDE plot for Setosa species(sepal length vs sepal width)

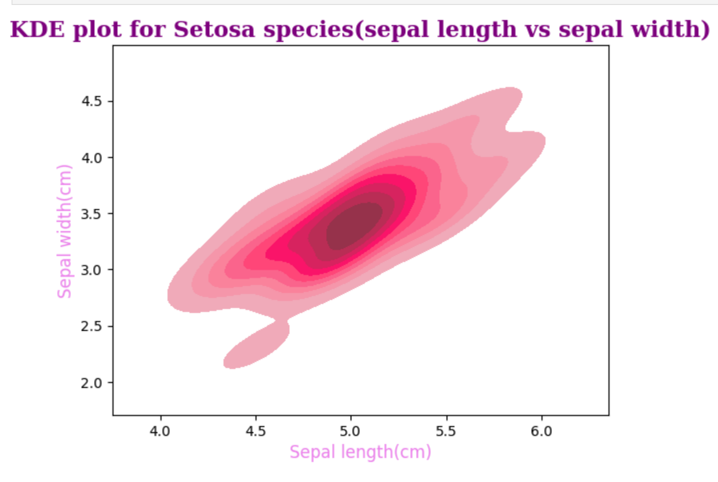
kDE plot

A KDE plot (Kernel Density Estimate plot) is a method for visualizing the distribution of a dataset in a smooth and continuous way. Unlike histograms, which show the frequency of data points using bars, KDE plots create a smooth curve to represent the probability density function (PDF) of the data, making it easier to identify underlying patterns or distributions.

KDE plots are particularly useful when you want to understand the probability distribution of a dataset without the abrupt binning of a histogram. They are often used for univariate (single variable) and bivariate (two-variable) data analysis.

In [16]:
```
setosa=iris_data[iris_data['species']=='setosa']
```

In [61]:
```
sns.kdeplot(x='sepal_length',y='sepal_width',data=setosa,fill=True,color='pink')
plt.title('KDE plot for Setosa species(sepal length vs sepal width)',fontdict={'size':16,'weight':'bold','family':'serif'},color='purple')
plt.xlabel('Sepal length(cm)',fontsize=12,color='violet')
plt.ylabel('Sepal width(cm)',fontsize=12,color='violet')
plt.show()
```

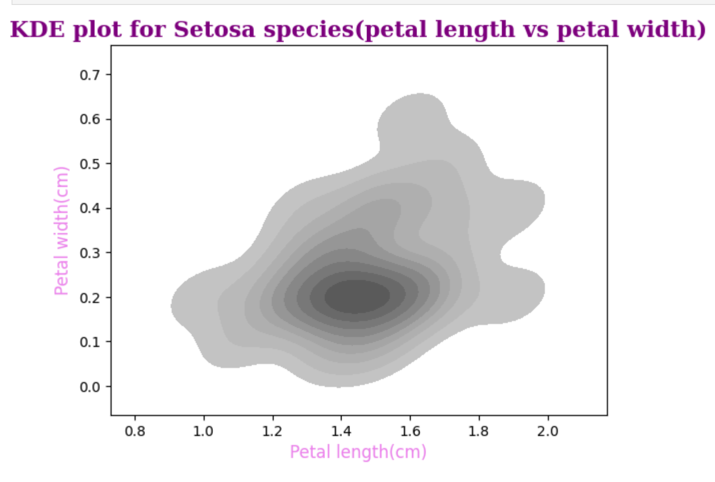


Description: The code creates a Kernel Density Estimate (KDE) plot specifically for the Setosa species of the Iris dataset, visualizing the joint distribution of sepal length and sepal width. The plot is filled and colored pink, with a bold, purple title and violet axis labels. The goal is to explore the smooth, continuous probability distribution of these two variables for the Setosa species.

This bivariate KDE plot helps in understanding the relationship between the two features, providing a clearer picture of where the density of points is highest, which could indicate the central tendency or clustering of the data.

7. 6. KDE plot for Setosa species(petal length vs petal width)

In [62]:
```
sns.kdeplot(x='petal_length',y='petal_width',data=setosa,fill=True,color='black')
plt.title('KDE plot for Setosa species(petal length vs petal width)',fontdict={'size':16,'weight':'bold','family':'serif'},color='purple')
plt.xlabel('Petal length(cm)',fontsize=12,color='violet')
plt.ylabel('Petal width(cm)',fontsize=12,color='violet')
plt.show()
```



Description: This KDE plot helps in understanding the joint distribution of the petal dimensions (length and width) for Setosa, showing where the density of data points is highest. Areas with darker shading indicate higher density, while lighter areas represent lower density, revealing patterns in how the petal size varies in this species.

By visualizing the KDE, you can infer tendencies, such as whether Setosa's petals tend to cluster around certain dimensions or if there's a notable spread in their sizes.

In [ ]: