iris-dataset

September 17, 2024

Matplotlib

Matplotlib is a comprehensive library for creatinhg static,animated,and interactive visualizations in Python.It is widely used for plotting data and offers flexibility and control over the appearance of plots.It provides a wide variety of plotting options including line plots, scatter plots, bar charts, histograms, and more.

Seaborn

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. Seaborn simplifies the process of creating common visualization types and integrates well with Pandas data structures.

[2]:
```
import seaborn as sns
import matplotlib.pyplot as plt

#load the iris dataset
iris=sns.load_dataset("iris")
iris.head()
```

[2]:

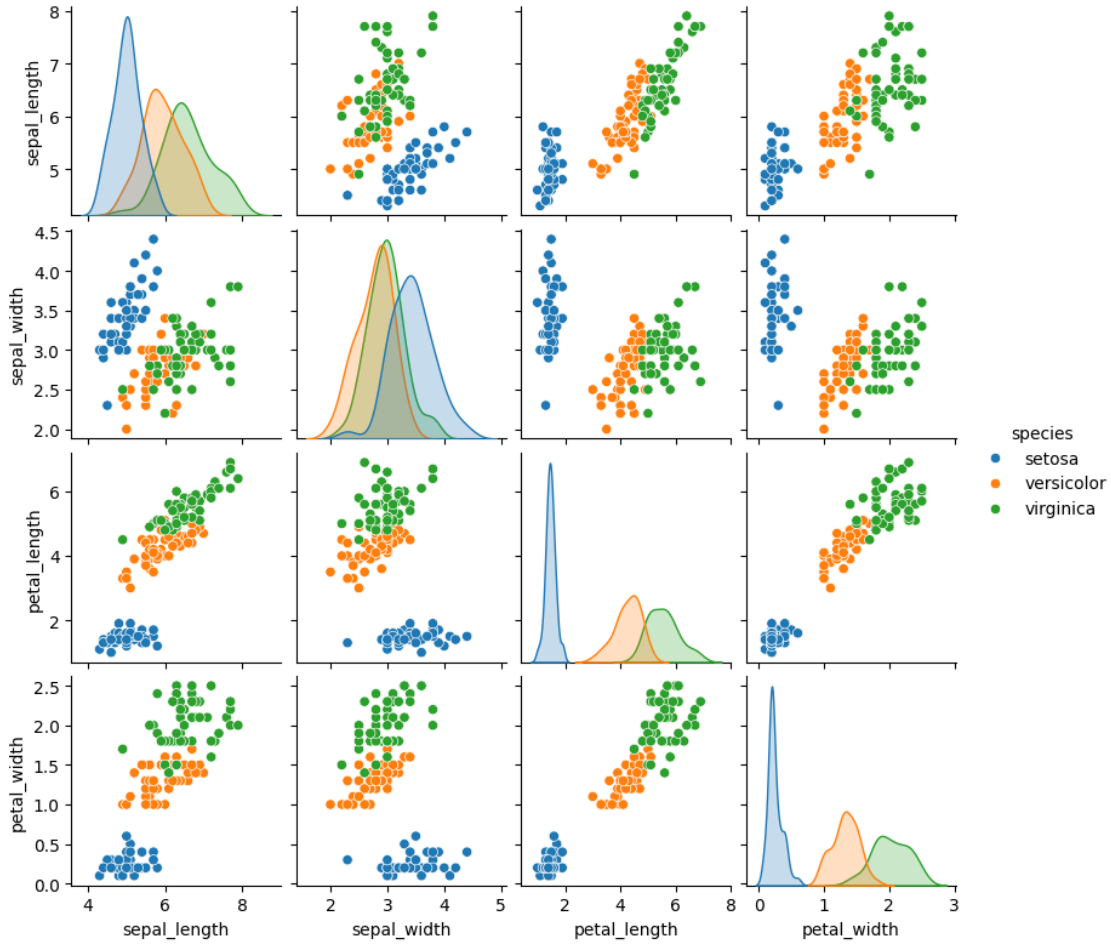
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

General Statistics Plot (Matplotlib or Seaborn):

Pairplot : A pairplot is a great way to visualize relationships between all pairs of features in a dataset. It creates a matrix of scatter plots, where each scatter plot shows the relationship between two different features.The diagonal of the matrix shows the distribution of each feature using histograms or kernel density estimations.

[3]:
```
# Create a pairplot to visualize relationships between features in the Iris␣
 ↪dataset.
# The 'hue' argument colors points based on the 'species' column
# 'height' sets the height of each subplot in the pairplot
sns.pairplot(iris,hue="species",height=2)
```

```
# Display the generated plot
plt.show()
```

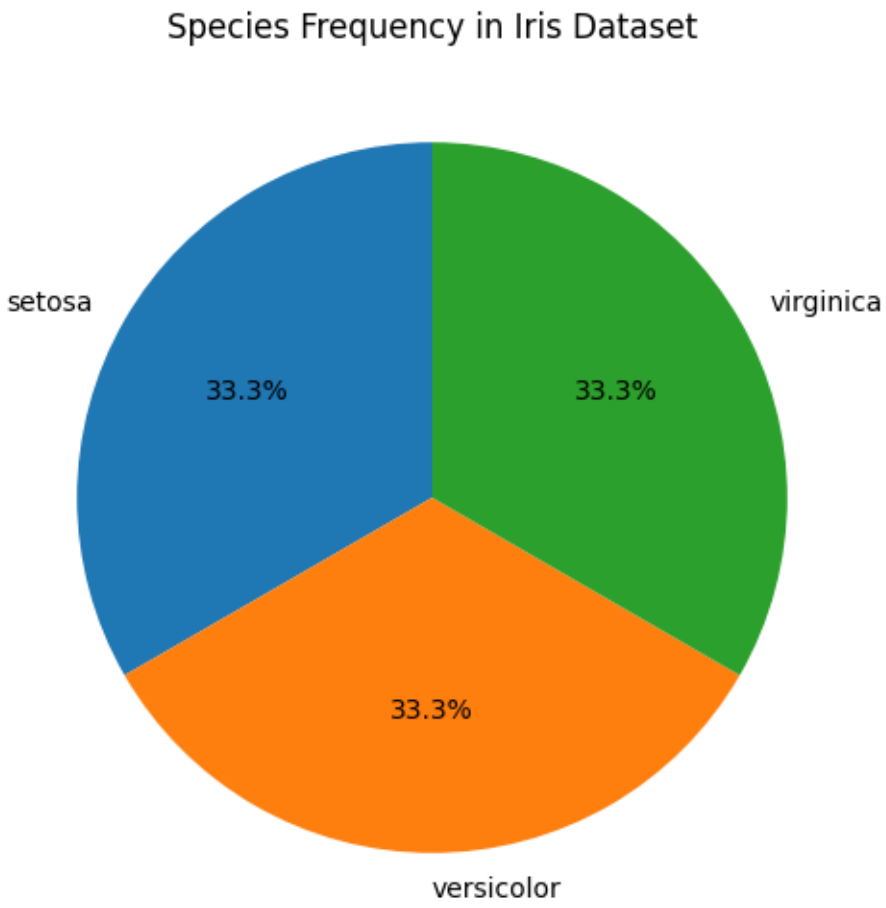


Pie Plot for Species Frequency:

Pie chart : A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice (and consequently its central angle and area), is proportional to the quantity it represents. While it is named for its resemblance to a pie which has been sliced, there are variations on the way it can be presented.

[4]:
```
# Calculate the frequency of each species in the 'species' column
species_counts=iris['species'].value_counts()

# Create a new figure for the plot with a specified size
plt.figure(figsize=(6,6))

# Create a pie chart of species frequencies.
# 'labels' sets the labels for each slice to the species names.
```

```
# 'autopct' formats the percentage displayed on each slice.
# 'startangle' sets the starting angle for 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()
```

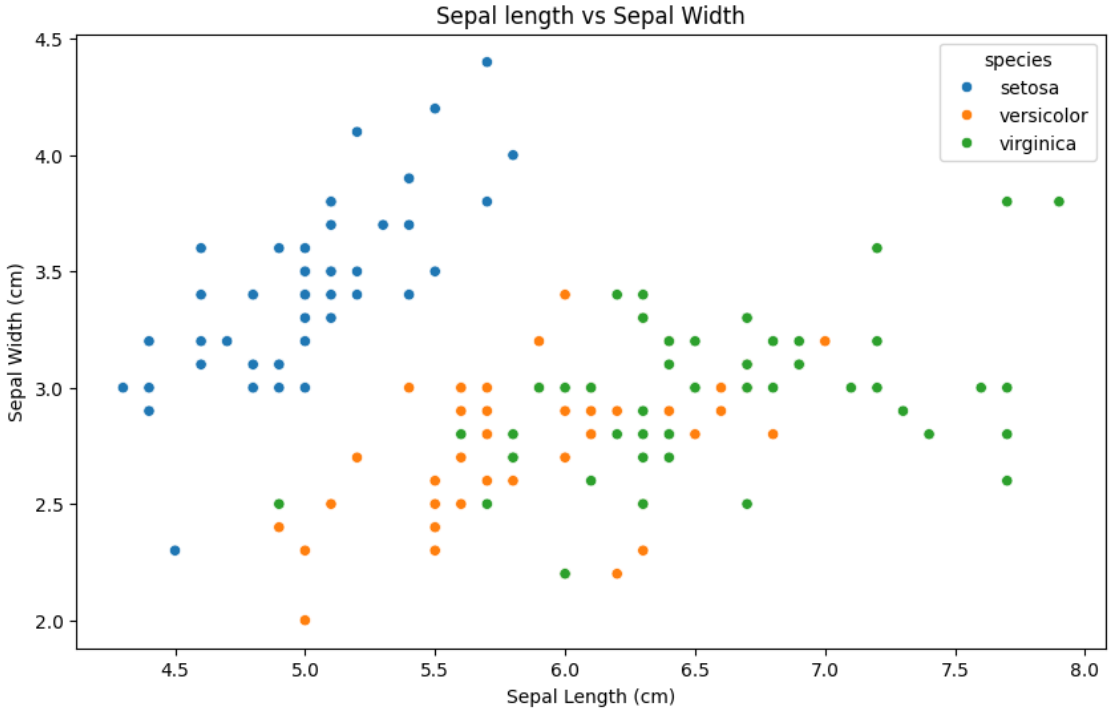


Relationship Between Sepal Length and Width:

Scatter plot : A scatter plot uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

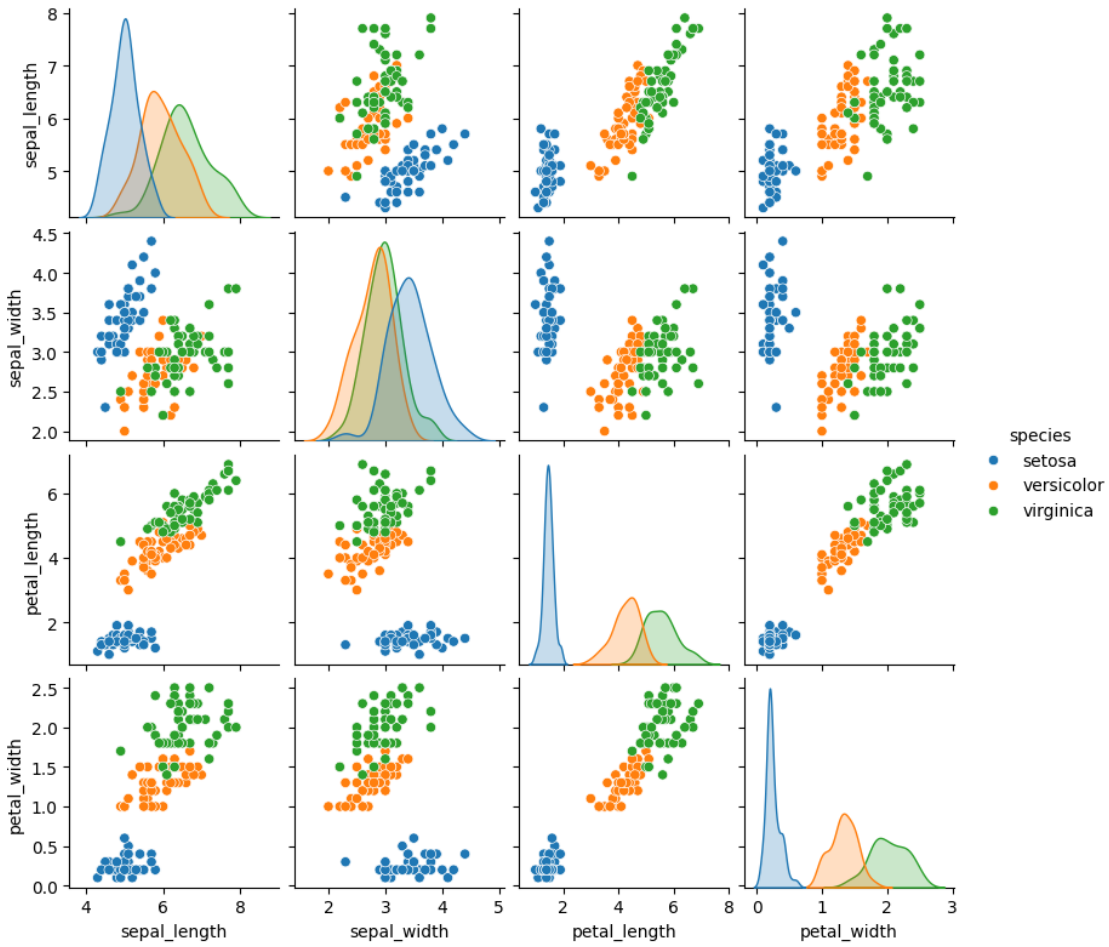
[5]:
```
# Create a new figure for the plot with a specified size.
plt.figure(figsize=(10, 6))
```

```
# Create a scatter plot to visualize the relationship between sepal length and␣
 ↪width.
# 'x' and 'y' specify the columns for the x and y axes.
# 'hue' colors points based on the 'species' column.
# 'data' specifies the dataset to use for the plot.
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=iris)

plt.title('Sepal length vs Sepal Width')  # title of scatter plot

plt.xlabel('Sepal Length (cm)')           # x axis label of scatter plot

plt.ylabel('Sepal Width (cm)')            # y axis label of scatter plot

plt.show()
```



Distribution of Sepal and Petal Features:

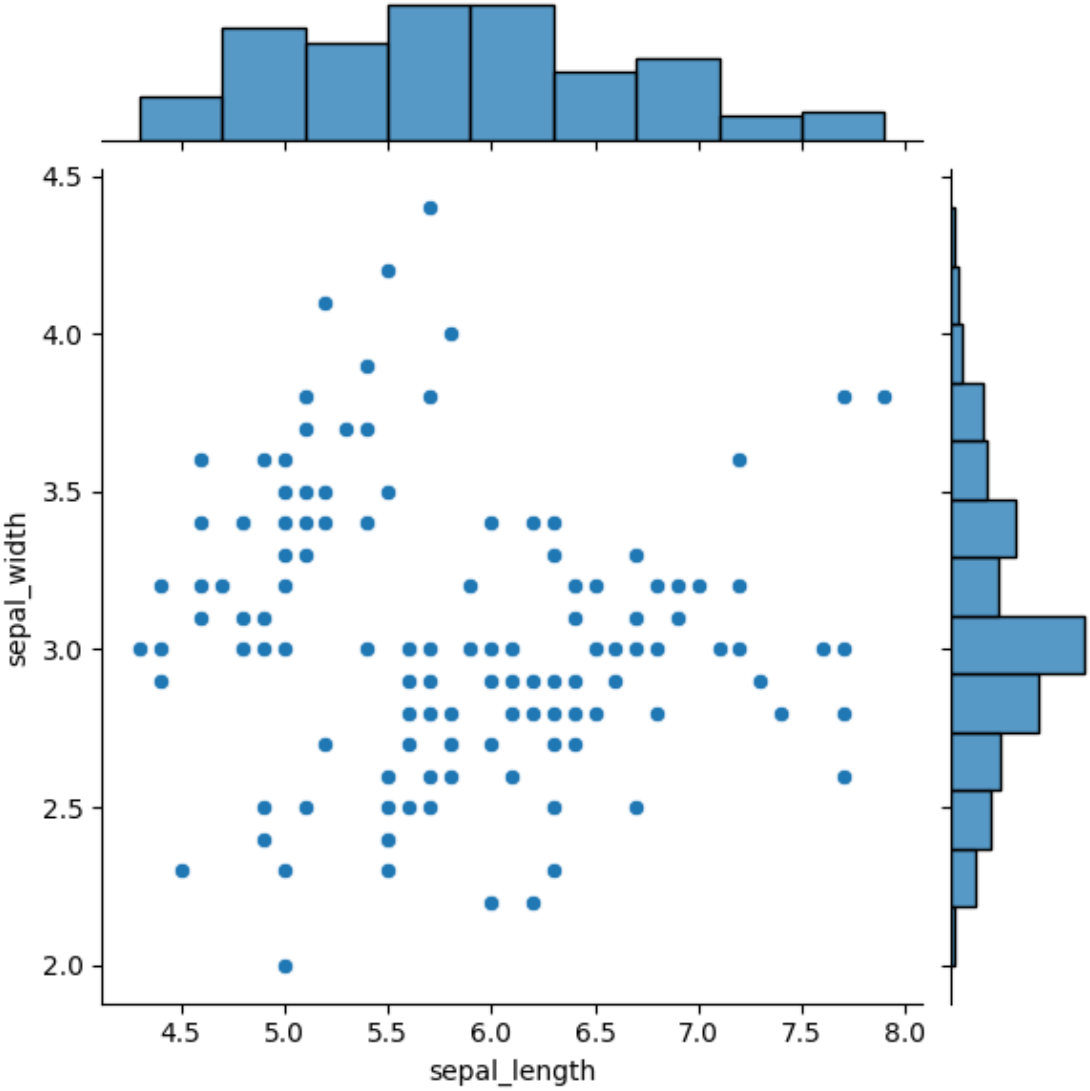
[6]:
```
sns.pairplot(iris,hue="species",height=2)
plt.show()
```



Jointplot of Sepal Length vs Sepal Width:

Joint plot : A jointplot is a way to visualize the relationship between two variables, along with their individual distributions. It combines a scatter plot with histograms or kernel density estimations to show both the bivariate and univariate distributions.

[7]:
```
# Create a jointplot to visualize the relationship between sepal length and␣
  ↪sepal width.
sns.jointplot(x='sepal_length',y='sepal_width',data=iris,kind='scatter')
plt.show()
```



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

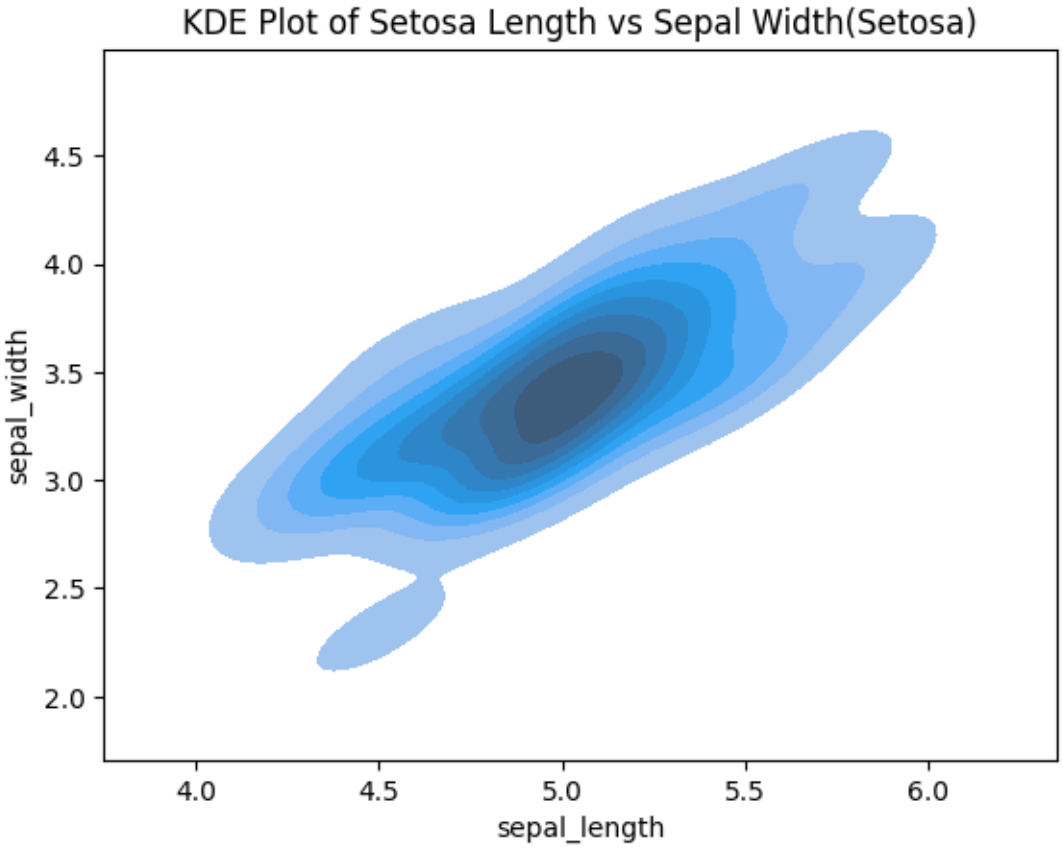
KDE Plot : A kdeplot (Kernel Density Estimation plot) is a way to visualize the distribution of a single variable or the relationship between two variables. It uses a kernel density estimate to show the probability density function of the data.

KDE plots are especially useful for visualizing the shape of a distribution and identifying any clusters or outliers.

[8]:
```
# Create a subset of the Iris dataset containing only the 'setosa' species
setosa=iris[iris['species']=='setosa']

# Create a KDE plot to visualize the distribution of sepal length and width for␣
 ↪the 'setosa' species
```

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

plt.title('KDE Plot of Setosa Length vs Sepal Width(Setosa)')
plt.show()
```

```
<ipython-input-8-c8c0981bcc8e>: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='sepal_length',y='sepal_width',data=setosa,shade=True)
```



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

[9]:
```
# Generate a KDE plot visualizing the distribution of petal length and width␣
  ↪for the 'setosa' species
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-9-1172a4bf08ba>:2: 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)
```