IRIS DATASET BASIC ANALYSIS

1. **Exploratory Data Analysis (EDA) with Python**
2. **Data Visualization with Power BI or Tableau**
3. **Documentation**

Iris Dataset Basic Analysis

Project Requirements:

1.Exploratory Data Analysis (EDA) with Python:

- Use Python to perform basic EDA on the Iris dataset.

- Visualize key statistics and distributions to gain insights into the dataset.

2. Data Visualization with Power BI or Tableau:

- Create visualizations in Power BI or Tableau to represent the patterns observed during EDA.

- Explore correlations, patterns, and trends within the Iris dataset.

3. Documentation:

- Document your approach and methodologies used in the analysis.

- Provide clear explanations for the patterns identified in the Iris dataset.

IRIS DATASET CLASSIFICATION

The Iris dataset is a famous dataset in machine learning and statistics. It is often used for practicing classification algorithms. The dataset consists of 150 samples of iris flowers, each belonging to one of three species: setosa, versicolor, or virginica. Each sample has four features: sepal length, sepal width, petal length, and petal width.

Here's a basic description of the dataset:

Features:

- ✓ **Sepal Length (in cm)**
- ✓ **Sepal Width (in cm)**
- ✓ **Petal Length (in cm)**
- ✓ **Petal Width (in cm)**

Target Classes:

- ✓ **Setosa**
- ✓ **Versicolor**
- ✓ **Virginica**

EXPLORATORY DATA ANALYSIS

- ✓ **Exploratory Data Analysis (EDA) is a critical step in understanding the characteristics of a dataset before applying machine learning algorithms. While there might not be a specific documentation for EDA on the Iris dataset, I can guide you through some common EDA steps and provide you with a sample Python code using popular libraries like pandas, matplotlib, and seaborn.**

CLASSIFICATIONS

- ✓ **Download the Dataset "Iris.csv" from device**
  Iris dataset is the Hello World for the Data Science, so if you have started your career in Data Science and Machine Learning you will be practicing basic ML algorithms on this famous dataset.
- ✓ **Iris dataset contains five columns such as Petal Length, Petal Width, Sepal Length, Sepal Width and Species Type. Iris is a flowering plant, the researchers have measured various features of the different iris flowers and recorded digitally.**

GET STARTED

The function head() will display the top rows of the dataset, the default value of this function is 5, that is it will show top 5 rows when no argument is given to it.

Slicing means if you want to print or work upon a particular group of lines that is from 10^{th} row to 20^{th} row.

DATA VISILIZATION WITH POWER BI

1.Bar Chart

Bar charts are best suited for the visualization of categorical data because they allow you to easily see the difference between feature values by measuring the size(length) of the bars. There are 2 types of bar charts depending upon their orientation (i.e. vertical or horizontal). Moreover, there are 3 types of bar charts based on their representation that is shown below.

2. Scatter Plot

These are the charts/plots that are used to observe and display relationships between variables using Cartesian Coordinates. The values (x: first variable , y: second variable) of the variables are represented by dots. Scatter plots are also known as scattergrams, scatter graphs, scatter charts , or scatter diagrams. It is best suited for situations where the dependent variable can have multiple values for the independent variable.

3. Line Plot

Line plots is a graph that is used for the representation of continuous data points on a number line. Line plots are created by first plotting data points on the Cartesian plane then joining those points with a number line. Line plots can help display data points for both single variable analysis as well as multiple variable analysis.

4. Histograms

Histograms are used to represent the frequency distribution of continuous variables. The width of the histogram represents interval and the length represents frequency. To create a histogram you need to create bins of the interval which are not overlapping. Histogram allows the inspection of data for its underlying distribution, outliers, skewness.

5. Line Histograms (Setosa species,Versicoloe species)

Line histograms are the modification to the standard histogram to understand and represent the distribution of a single feature with different data points. Line histograms have a density curve passing through the histogram.

1. Setosa species

The Setosa species has smaller and less distributed features compared to the other two species.

2.Versicolor species

The Versicolor species is distributed in an average manner and has average-sized features. The Virginica species, on the other hand, is highly distributed with a large number of values and features.

The mean and median values of various features (such as sepal length and width, and petal length and width) are clearly shown by each plot for each species. This suggests that the distribution of these features varies significantly between the three species.

3.Visualization (jointplot)

A jointplot is a type of visualization that displays the joint distribution of two variables. It is a combination of a scatter plot and one or more histograms, and can be used to visualize the relationship between the two variables and the distribution of each variable separately.

The scatter plot portion of a jointplot shows the relationship between the two variables, while the histograms show the distribution of each variable individually. This allows you to see both the relationship between the variables and the distribution of each variable at the same time.

Source Code:

DATA VISILIZATION WITH POWER BI

Importing pandas to use in our code as pd.

```
Import pandas as pd
Import matplotlib.pyplot as plt
Import numpy as np
Import seaborn as sns
```

Reading the dataset "Iris.csv".

```
df=pd.read_csv("Iris.csv")
print(df)
```

OUTPUT

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

Displaying up the top rows of the dataset with their columns

```
df . head()
df.describe()
```

OUTPUT

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

Displaying Column

df.columns

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

Removing Duplicates

data=df.drop_duplicates(subset='Species')

data

OUTPUT

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
100	101	6.3	3.3	6.0	2.5	Iris-virginica

Data Slicing

print(df[10:21])

sliced_data=df[10:21]

```
print(sliced_data)
```

OUTPUT

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa
17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa
20	21	5.4	3.4	1.7	0.2	Iris-setosa
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa
17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa
20	21	5.4	3.4	1.7	0.2	Iris-setosa

Displayment Species

```
df.value_counts('Species')
```

OUTPUT

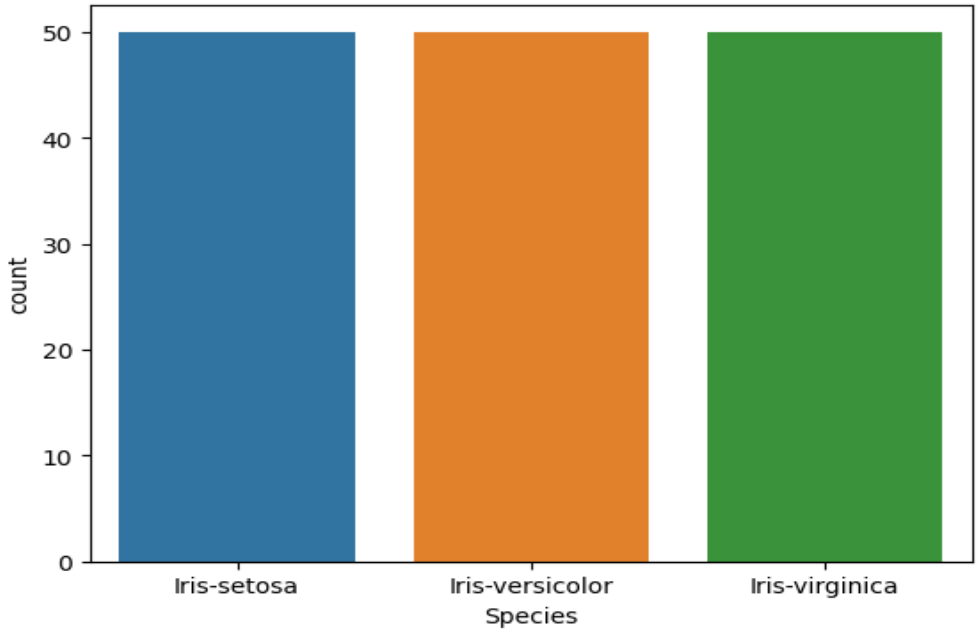
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

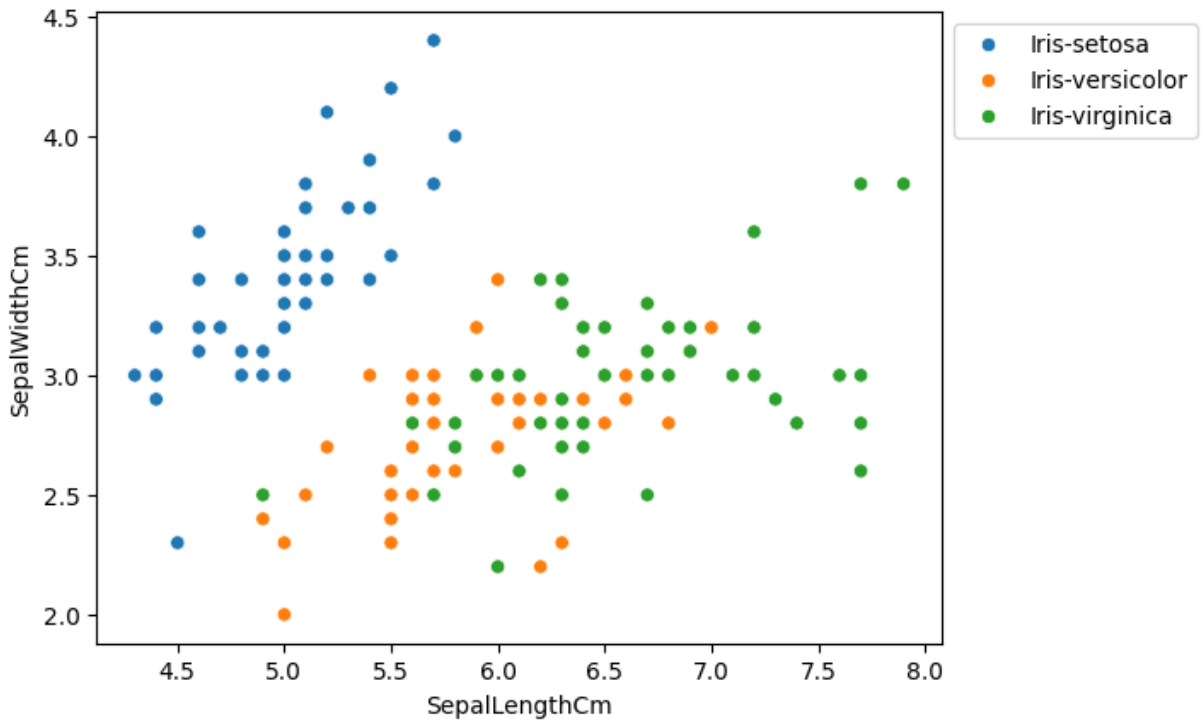
Displaying Visualization

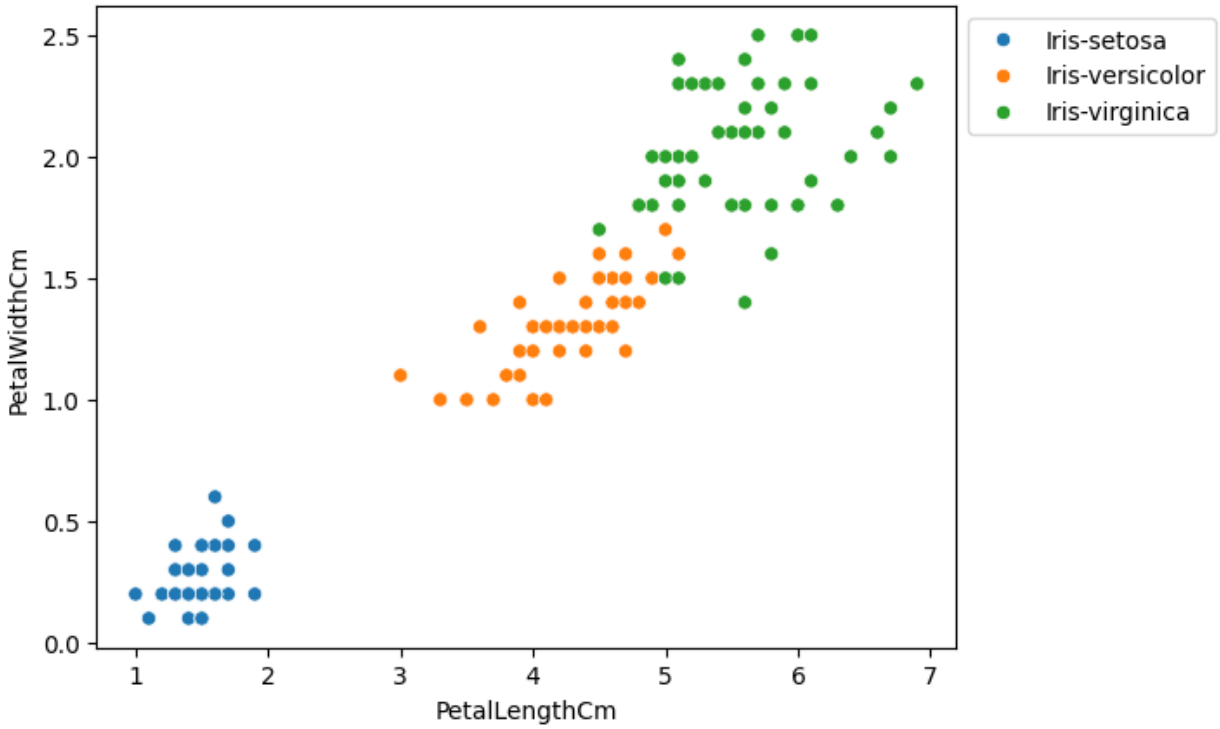
```
sns.countplot(x='Species', data=df ,hue='Species' )
plt.show()
```

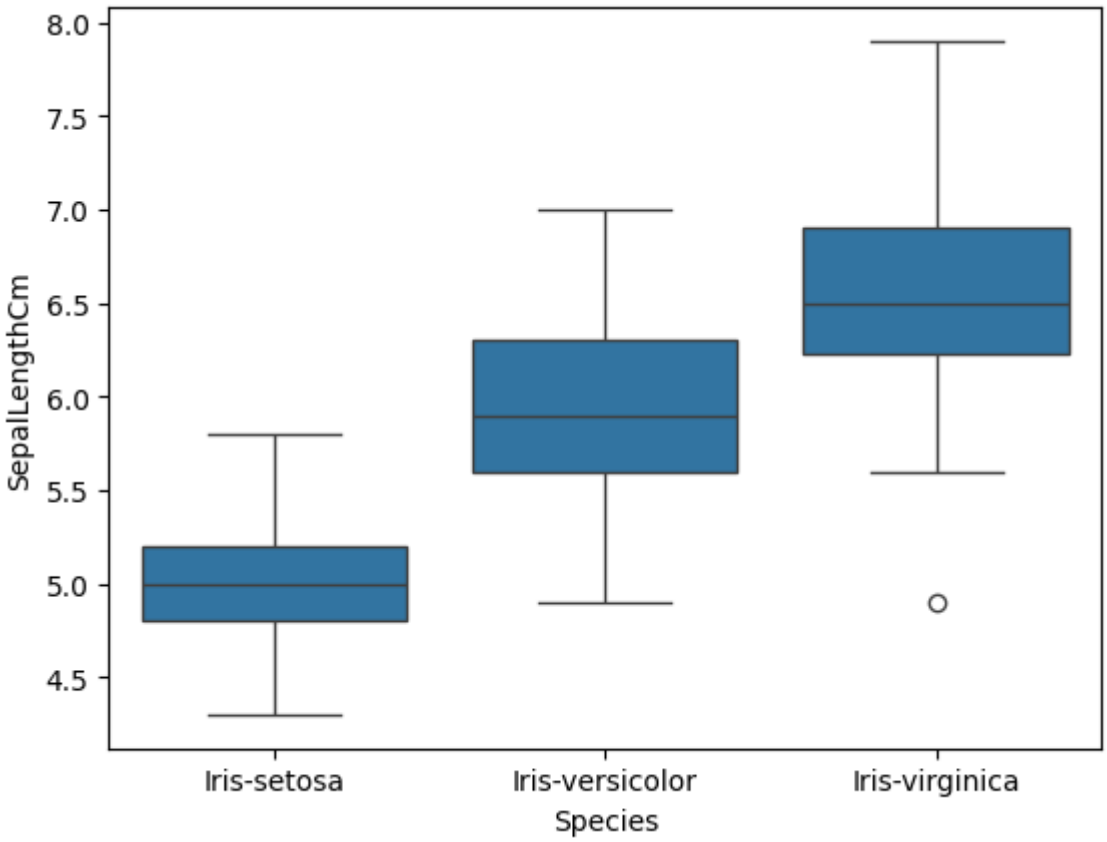
```
        sns.scatterplot(x='SepalLengthCm' , y='SepalWidthCm' , data=df, hue
='Species' )
        plt.legend(bbox_to_anchor=(1 , 1), loc=2)
        plt.show()
        sns.scatterplot(x='PetalLengthCm' , y='PetalWidthCm' , data=df, hue ='Species'
 )
        plt.legend(bbox_to_anchor=(1 , 1), loc=2)
        plt.show()
        for col in data.columns[ :-1]:
        sns.boxplot(x='Species', y=col, data=df)
        plt.show()
```

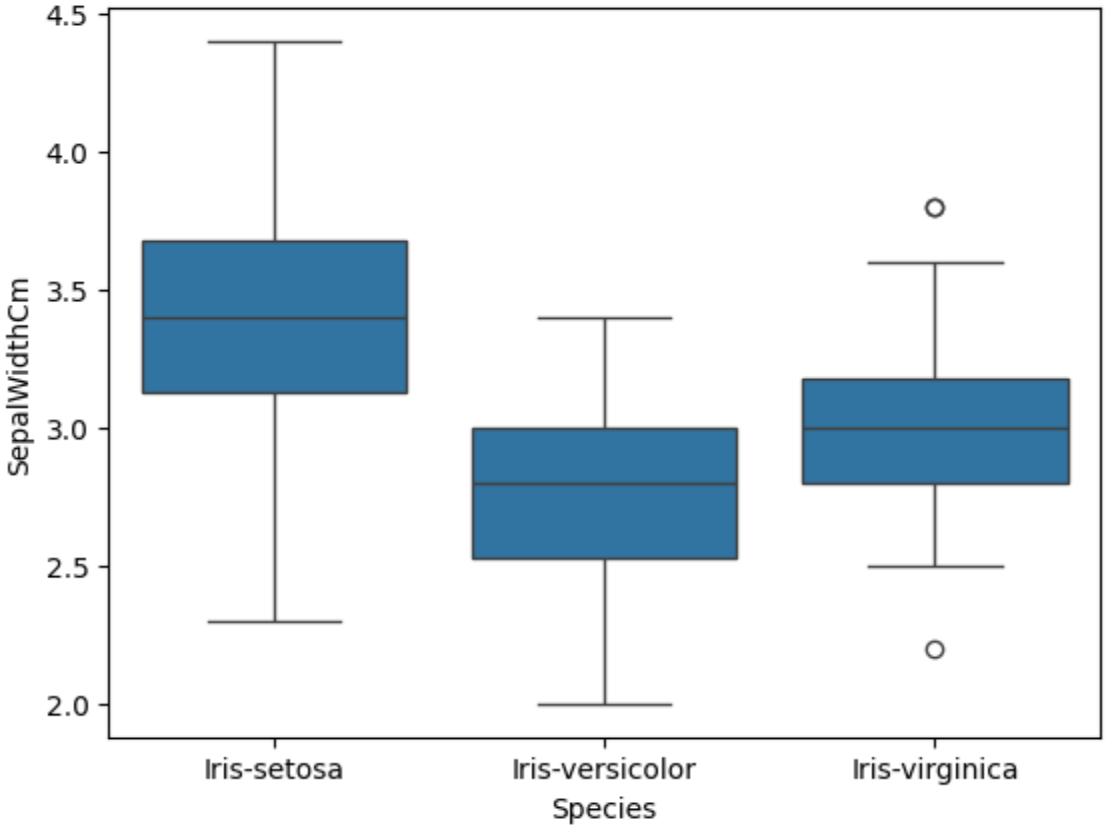
OUTPUT

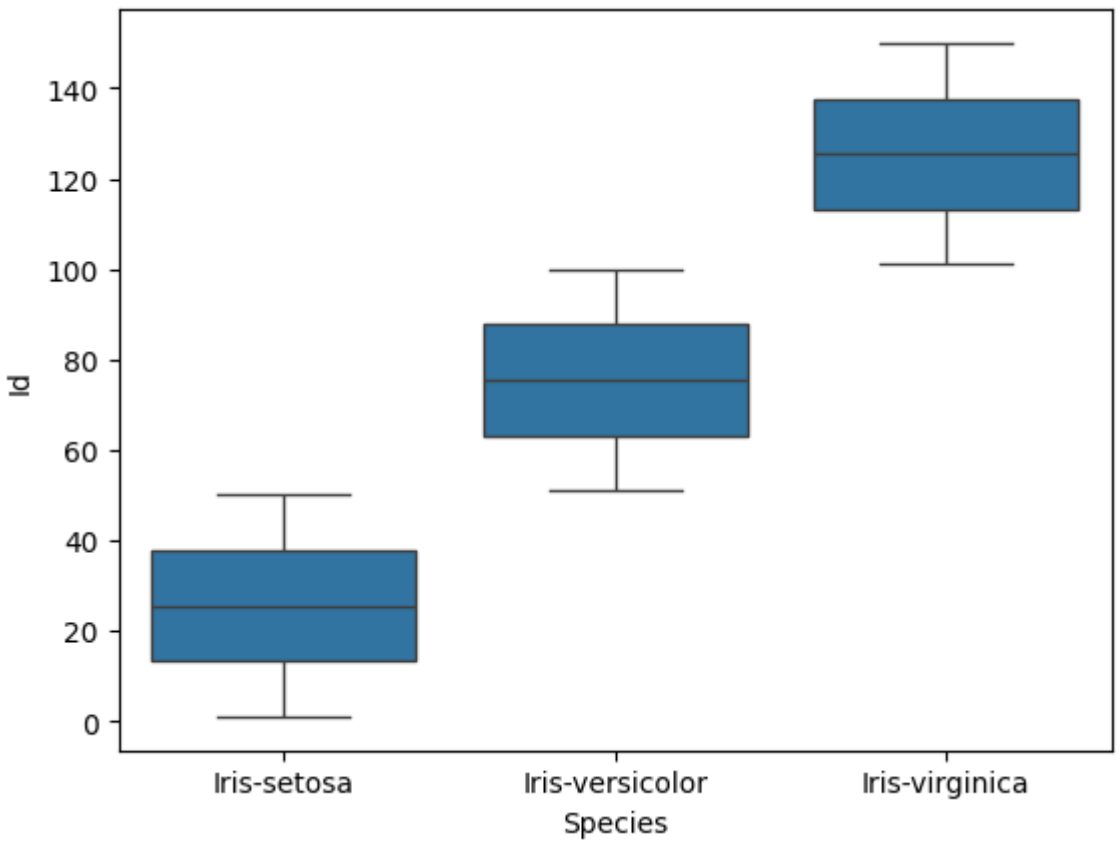


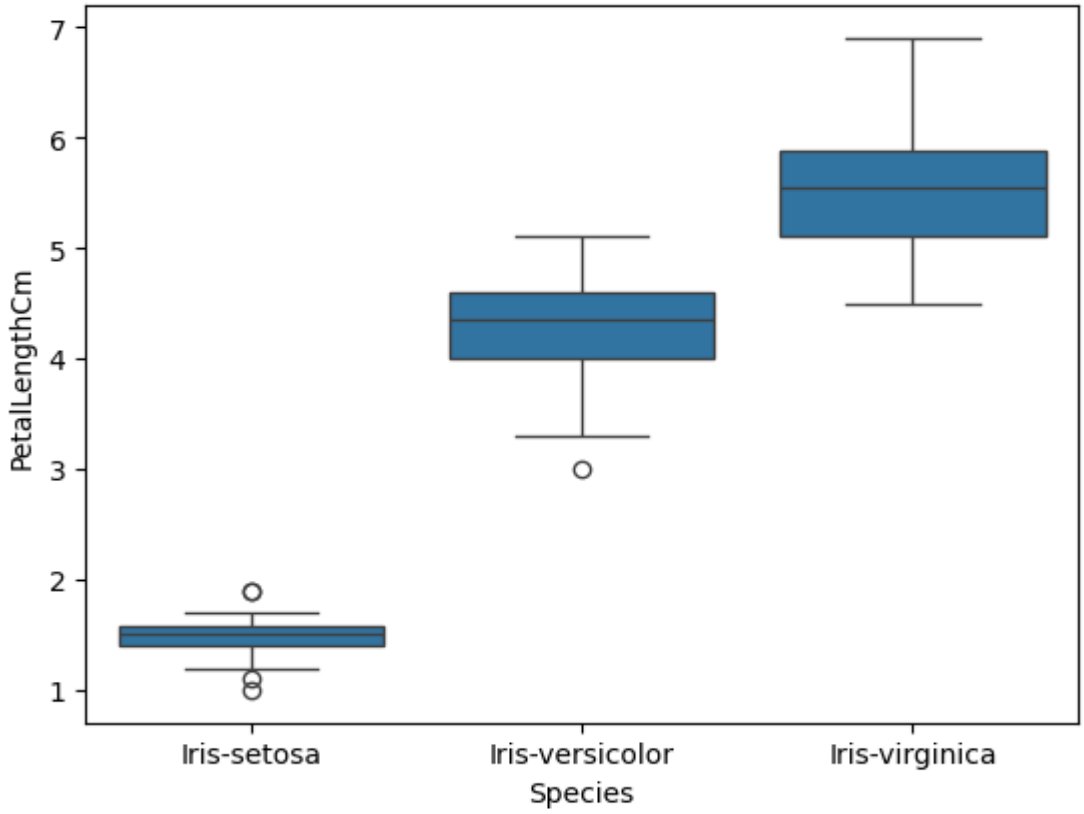

Iris-setosa
Iris-versicolor
Iris-virginica
SepalWidthCm
SepalLengthCm
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

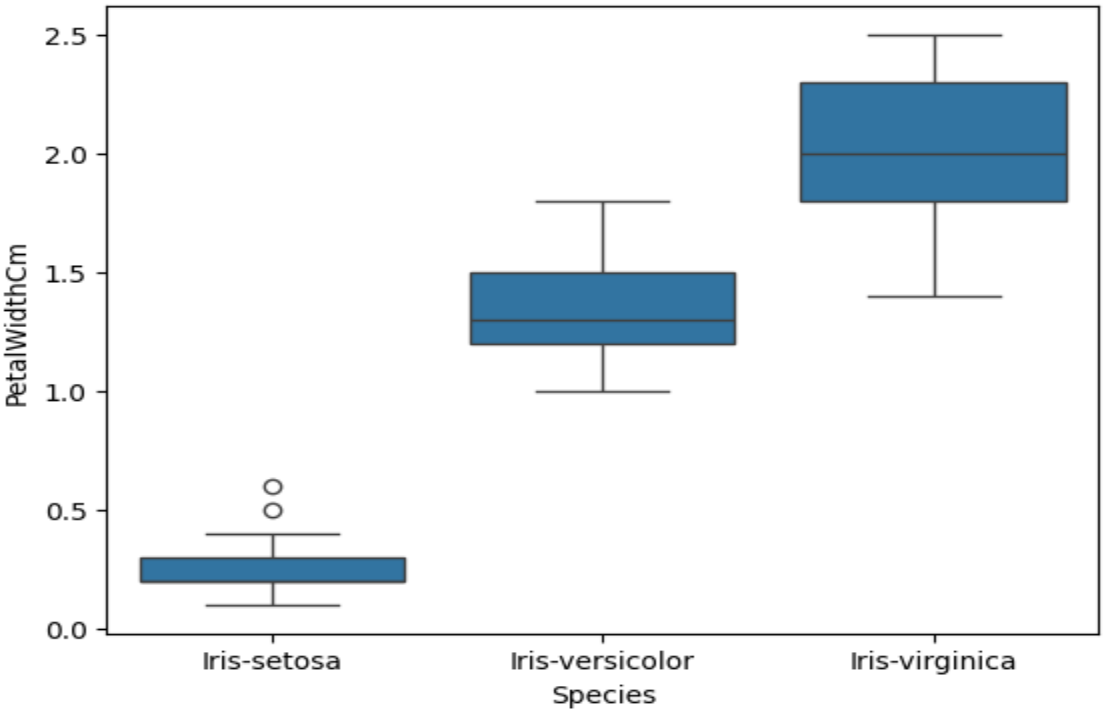

Iris-setosa
Iris-versicolor
Iris-virginica
PetalWidthCm
PetalLengthCm
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7


8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
SepalLengthCm
Iris-setosa
Iris-versicolor
Iris-virginica
Species

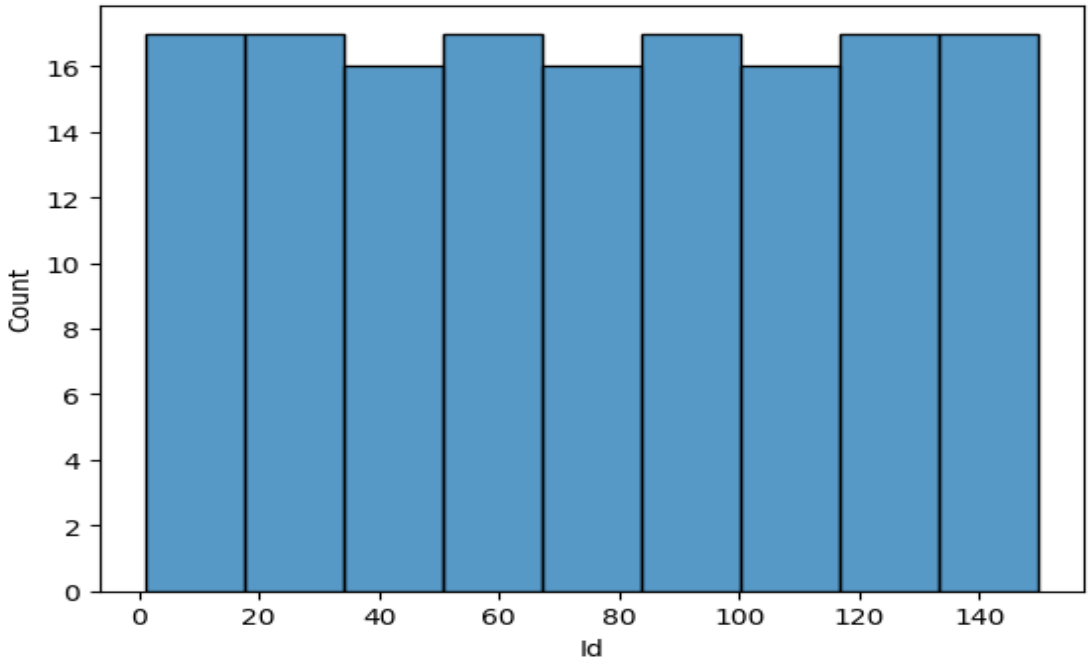

4.5
4.0
3.5
3.0
2.5
2.0
SepalWidthCm
Iris-setosa
Iris-versicolor
Iris-virginica
Species

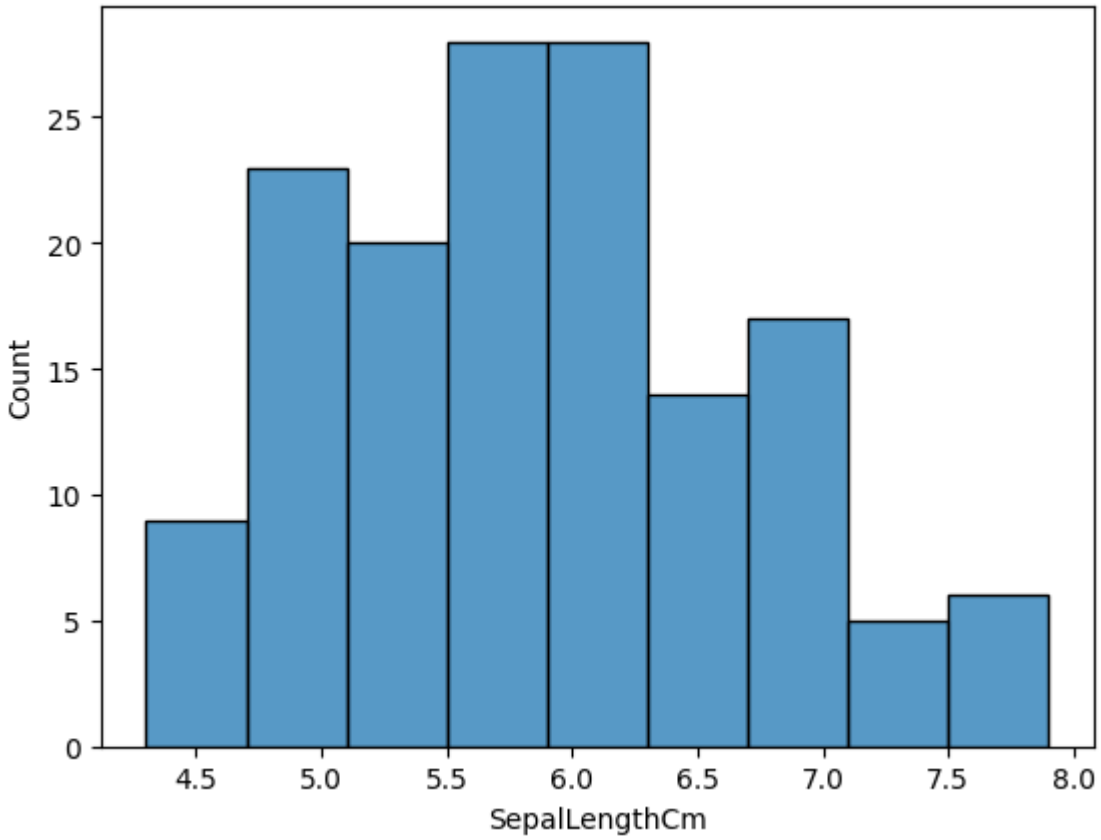

Id
0
20
40
60
80
100
120
140
Iris-setosa
Iris-versicolor
Iris-virginica
Species

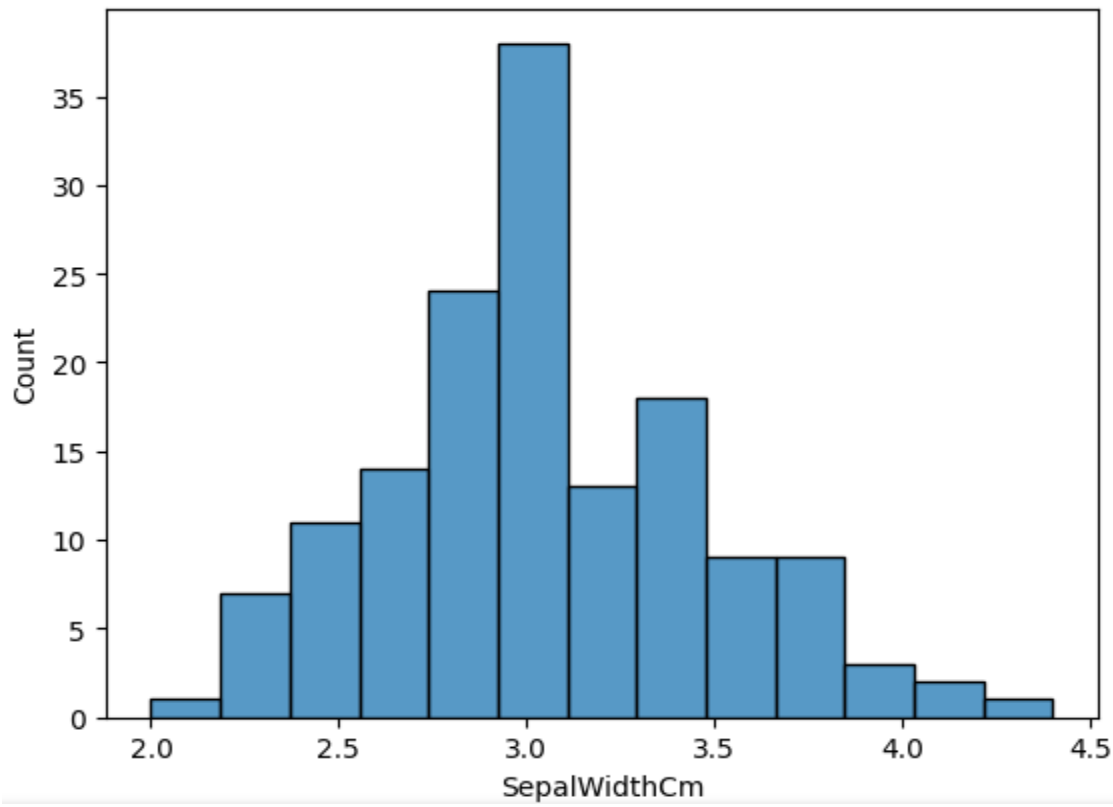

PetalLengthCm
1
2
3
4
5
6
7
Iris-setosa
Iris-versicolor
Iris-virginica
Species

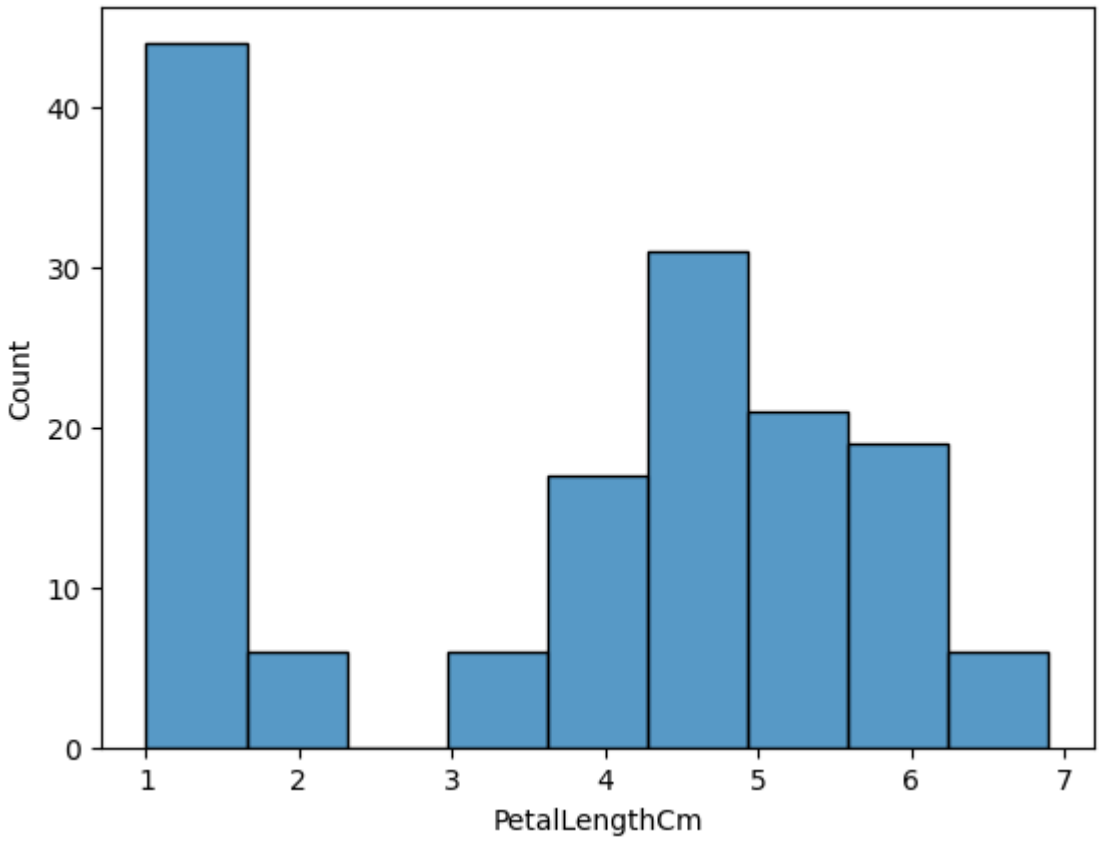


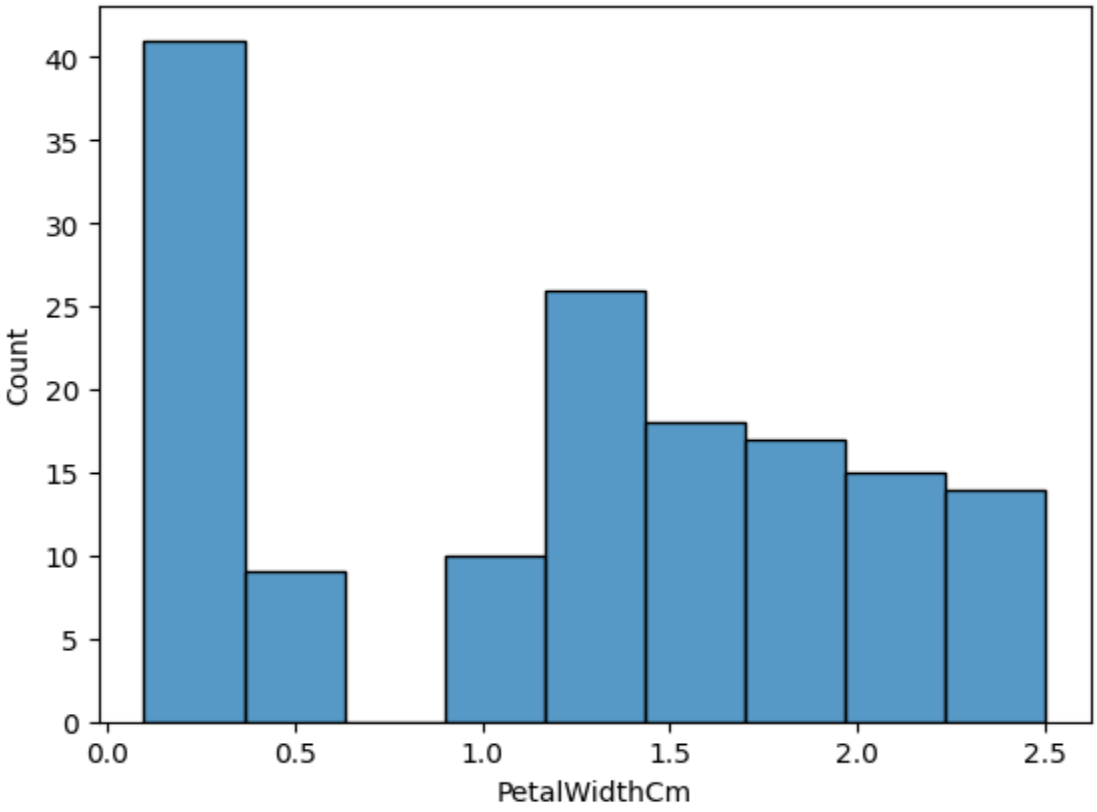

```
for col in data.columns[ :-1]:
sns.histplot(x=col , data=df)
plt.show()
```


Count
Id
0
2
4
6
8
10
12
14
16
0
20
40
60
80
100
120
140

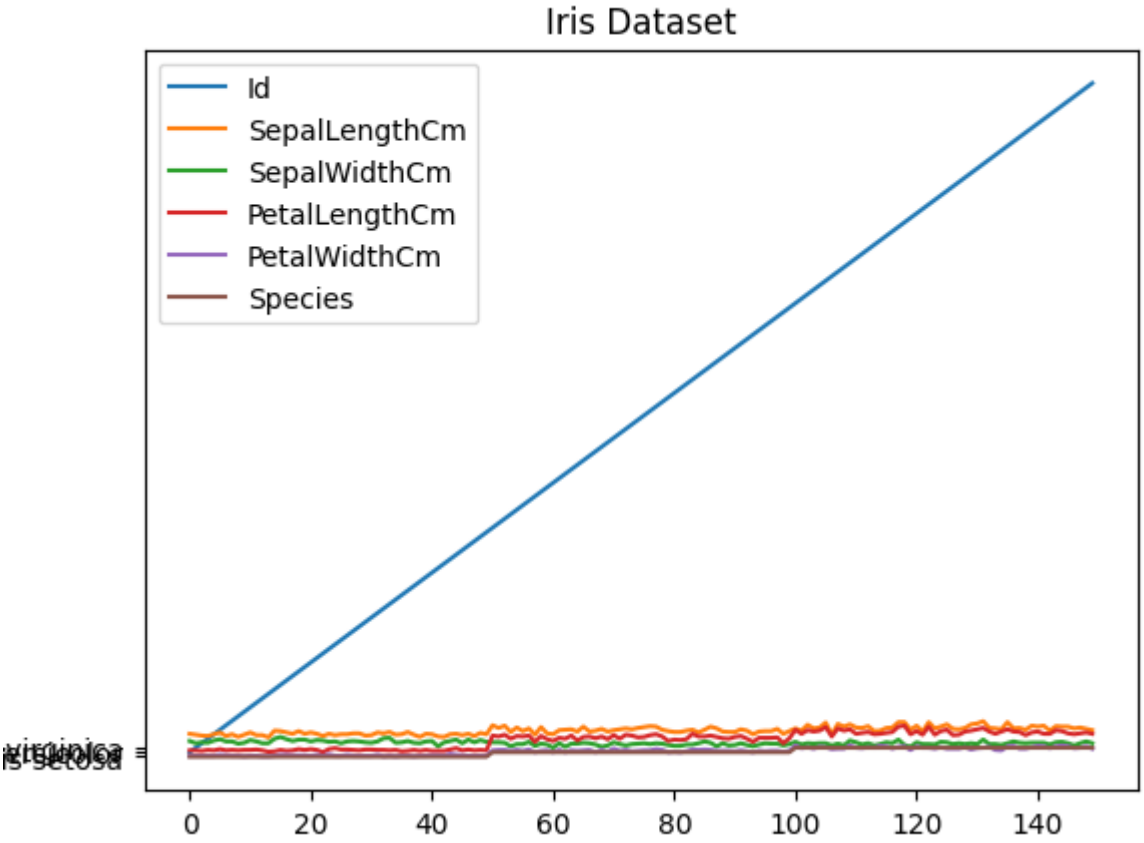

Count
SepalLengthCm
0
5
10
15
20
25
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

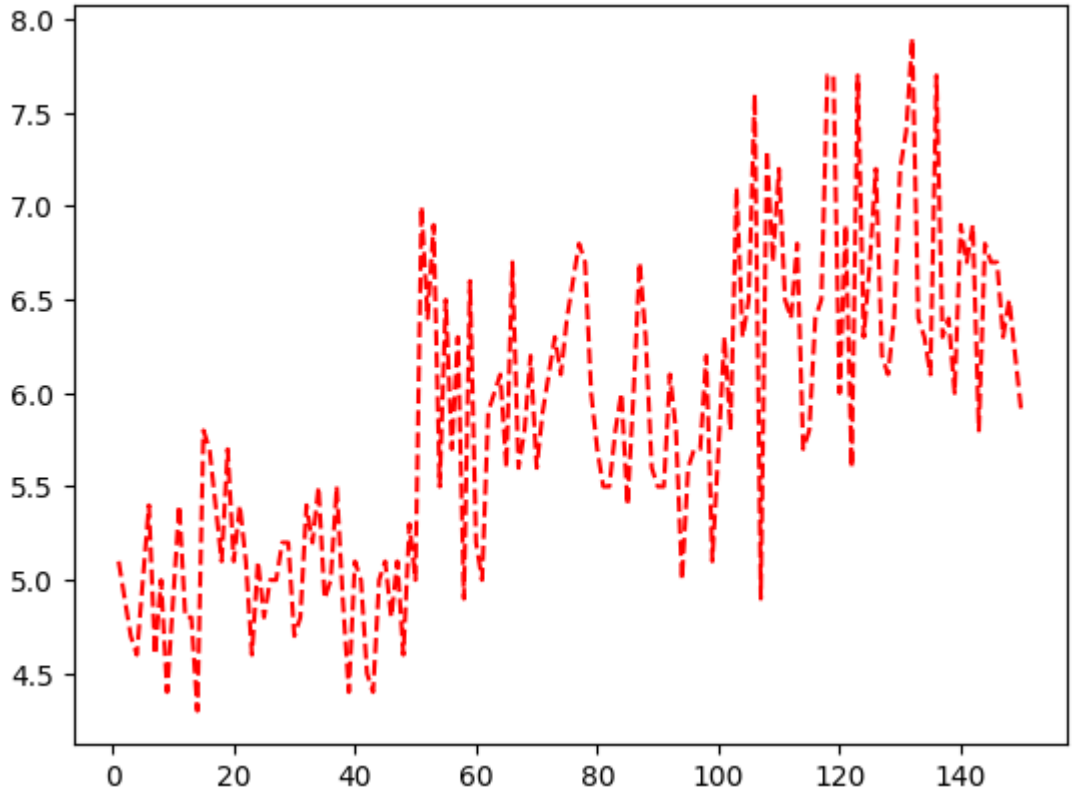

Count
0
5
10
15
20
25
30
35
2.0
2.5
3.0
3.5
4.0
4.5
SepalWidthCm

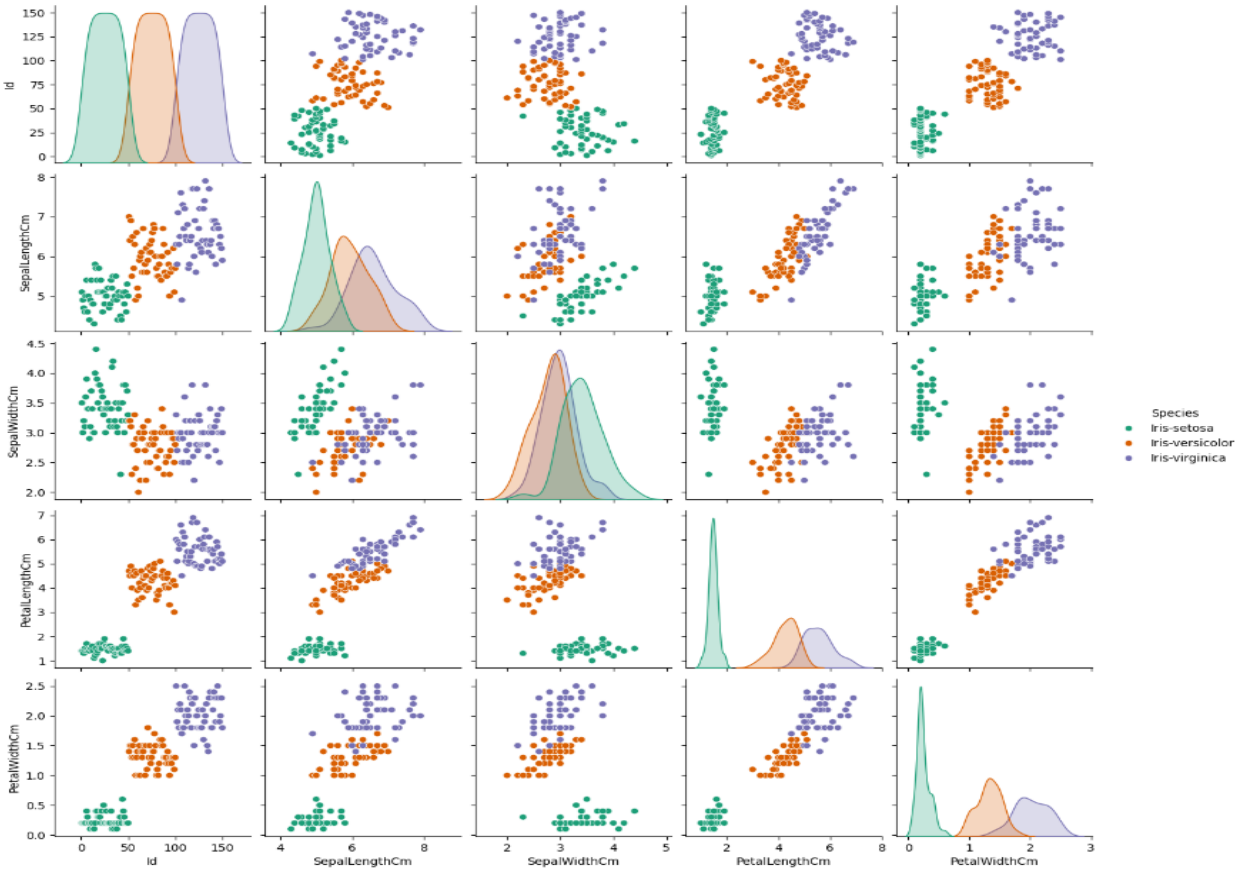

Count
0
10
20
30
40
1
2
3
4
5
6
7
PetalLengthCm




```
columns = df.columns
x_data = range(0, df.shape[0])
fig, ax = plt.subplots()
for column in columns:
ax.plot(x_data, df[column], label=column)
ax.set_title('Iris Dataset')
ax.legend()
plt.plot(df.Id, df['SepalLengthCm'], "r--")
plt.show()
sns.pairplot(data=df, hue="Species" , palette="Dark2")
```


Iris Dataset
Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm
Species
0
20
40
60
80
100
120
140


8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
0
20
40
60
80
100
120
140



Plotting the violinplot

sns.violinplot(x='Species', y='SepalLengthCm', data=df)

plt.show()

sns.violinplot(x='Species', y='SepalWidthCm', data=df)

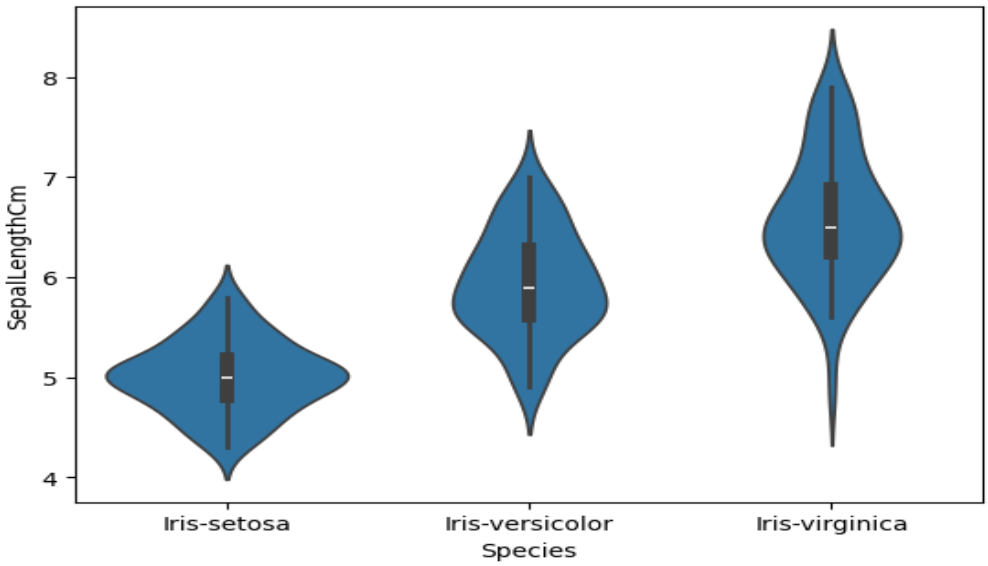
plt.show()

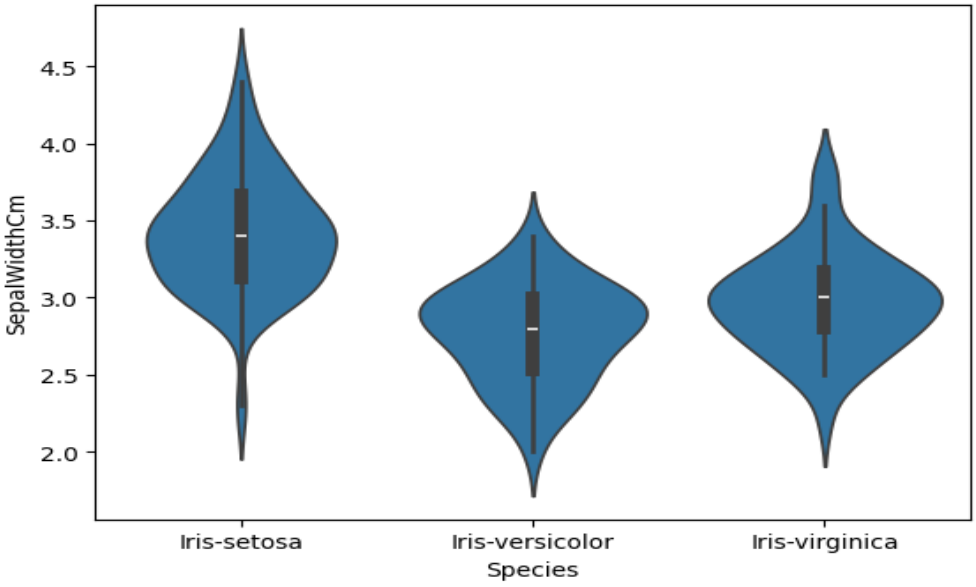
sns.violinplot(x='Species', y='PetalLengthCm', data=df)

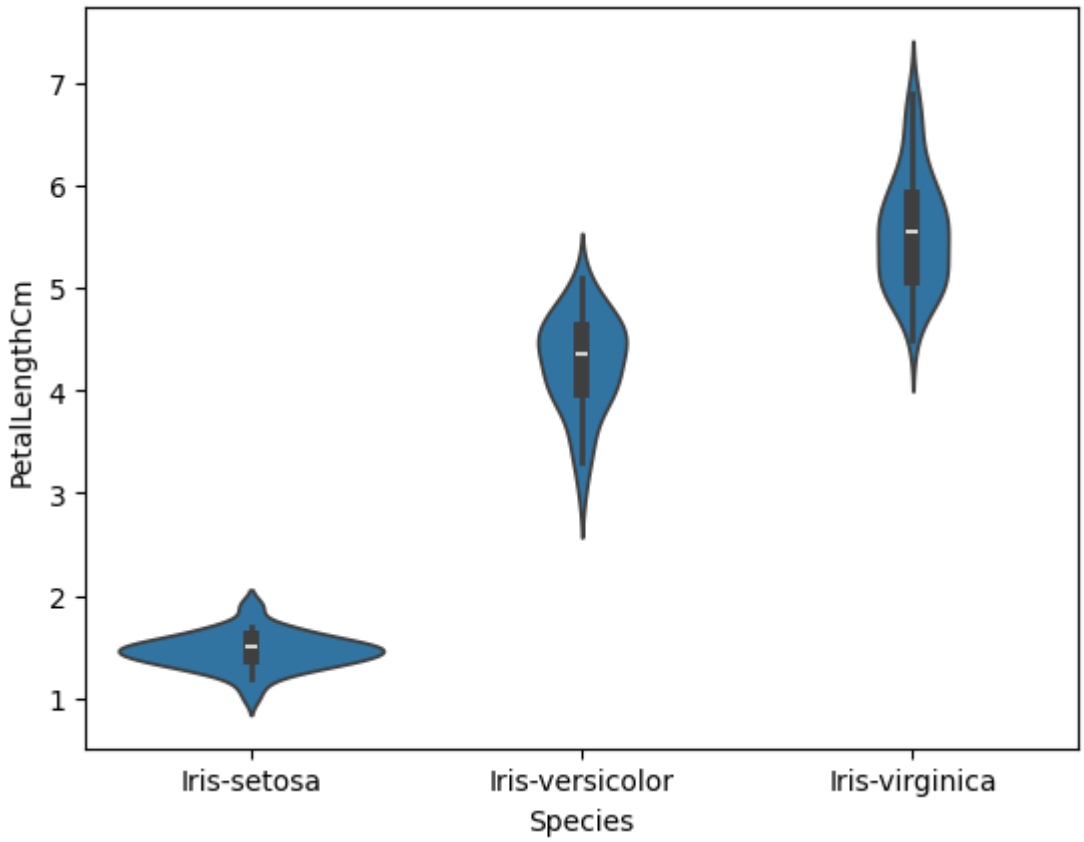
plt.show()

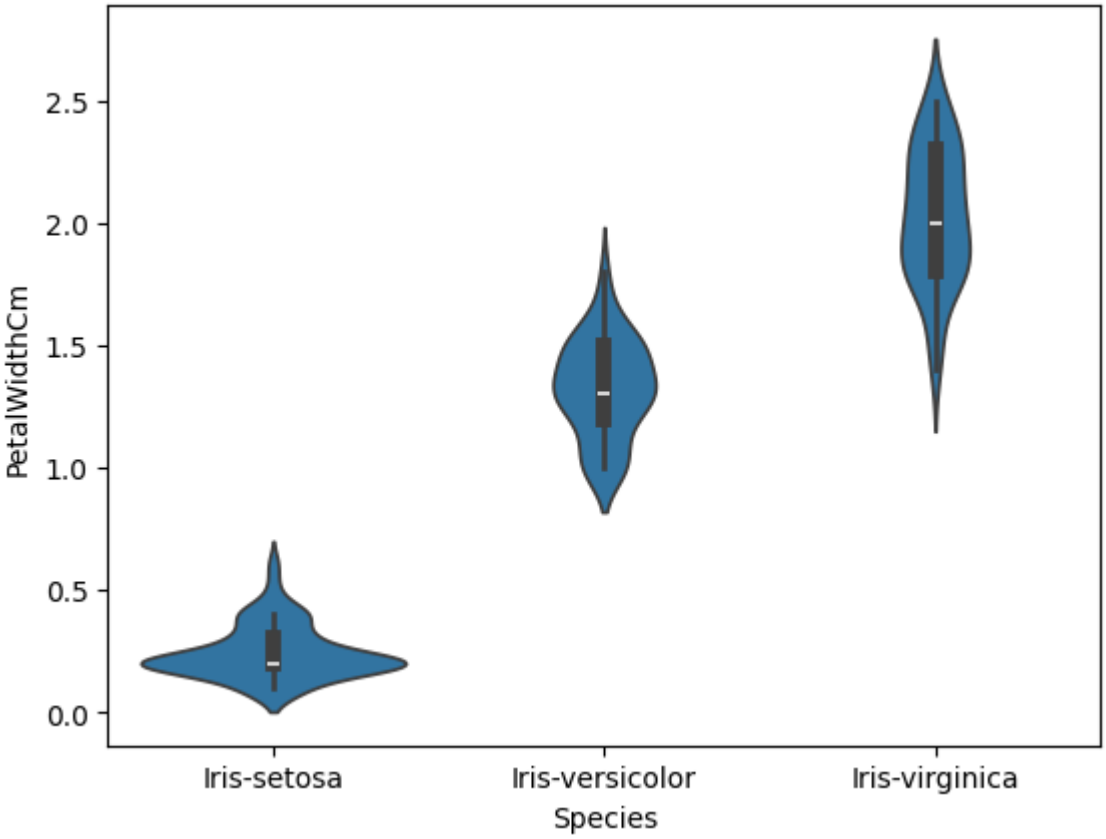
sns.violinplot(x='Species', y='PetalWidthCm', data=df)

plt.show()


SepalLengthCm
8
7
6
5
4
Iris-setosa
Iris-versicolor
Iris-virginica
Species

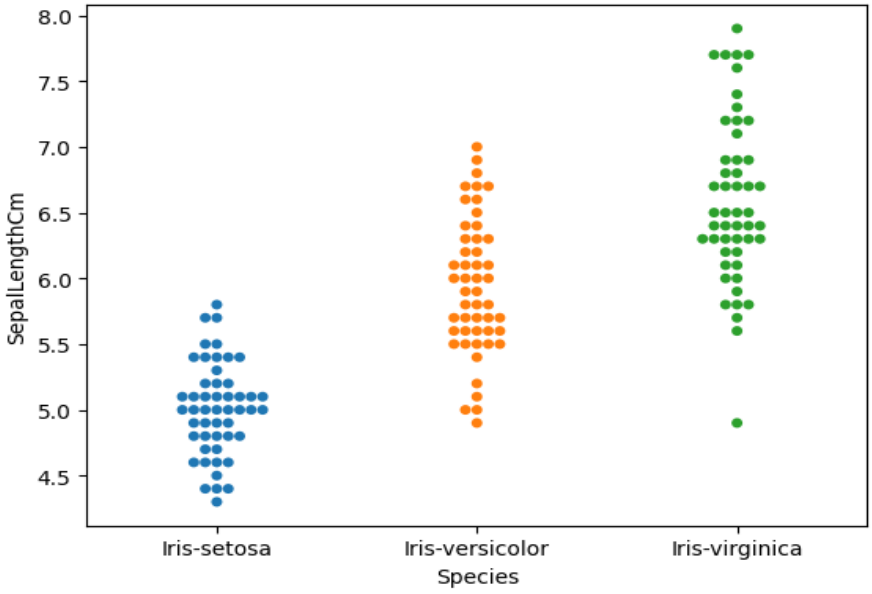

SepalWidthCm
4.5
4.0
3.5
3.0
2.5
2.0
Iris-setosa
Iris-versicolor
Iris-virginica
Species

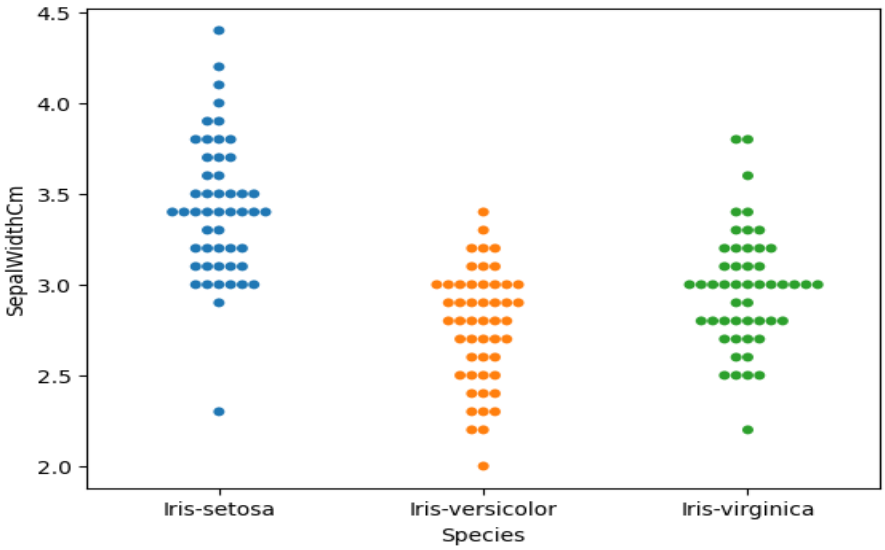

PetalLengthCm
7
6
5
4
3
2
1
Iris-setosa
Iris-versicolor
Iris-virginica
Species

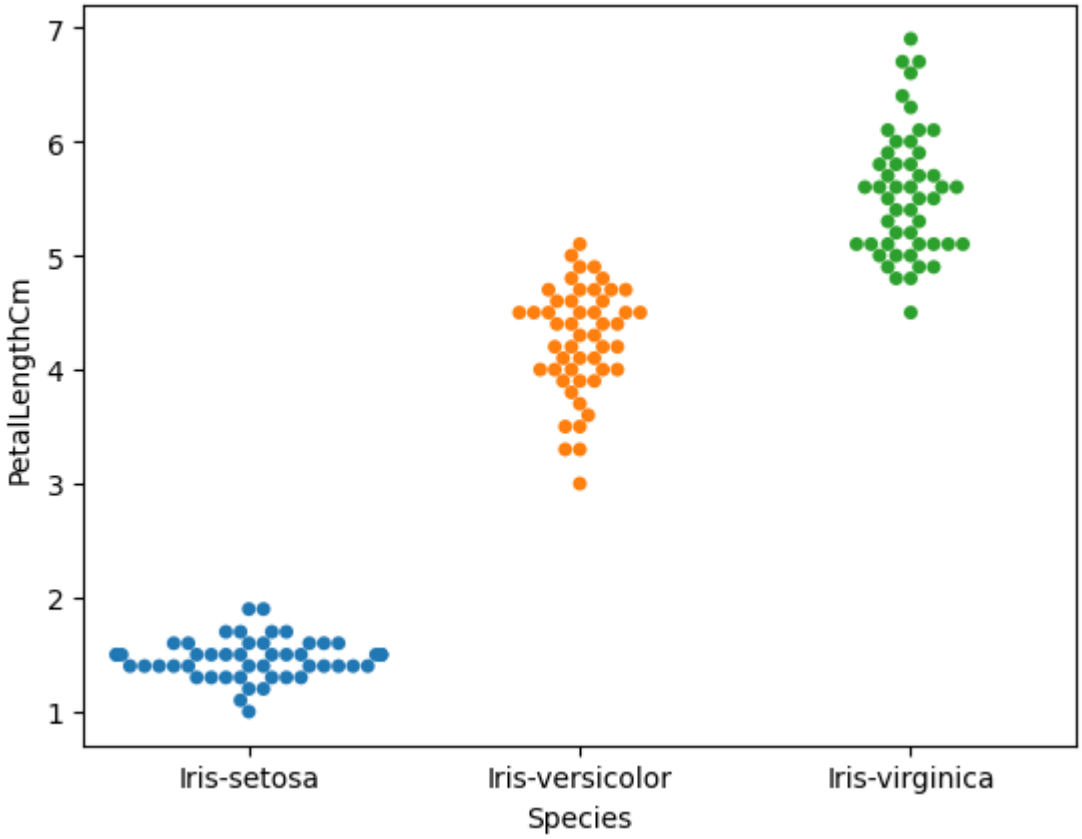

PetalWidthCm
2.5
2.0
1.5
1.0
0.5
0.0
Iris-setosa
Iris-versicolor
Iris-virginica
Species

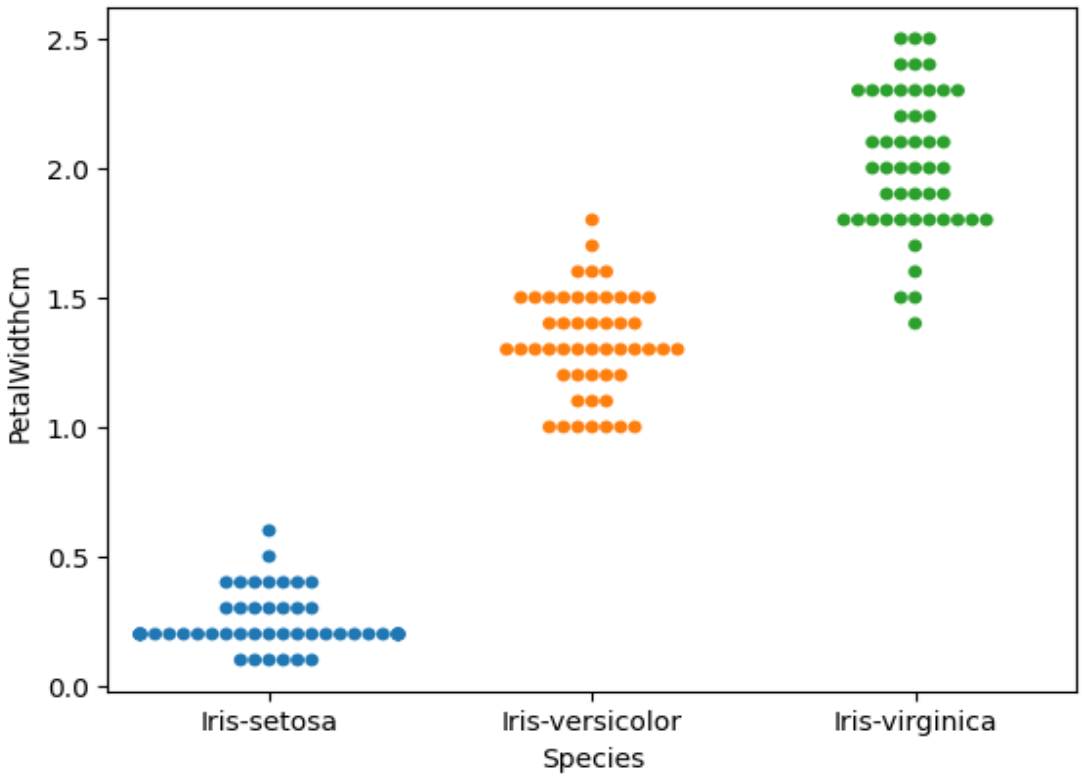
Plotting the swarmplot

```
sns.swarmplot(x='Species', y='SepalLengthCm', data=df , hue="Species")
plt.show()
sns.swarmplot(x='Species', y='SepalWidthCm', data=df, hue="Species")
plt.show()
sns.swarmplot(x='Species', y='PetalLengthCm', data=df, hue="Species")
plt.show()
sns.swarmplot(x='Species', y='PetalWidthCm', data=df , hue="Species")
plt.show()
```

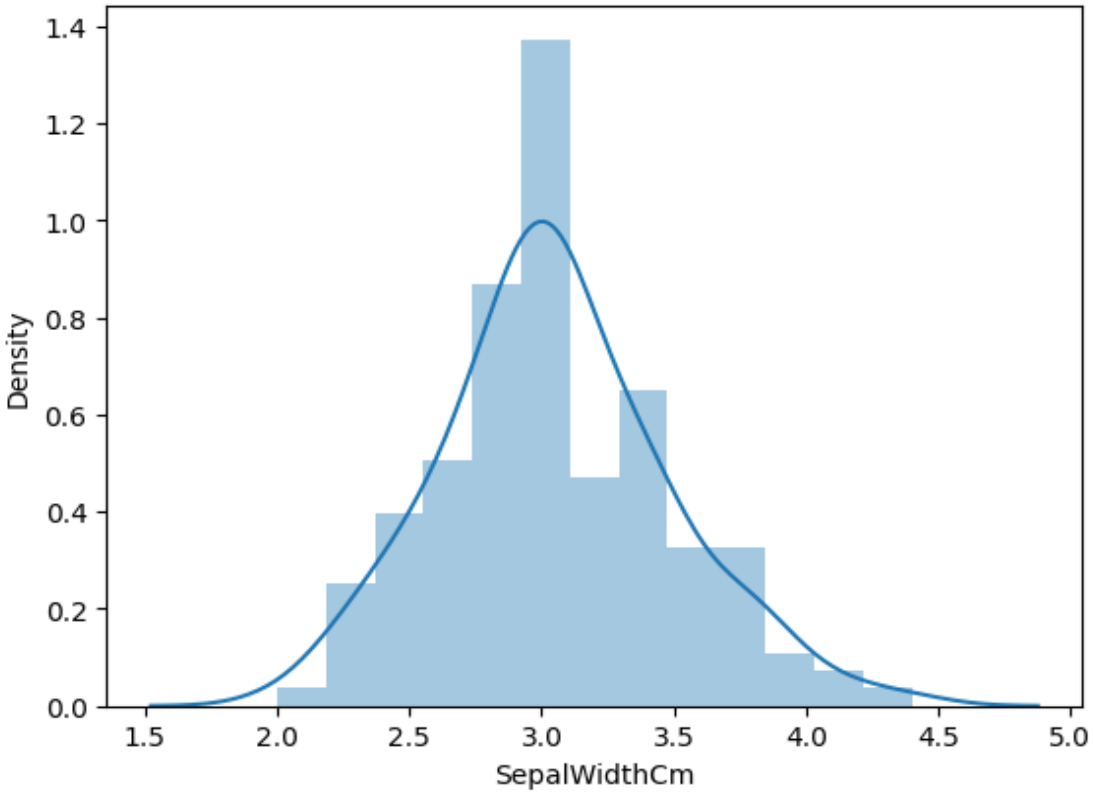


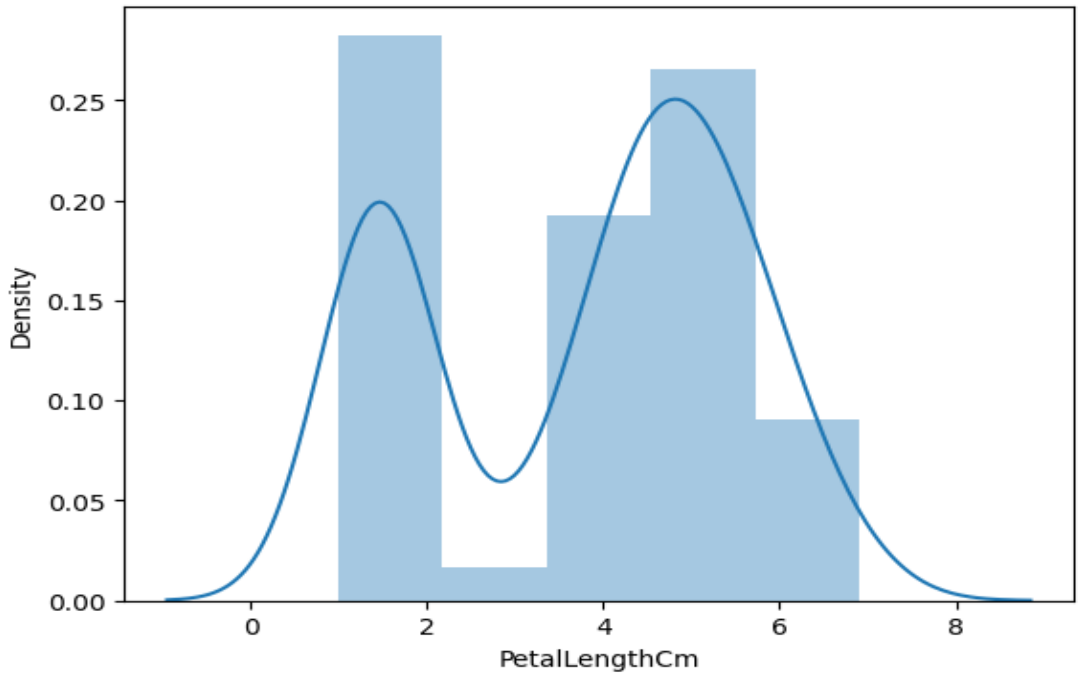

4.5
4.0
3.5
3.0
2.5
2.0
SepalWidthCm
Iris-setosa
Iris-versicolor
Iris-virginica
Species

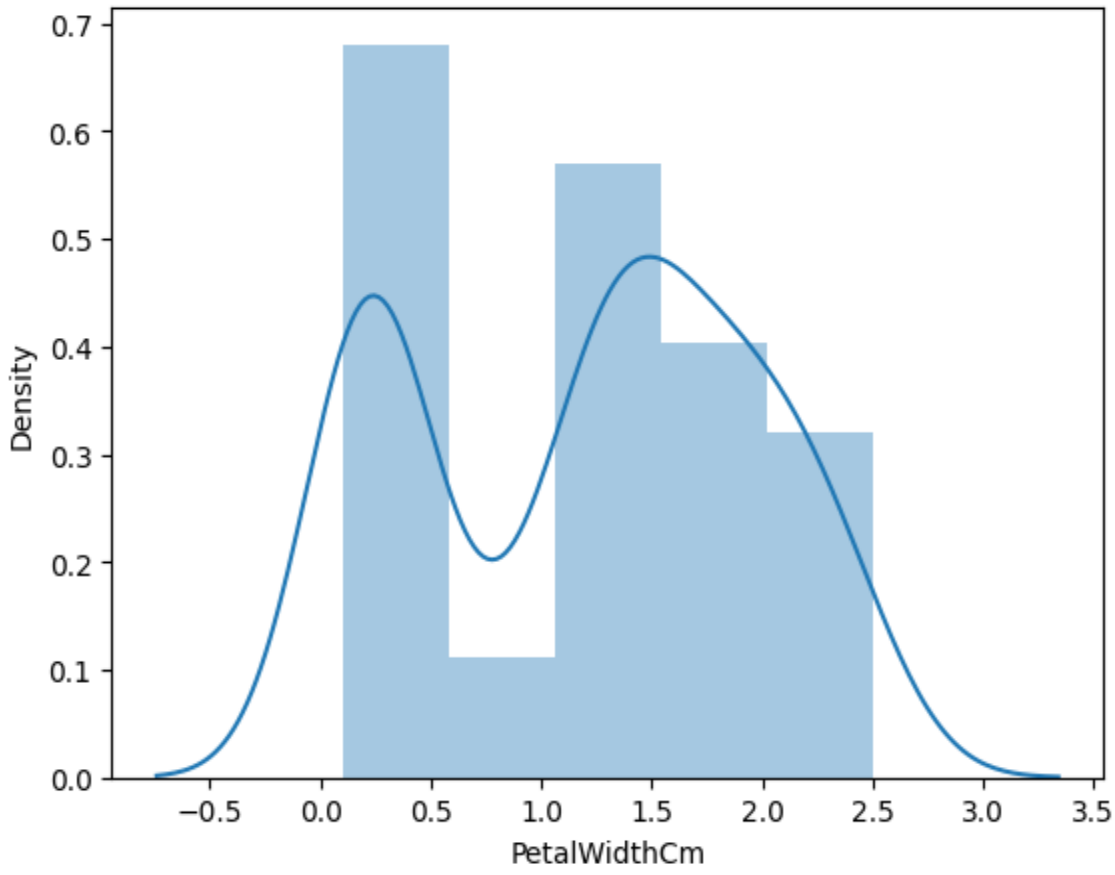

7
6
5
4
3
2
1
PetalLengthCm
Iris-setosa
Iris-versicolor
Iris-virginica
Species




```
sns.distplot(df['SepalWidthCm'])
plt.show()
sns.distplot(df['PetalLengthCm'])
plt.show()
sns.distplot(df['PetalWidthCm'])
plt.show()
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
sns.distplot(df['SepalWidthCm'])
plt.show()
sns.distplot(df['PetalLengthCm'])
plt.show()
sns.distplot(df['PetalWidthCm'])
plt.show()
```

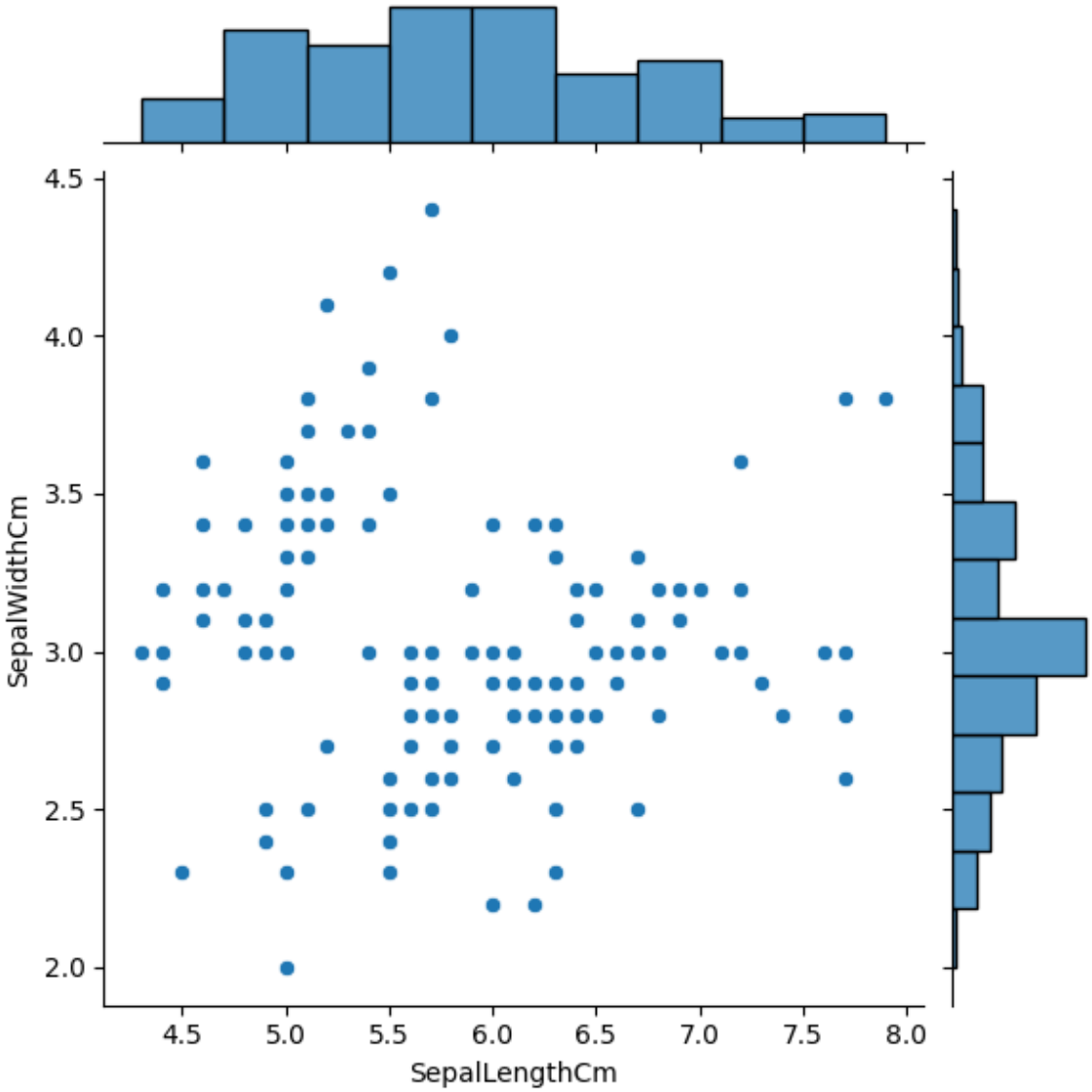

Density
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
SepalWidthCm

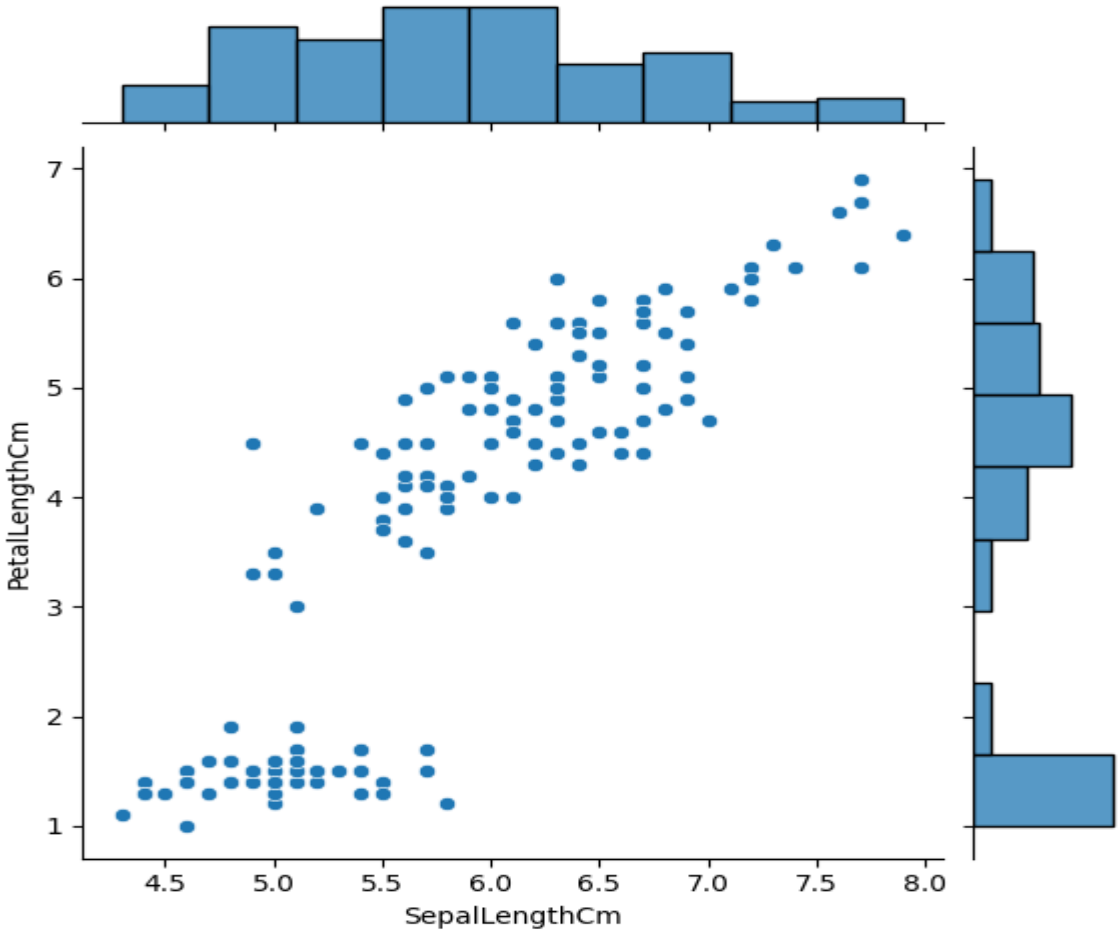

Density
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
PetalLengthCm

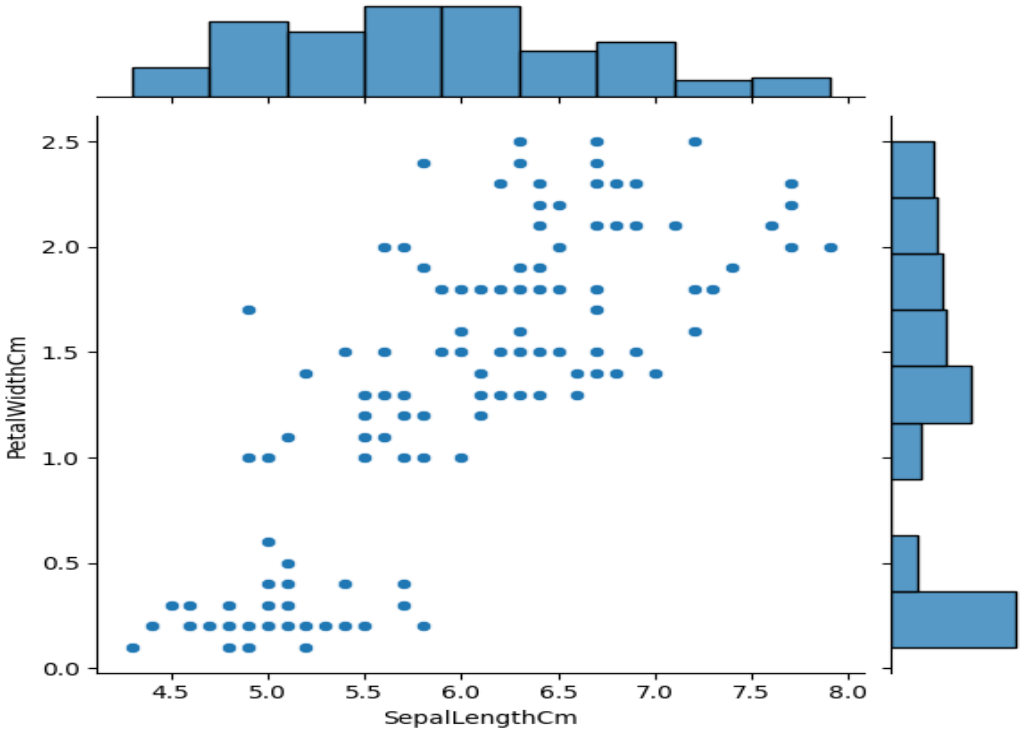


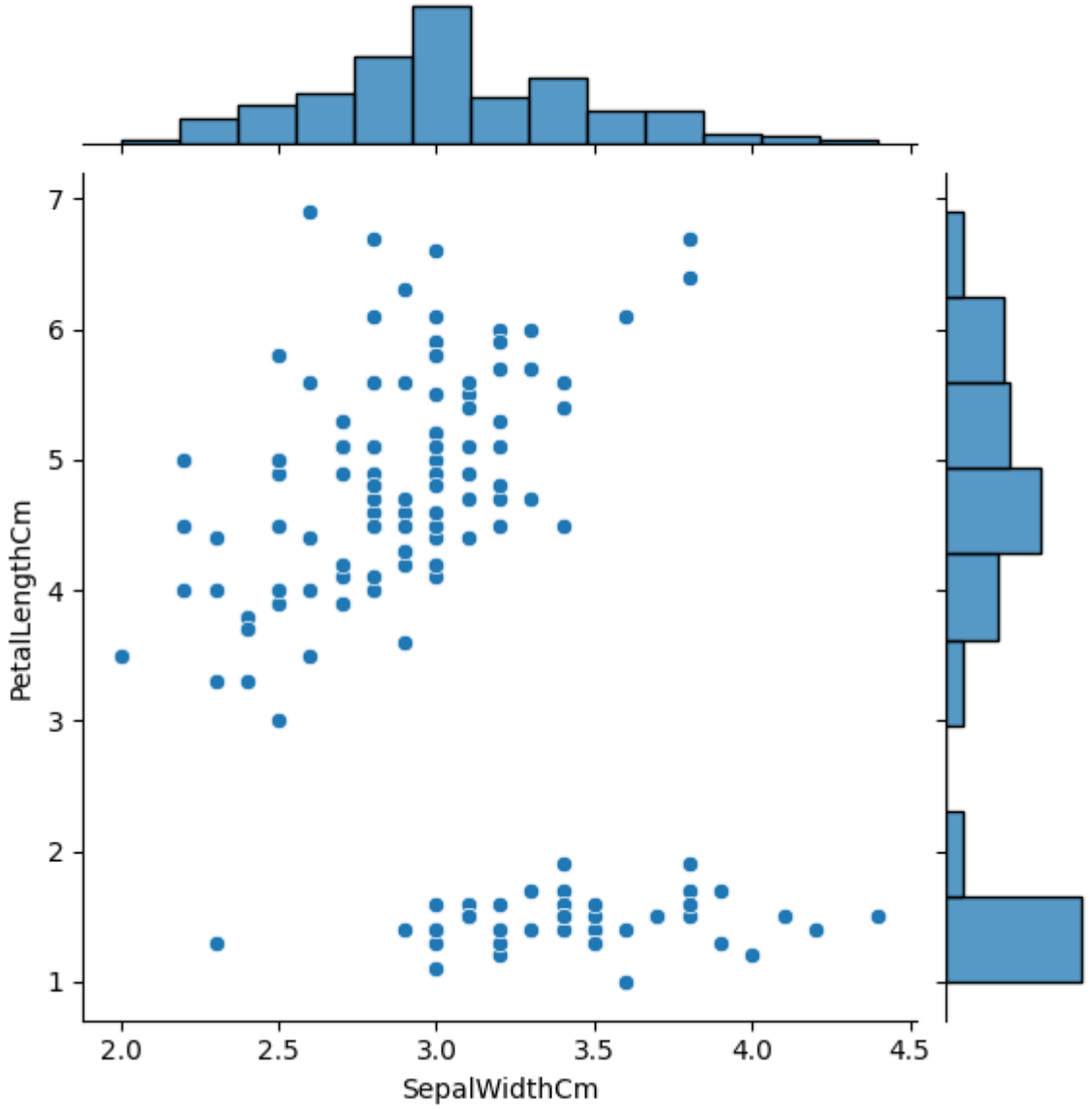
Plotting the jointplot

```
sns.jointplot(x='SepalLengthCm', y='SepalWidthCm', data=df)
plt.show()
sns.jointplot(x='SepalLengthCm', y='PetalLengthCm', data=df)
plt.show()
sns.jointplot(x='SepalLengthCm', y='PetalWidthCm', data=df)
plt.show()
sns.jointplot(x='SepalWidthCm', y='PetalLengthCm', data=df)
plt.show()
```

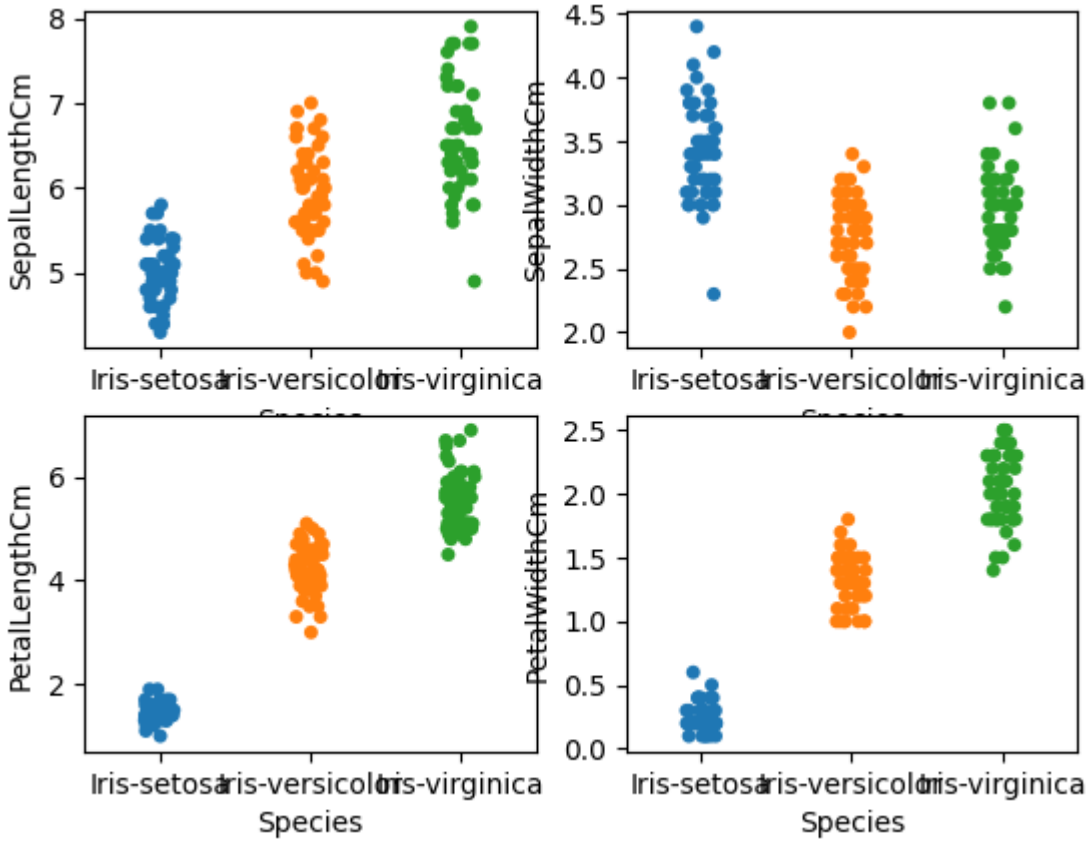

4.5
4.0
3.5
3.0
2.5
2.0
SepalWidthCm
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
SepalLengthCm


PetalLengthCm
SepalLengthCm
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


PetalWidthCm
SepalLengthCm
0.0
0.5
1.0
1.5
2.0
2.5
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0




```
fig, ax = plt.subplots(2, 2)
sns.stripplot(x='Species', y='SepalLengthCm', data=df, ax=ax[0, 0], hue="Species")
sns.stripplot(x='Species', y='SepalWidthCm', data=df, ax=ax[0, 1],hue="Species")
sns.stripplot(x='Species', y='PetalLengthCm', data=df, ax=ax[1, 0],hue="Species")
sns.stripplot(x='Species', y='PetalWidthCm', data=df, ax=ax[1, 1],hue="Species")
plt.show()
```



Bar Chart

- **sns.countplot(x='Species', data=df ,hue='Species')**

  plt.show()

Scatter Plot

- **sns.scatterplot(x='SepalLengthCm' , y='SepalWidthCm' , data=df, hue ='Species')**

  plt.legend(bbox_to_anchor=(1 , 1), loc=2)

  plt.show()

Line Plot

- **columns = df.columns**

```
x_data = range(0, df.shape[0])
fig, ax = plt.subplots()
for column in columns:
ax.plot(x_data, df[column], label=column)
ax.set_title('Iris Dataset')
ax.legend()
```

Histograms

- `for col in data.columns[ :-1]:`

```
sns.histplot(x=col , data=df)
plt.show()
```

Line Histograms (Setosa species,Versicolor species,Virginica species)

- `sns.violinplot(x='Species', y='SepalLengthCm', data=df )`

```
plt.show()
sns.violinplot(x='Species', y='SepalWidthCm', data=df)
plt.show()
sns.violinplot(x='Species', y='PetalLengthCm', data=df)
plt.show()
sns.violinplot(x='Species', y='PetalWidthCm', data=df)
plt.show()
```

Visualization (jointplot)

- `sns.jointplot(x='SepalLengthCm', y='SepalWidthCm', data=df)`

```
plt.show()
sns.jointplot(x='SepalLengthCm', y='PetalLengthCm', data=df)
plt.show()
```

```
sns.jointplot(x='SepalLengthCm', y='PetalWidthCm', data=df)
plt.show()
sns.jointplot(x='SepalWidthCm', y='PetalLengthCm', data=df)
plt.show()
```

Conclusion

Exploratory Data Analysis is not merely a preliminary step in the data analysis workflow; it is a fundamental and integral part of the entire process. From ensuring data quality to guiding feature engineering and model selection, EDA serves as the compass that steers analysts through the intricate landscape of data.

Embracing the significance of EDA not only facilitates a deeper understanding of the data but also sets the stage for more robust, accurate, and actionable insights in the dynamic field of data science.