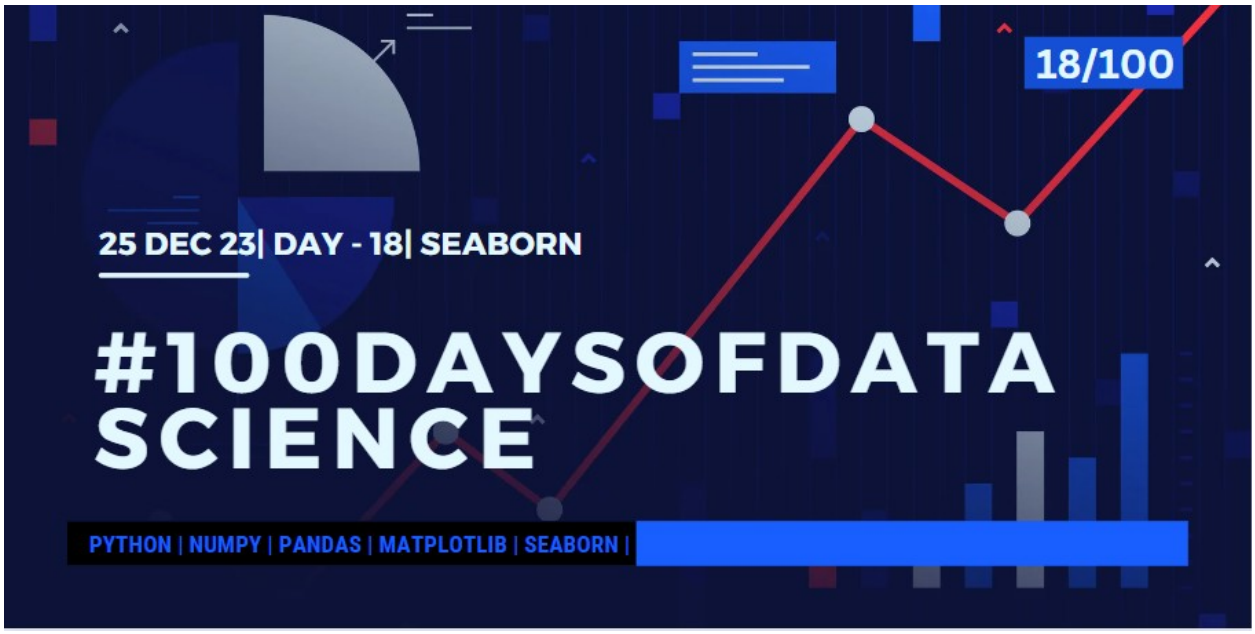


Seaborn

- Mostly used for statistical plotting in Python.
- It is built on top of Matplotlib and provides beautiful default styles and color palettes to make statistical plots more attractive.
- Seaborn is also closely integrated with the Panda's data structures.

Installation of Seaborn:

- First, we should ensure that, Python and PIP are installed in the system, then can install seaborn using the following pip command:
  - pip install seaborn

Import Seaborn:

- After successful installation of seaborn, we can import it using the following import module statement:
  - import seaborn

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Checking Seaborn Version

- The version string is stored under **version** attribute.

```
print(sns.__version__)
```

```
0.12.2
```

Categories of Plots in Python's seaborn library

Distribution plots: This type of plot is used for examining both types of distributions, i.e., univariate and bivariate distribution.

Relational plots: This type of plot is used to understand the relation between the two given variables.

Regression plots: Regression plots in the seaborn library are primarily intended to add an additional visual guide that will help to emphasize dataset patterns during the analysis of exploratory data.

Categorical plots: The categorical plots are used to deals with categories of variables and how we can visualize them.

Multi-plot grids: The multi-plot grids are also a type of plot that is a useful approach is to draw multiple instances for the same plot with different subsets of a single dataset.

Matrix plots: The matrix plots are a type of arrays of the scatterplots.

Loading Dataset

```
iris = sns.load_dataset("iris")
iris.head()
```

```
   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
```

```
print("-" * 100)
print("Shape: ",iris.shape)
print("-" * 100)
print("Columns: ",iris.columns)
print("-" * 100)
print("Info: ")
print(iris.info())
print("-" * 100)
print("Describe: ")
print(iris.describe())
print("-" * 100)
print("Checking Null values: ")
print(iris.isna().sum())
print("-" * 100)
```

```
----------------------------------------------------------------------------------------------------
Shape:  (150, 5)
----------------------------------------------------------------------------------------------------
```

```
------------------------------
Columns:  Index(['sepal_length', 'sepal_width', 'petal_length',
'petal_width',
       'species'],
      dtype='object')
------------------------------------------------------------------------
------------------------------
Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
------------------------------------------------------------------------
------------------------------
Describe:
       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
------------------------------------------------------------------------
------------------------------
Checking Null values:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
------------------------------------------------------------------------
------------------------------
```

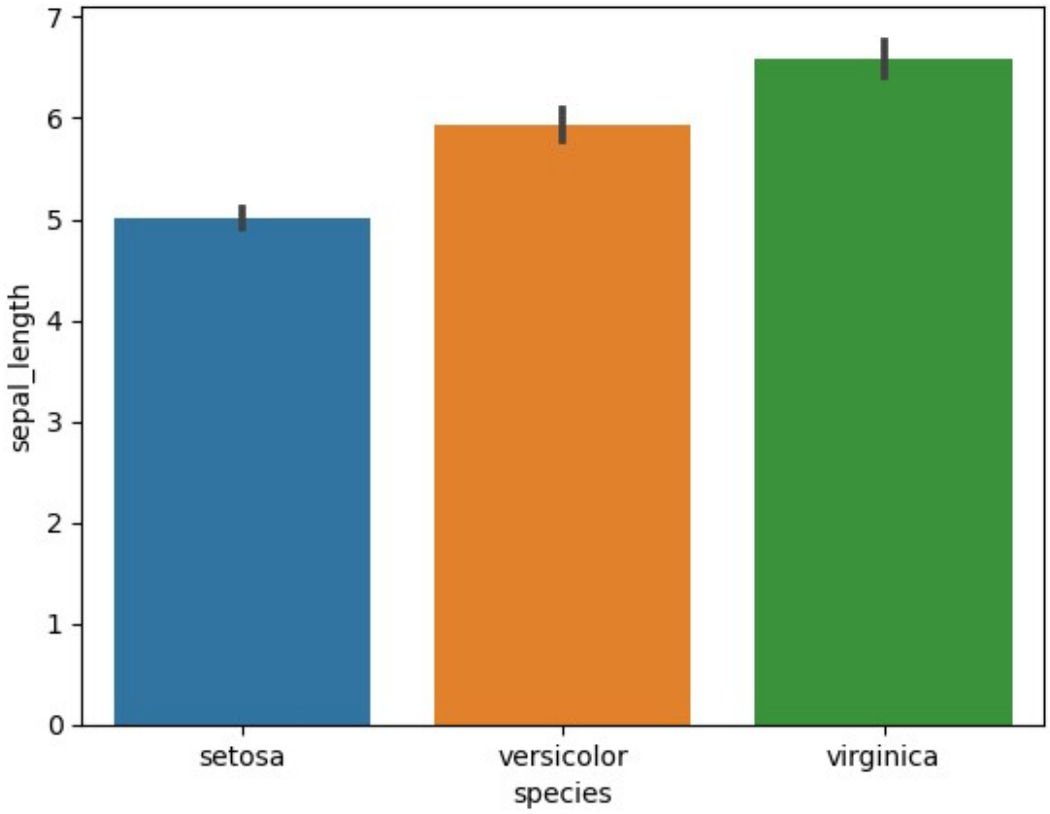
Categorical Plots

- Categorical Plots are used where we have to visualize relationship between two numerical values.
- A more specialized approach can be used if one of the main variable is categorical which means such variables that take on a fixed and limited number of possible values.
- There are various types of categorical plots:
    a. Bar Plot
    b. Count Plot
    c. Box Plot
    d. Violin Plot
    e. Strip Plot

1. Bar Plot:

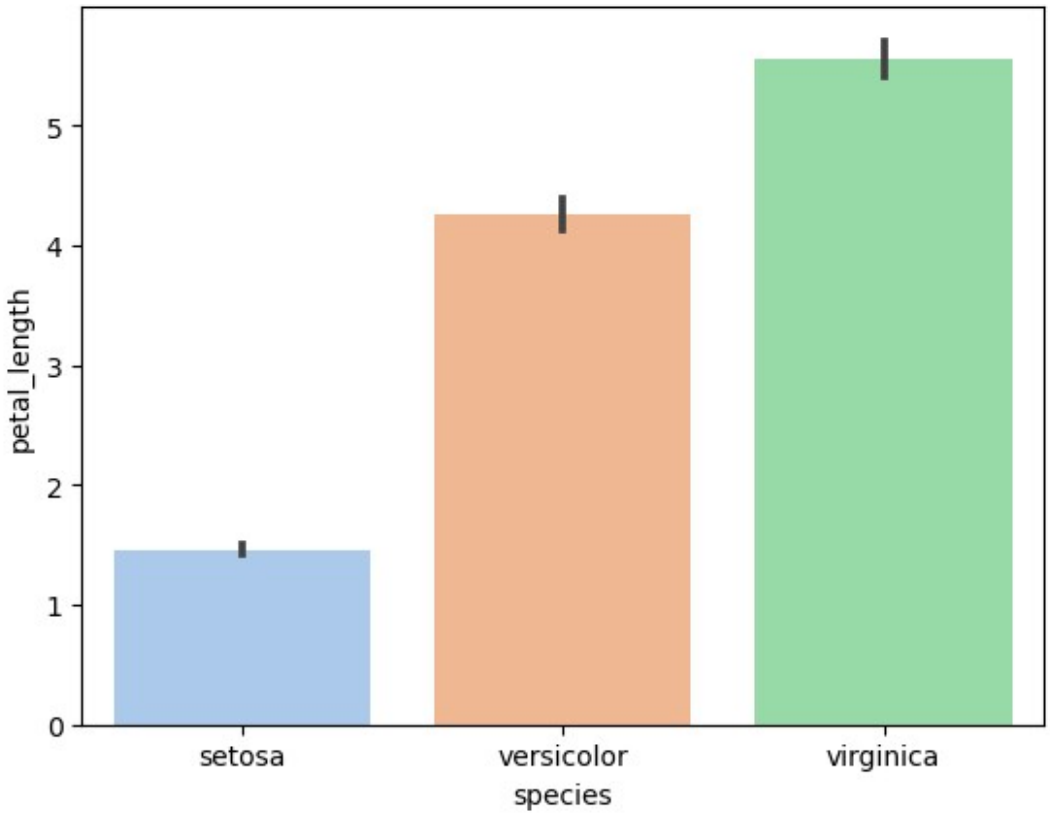
- A barplot is basically used to aggregate the categorical data according to some methods and by default, its the mean.
- It can also be understood as a visualization of the group by action.
- To use this plot we choose a categorical column for the x axis and a numerical column for the y axis and we see that it creates a plot taking a mean per categorical column.
- It can be created using the barplot() method.
- Syntax:
    - `barplot([x, y, hue, data, order, hue_order, …])`

```
sns.barplot(x='species',y='sepal_length',data=iris)
plt.show()
```




```
sns.barplot(x='species',y='petal_length',data=iris,palette='pastel')
```

```
<Axes: xlabel='species', ylabel='petal_length'>
```

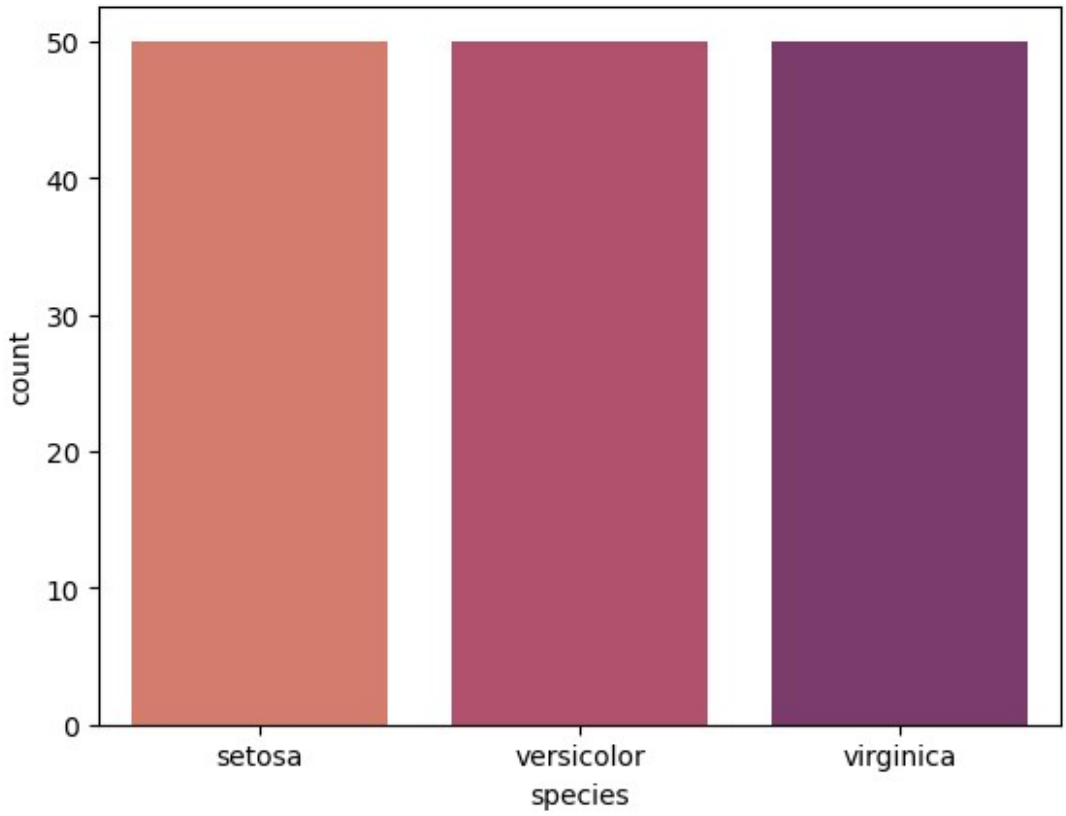


2. Count Plot:

- A countplot basically counts the categories and returns a count of their occurrences.
- It is one of the most simple plots provided by the seaborn library.
- It can be created using the countplot() method.
- Syntax:
  - `countplot([x, y, hue, data, order, …])`

```
sns.countplot(x='species',data=iris,palette='flare')
```

```
<Axes: xlabel='species', ylabel='count'>
```

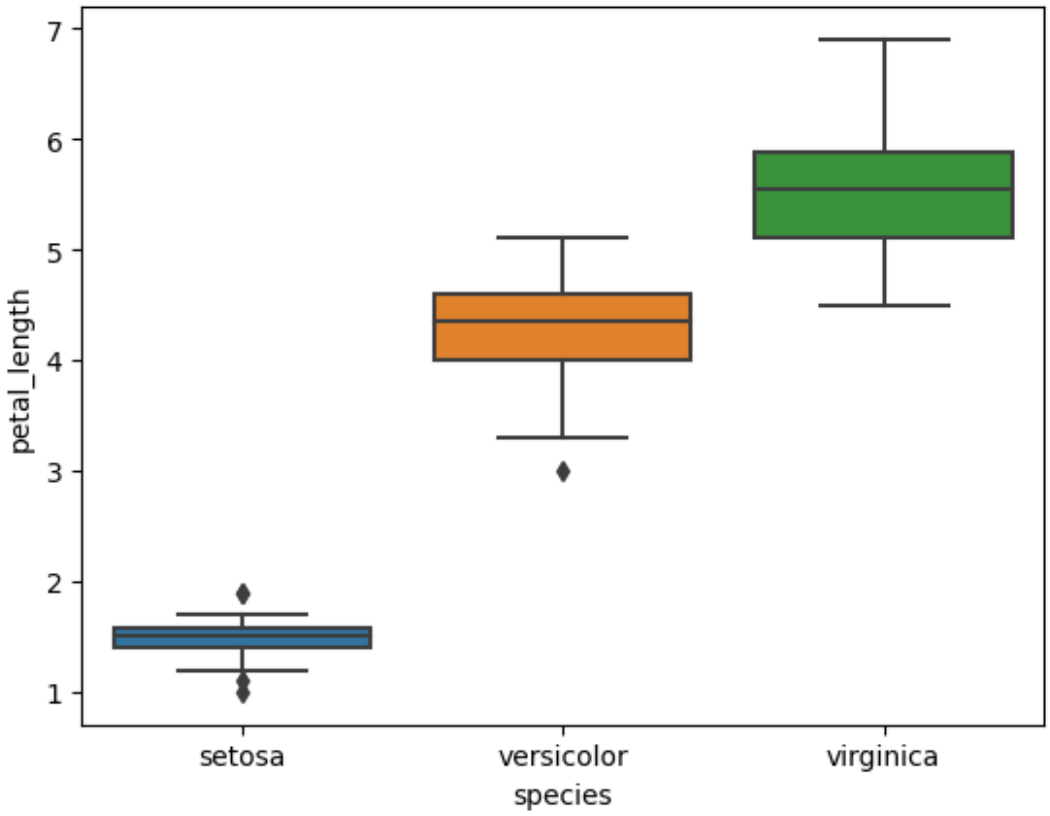


3. Box Plot:

- A boxplot is sometimes known as the box and whisker plot.
- It shows the distribution of the quantitative data that represents the comparisons between variables.
- Boxplot shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution i.e. the dots indicating the presence of outliers.
- It is created using the boxplot() method.
- Syntax:
  - `boxplot([x, y, hue, data, order, hue_order, …])`

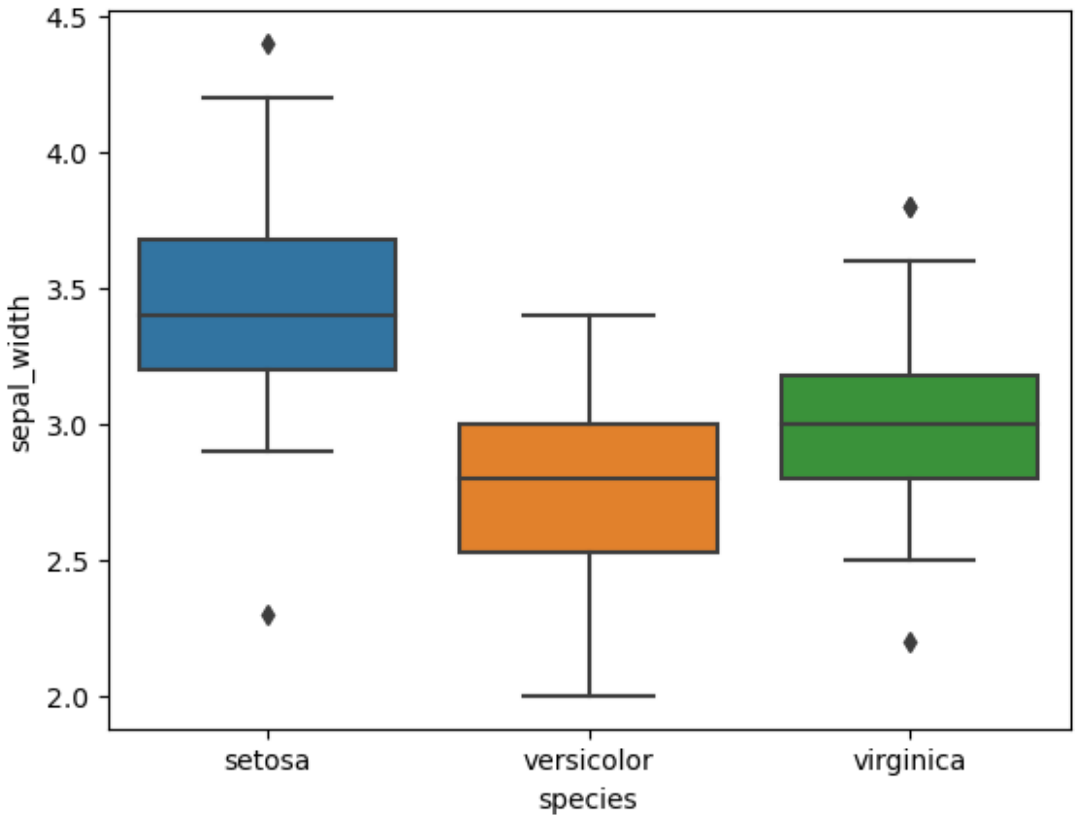
```
sns.boxplot(x='species',y='petal_length',data=iris)
```

```
<Axes: xlabel='species', ylabel='petal_length'>
```




```
sns.boxplot(x='species',y='sepal_width',data=iris)
```

```
<Axes: xlabel='species', ylabel='sepal_width'>
```

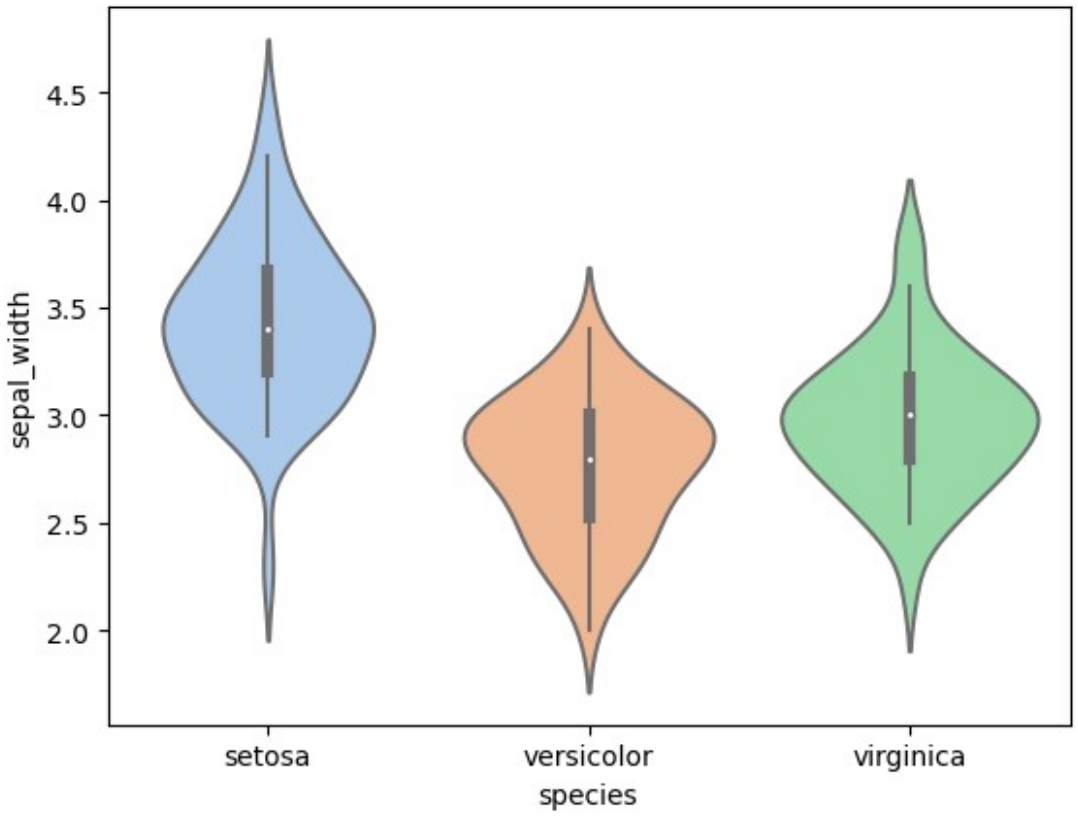


4. Violin Plot:

- It is similar to the boxplot except that it provides a higher, more advanced visualization and uses the kernel density estimation to give a better description about the data distribution.
- It is created using the violinplot() method.
- Syntax:
  - `violinplot([x, y, hue, data, order, …])`

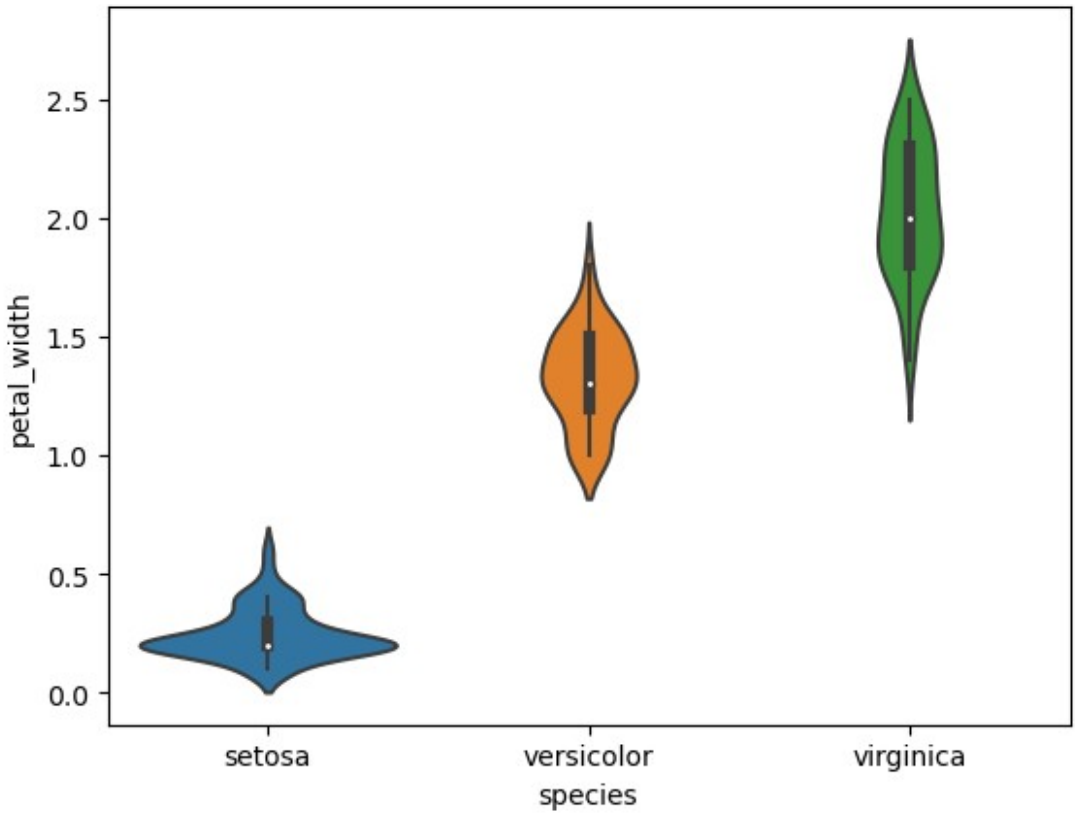
```
sns.violinplot(x='species',y='sepal_width',data=iris,
palette='pastel')
```

```
<Axes: xlabel='species', ylabel='sepal_width'>
```




```
sns.violinplot(x='species',y='petal_width',data=iris)
```

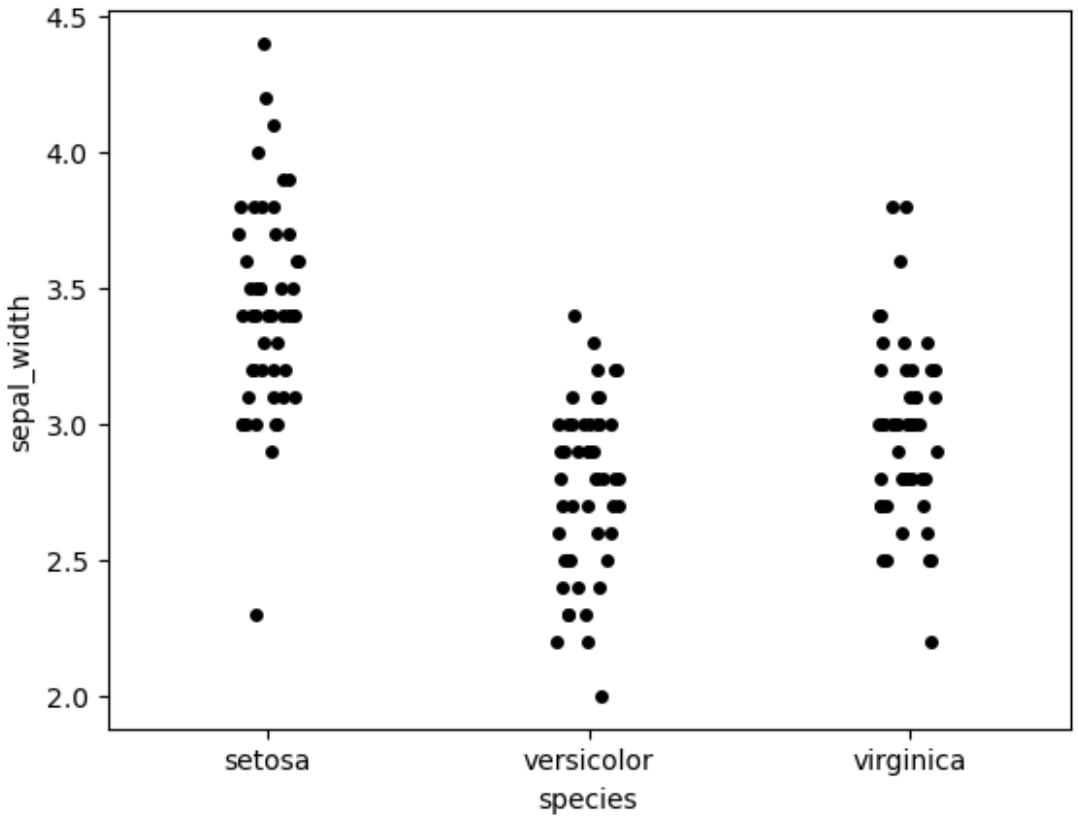
```
<Axes: xlabel='species', ylabel='petal_width'>
```



5. Strip Plot:

- It basically creates a scatter plot based on the category.
- It is created using the stripplot() method.
- Syntax:
  - `stripplot([x, y, hue, data, order, …])`

```
sns.stripplot(x='species', y='sepal_width', data=iris, color='black')
plt.show()
```



Relational Plots

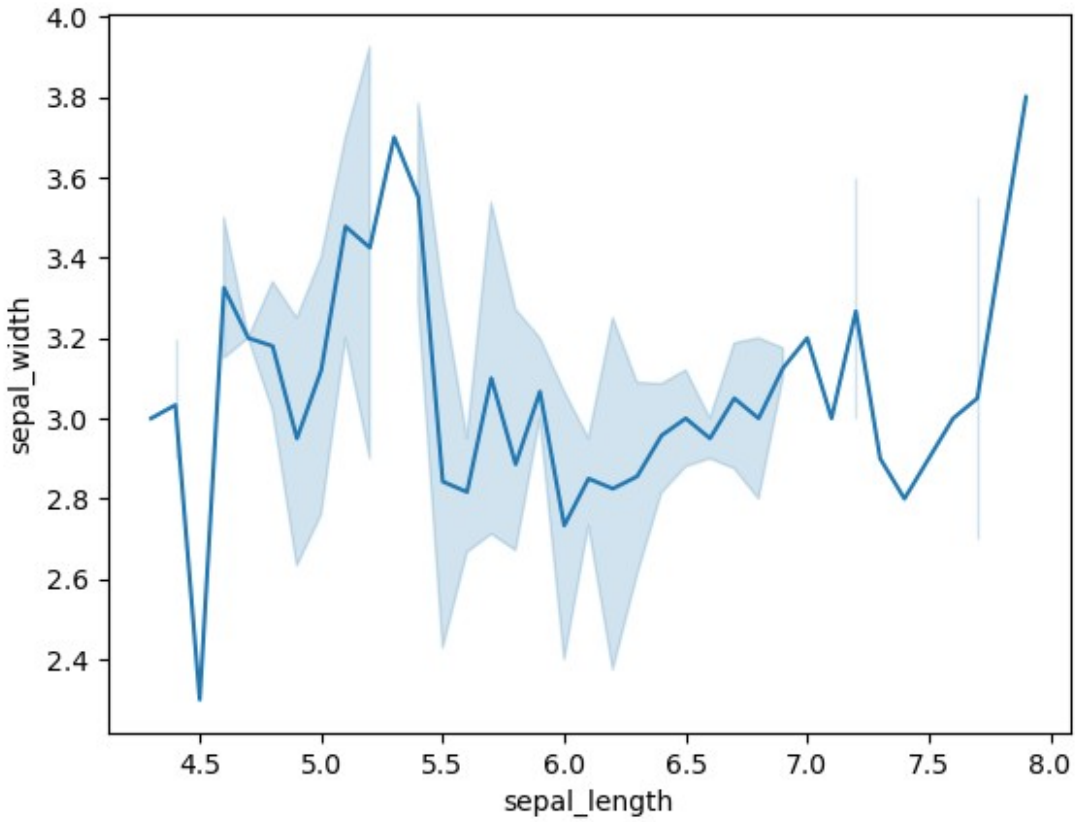
- Relational plots are used for visualizing the statistical relationship between the data points.
- It allows us to visualise how variables within a dataset relate to each other.
- There are various types of categorical plots:
    a. Line Plot
    b. Scatter Plot

1. Line Plot:

- The seaborn line plot is one of the most basic plots presents in the seaborn library.
- We use the seaborn line plot mainly to visualize the given data in some time-series form, i.e., in a continuous manner with respect to time.
- Syntax:
    - `lineplot([x, y, hue, data, style, order, …])`

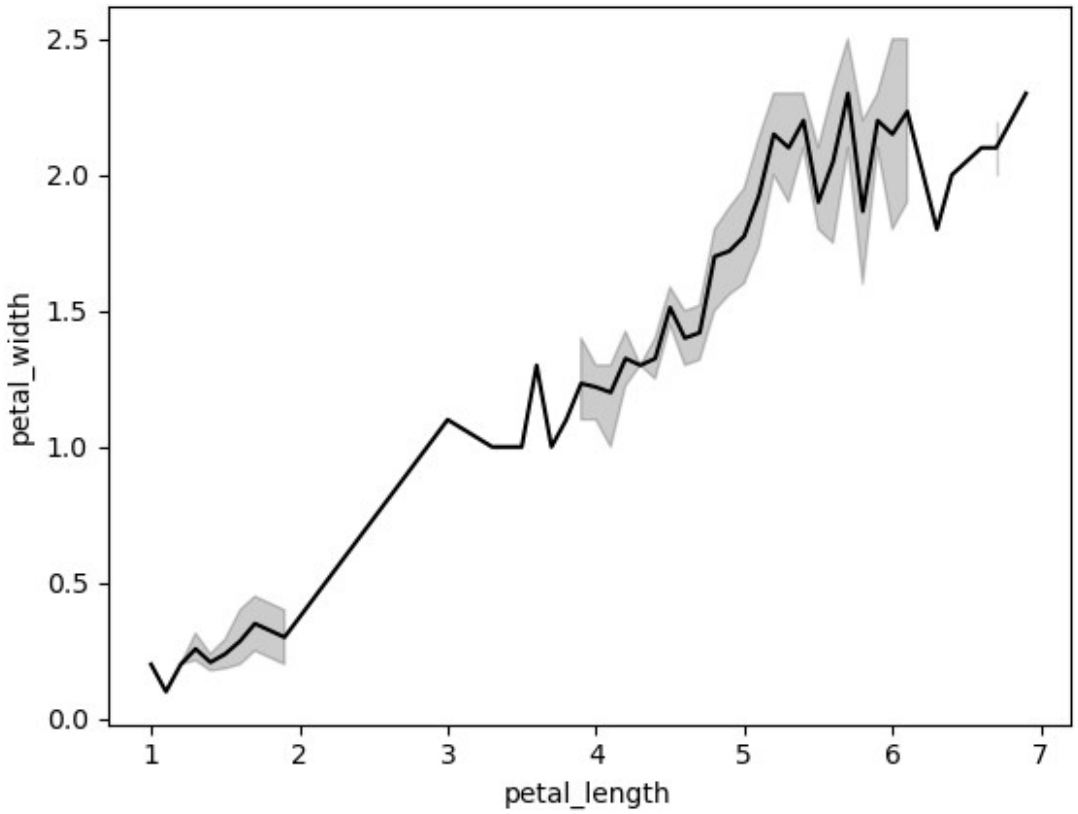
```
sns.lineplot(x="sepal_length", y="sepal_width", data=iris)
```

```
<Axes: xlabel='sepal_length', ylabel='sepal_width'>
```

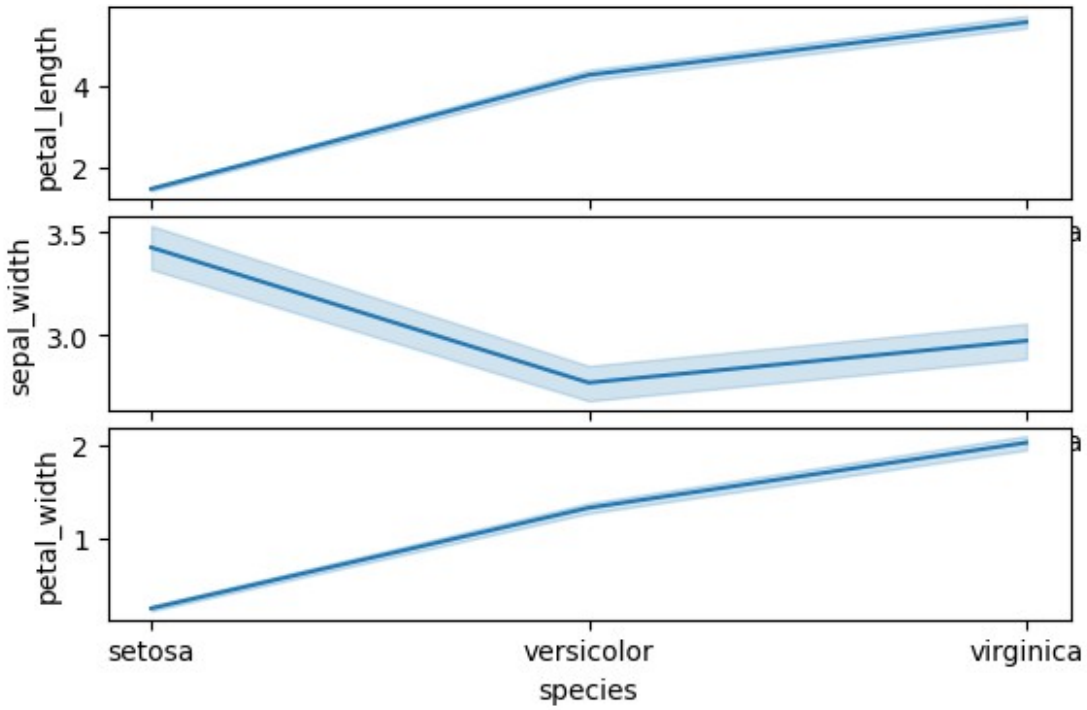



```
sns.lineplot(x="petal_length", y="petal_width", data=iris,
color='black')
```

```
<Axes: xlabel='petal_length', ylabel='petal_width'>
```



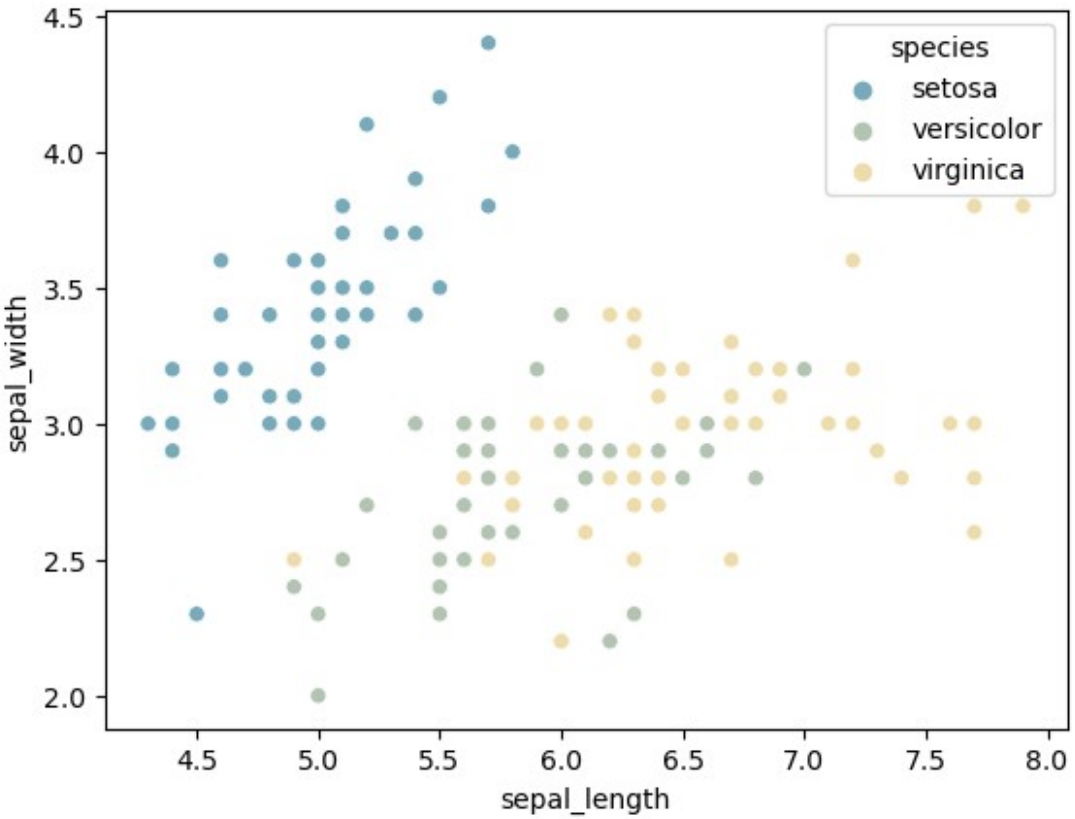

```
def graph1():
    sns.lineplot(x="species", y="sepal_width", data=iris)

def graph2():
    sns.lineplot(x="species", y="sepal_length", data=iris)

def graph3():
    sns.lineplot(x="species", y="petal_width", data=iris)

def graph4():
    sns.lineplot(x="species", y="petal_length", data=iris)


axes1 = plt.subplot2grid ((7, 1), (0, 0), rowspan = 2,  colspan = 1)
graph4()

axes2 = plt.subplot2grid ((7, 1), (2, 0), rowspan = 2, colspan = 1)
graph1()

axes3 = plt.subplot2grid ((7, 1), (4, 0), rowspan = 2, colspan = 1)
graph3()
```



2. Scatter Plot:

- The scatter plot is a mainstay of statistical visualization.
- It depicts the joint distribution of two variables using a cloud of points, where each point represents an observation in the dataset.
- This depiction allows the eye to infer a substantial amount of information about whether there is any meaningful relationship between them.
- It is plotted using the scatterplot() method.
- Syntax:
  - `scatterplot([x, y, hue, data, style, palette, order, …])`

```
sns.scatterplot(x='sepal_length', y='sepal_width', data=iris,
palette='blend:#7AB,#EDA', hue='species')
plt.show()
```


species
setosa
versicolor
virginica
sepal_width
sepal_length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0