Exploratory Data Analysis on Iris Dataset

```
In [ ]:
```

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a method used to understand and explore data using visual and statistical techniques.

With EDA, we can:

Get a summary of data (mean, median, etc.)

Find and handle missing or duplicate values

Detect outliers (unusual data points)

Discover patterns and trends in the datase

```
In [ ]:
```

Iris Dataset

The Iris dataset is one of the most famous datasets in data science — it’s often called the “Hello World” of data science.

It contains information about iris flowers, with the following columns:

Sepal Length

Sepal Width

Petal Length

Petal Width

Species Type (Iris-setosa, Iris-versicolor, Iris-virginica)

Scientists measured these features for three species of the Iris flower to study their differences.

In [ ]:

```

```

Step 1: Importing Required Libraries

In [3]:

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

Explanation:

pandas (pd): Used to handle and analyze datasets (works with rows & columns like Excel).

numpy (np): Helps perform numerical operations (like mean, median, etc.).

matplotlib.pyplot (plt): Used for creating plots and graphs.

seaborn (sns): Built on top of matplotlib, used for stylish and easy visualization.

In [ ]:

```python

```

Step 2: Importing the Dataset

In [6]:

```python
dataset=pd.read_csv(r"C:\Users\avani\Downloads\Iris.csv")
```

Step 3: Checking Dataset Size

In [26]:

```python
dataset.shape # to get the shape of the dataset
```

Out[26]:

```
(150, 6)
```

Step 4: Checking Data Information

In [27]:

```python
dataset.info() # getting information about the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Step 5: Statistical Summary

In [28]:

```python
dataset.describe() # to get a statistical summary of the dataset
```

Out[28]:

	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

Step 6: Checking for Missing Values

In [29]:

```
dataset.isnull().sum() # to check missing values
```

Out[29]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Step 7: Dropping Unnecessary Columns

In [31]:

```
data=dataset.drop_duplicates(subset="Species",)
```

In [32]:

```
data
```

Out[32]:

	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

Step 8 Count the Value

In [34]:

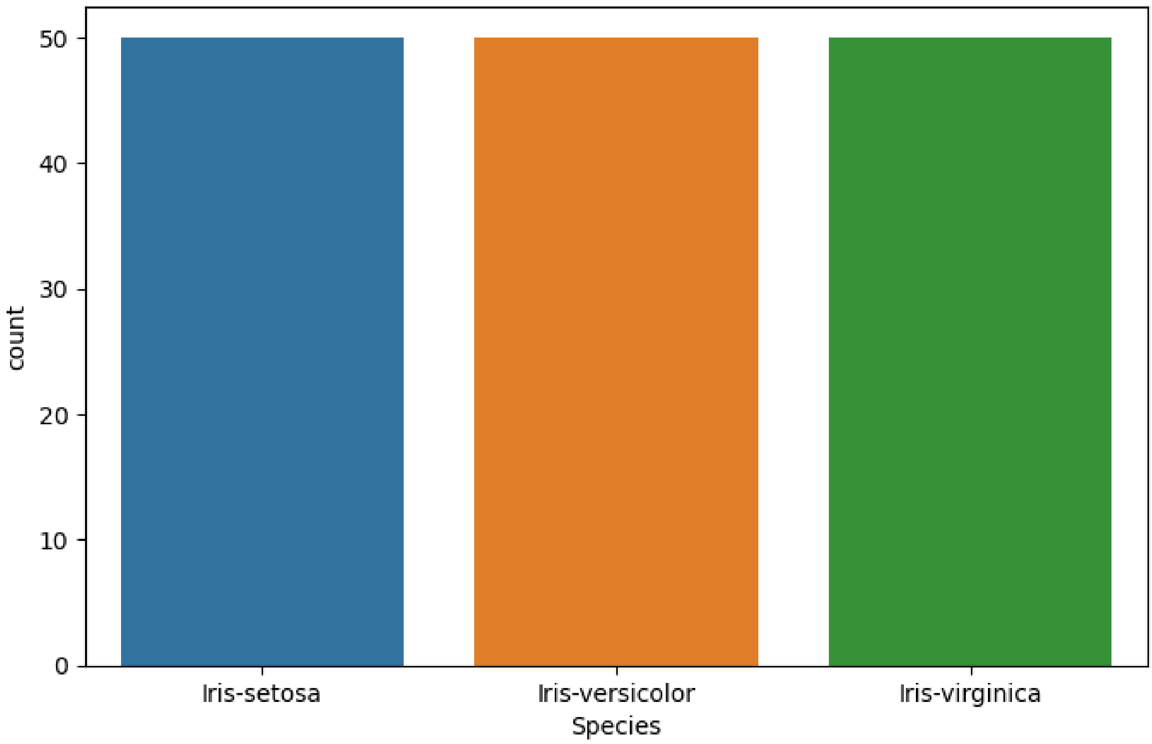
```
dataset.value_counts("Species")
```

Out[34]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

In [ ]:

Step 9: Visualizing Data Distribution

In [44]:
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
sns.countplot(x='Species', data=dataset,hue="Species" )
plt.show()
```



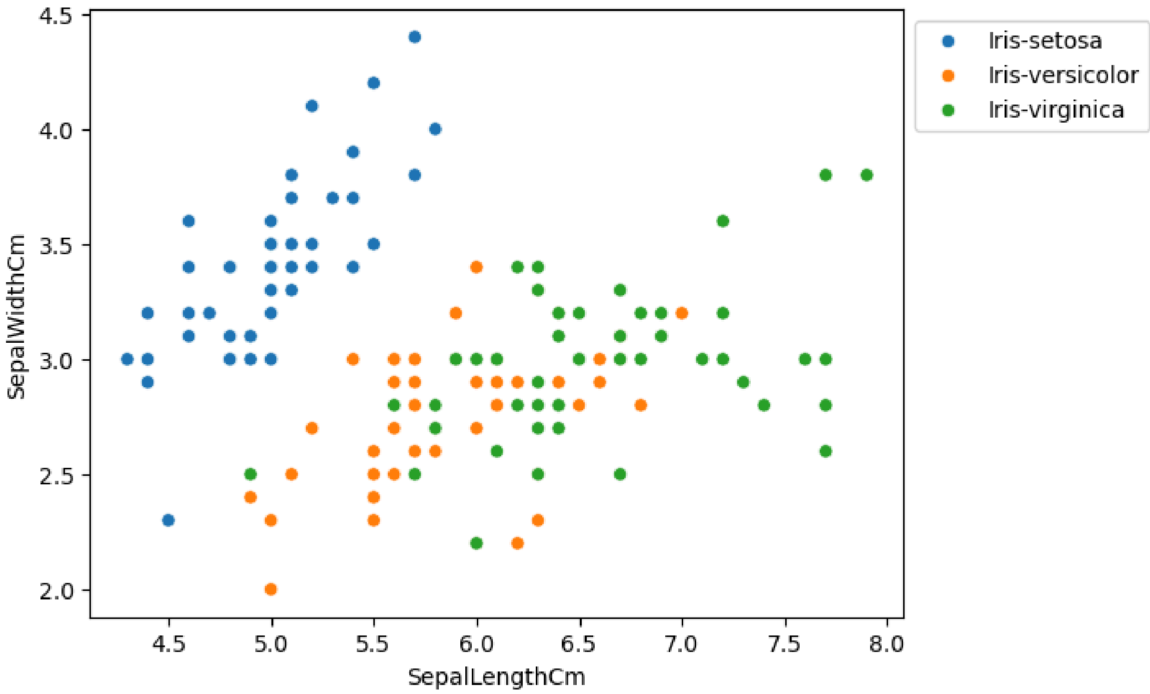
Relation between variables

We will see the relationship between the sepal length and sepal width and also between petal length and petal width.

In [ ]:

Example 1: Comparing Sepal Length and Sepal Width

In [45]:
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.scatterplot(x='SepalLengthCm', y='SepalWidthCm',
                hue='Species', data=dataset, )

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```



From the above plot, we can infer that -

Species Setosa has smaller sepal lengths but larger sepal widths.

Versicolor Species lies in the middle of the other two species in terms of sepal length and width

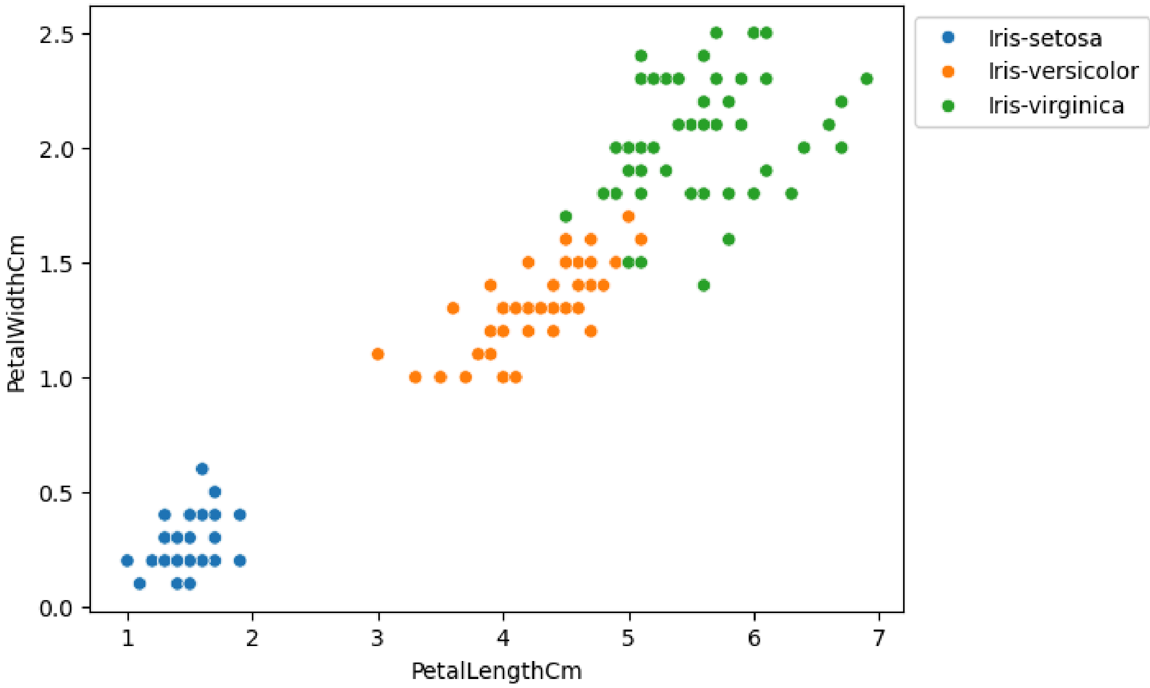
Species Virginica has larger sepal lengths but smaller sepal widths

In [ ]:

Example 2: Comparing Petal Length and Petal Width

In [46]:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.scatterplot(x='PetalLengthCm', y='PetalWidthCm',
                hue='Species', data=dataset, )

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```



From the above plot, we can infer that -

Species Setosa has smaller petal lengths and widths.

Versicolor Species lies in the middle of the other two species in terms of petal length and width

Species Virginica has the largest of petal lengths and widths.

In [ ]:

```

```

Let's plot all the column's relationships using a pairplot. It can be used for multivariate analysis.

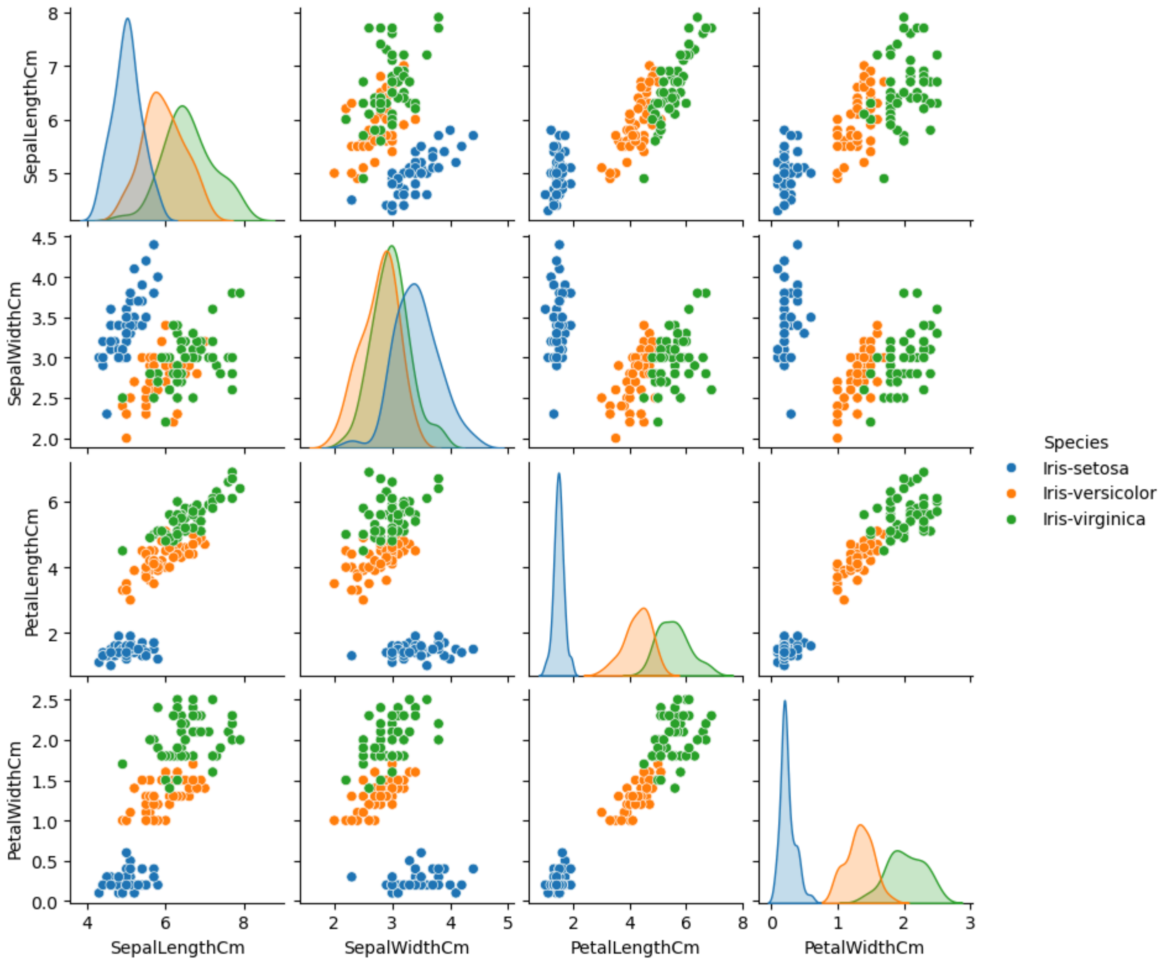
Example:

In [50]:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
sns.pairplot(dataset.drop(['Id'], axis = 1),
             hue='Species', height=2)
```

Out[50]:

```
<seaborn.axisgrid.PairGrid at 0x1f4c4ab6c00>
<Figure size 1000x1000 with 0 Axes>
```



We can see many types of relationships from this plot such as the species Setosa has the smallest of petals widths and lengths. It also has the smallest sepal length but larger sepal widths. Such information can be gathered about any other species.

In [ ]:

Histograms

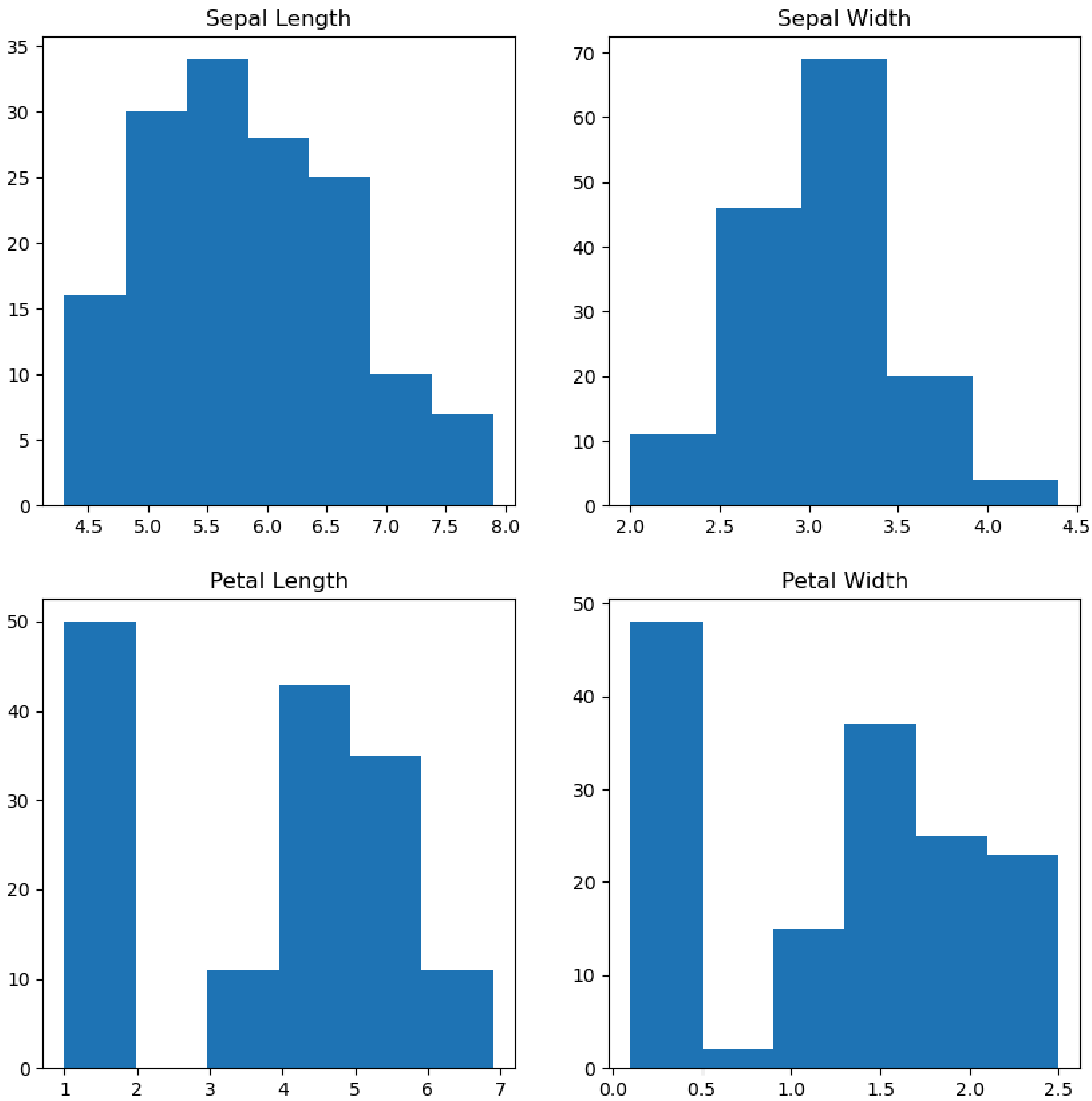
Histograms allow seeing the distribution of data for various columns. It can be used for uni as well as bi-variate analysis.

In [ ]:

Example:

In [51]:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


fig, axes = plt.subplots(2, 2, figsize=(10,10))

axes[0,0].set_title("Sepal Length")
axes[0,0].hist(dataset['SepalLengthCm'], bins=7)

axes[0,1].set_title("Sepal Width")
axes[0,1].hist(dataset['SepalWidthCm'], bins=5);

axes[1,0].set_title("Petal Length")
axes[1,0].hist(dataset['PetalLengthCm'], bins=6);

axes[1,1].set_title("Petal Width")
axes[1,1].hist(dataset['PetalWidthCm'], bins=6);
```



From the above plot, we can see that -

The highest frequency of the sepal length is between 30 and 35 which is between 5.5 and 6

The highest frequency of the sepal Width is around 70 which is between 3.0 and 3.5

The highest frequency of the petal length is around 50 which is between 1 and 2

The highest frequency of the petal width is between 40 and 50 which is between 0.0 and 0.5

In [ ]:

Histograms with Distplot Plot

Distplot is used basically for the univariant set of observations and visualizes it through a histogram i.e. only one observation and hence we choose one particular column of the dataset.

In [ ]:

Example:

In [53]:

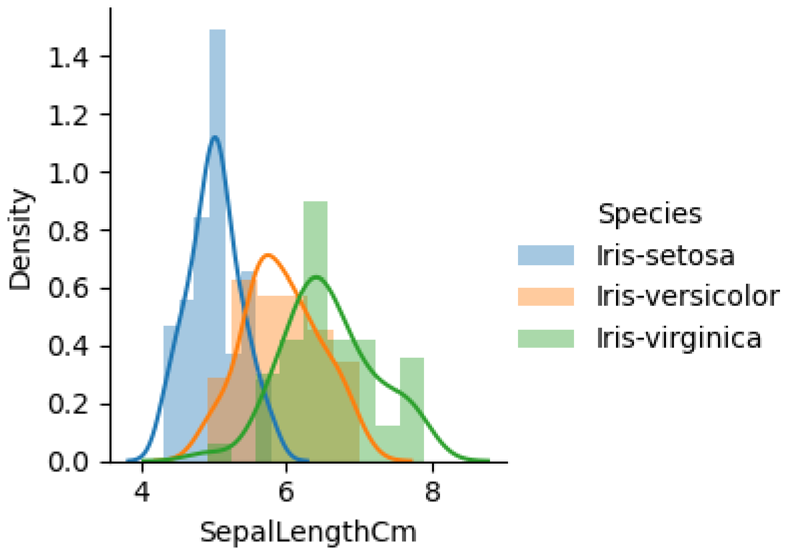
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

plot = sns.FacetGrid(dataset, hue="Species")
plot.map(sns.distplot, "SepalLengthCm").add_legend()
```

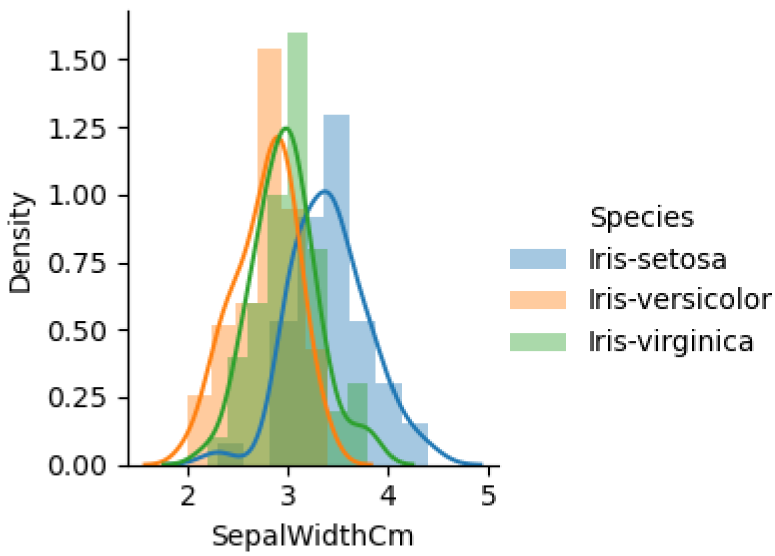
```
plot = sns.FacetGrid(dataset, hue="Species")
plot.map(sns.distplot, "SepalWidthCm").add_legend()

plot = sns.FacetGrid(dataset, hue="Species")
plot.map(sns.distplot, "PetalLengthCm").add_legend()

plot = sns.FacetGrid(dataset, hue="Species")
plot.map(sns.distplot, "PetalWidthCm").add_legend()

plt.show()
```

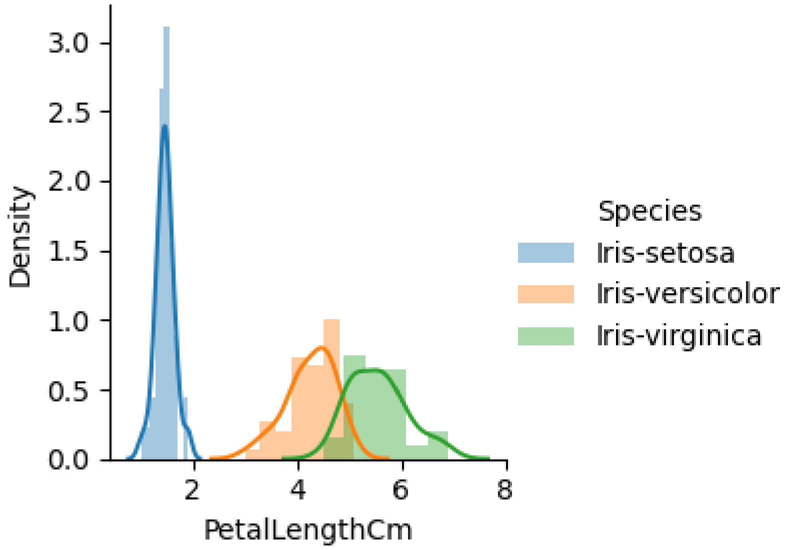
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
```

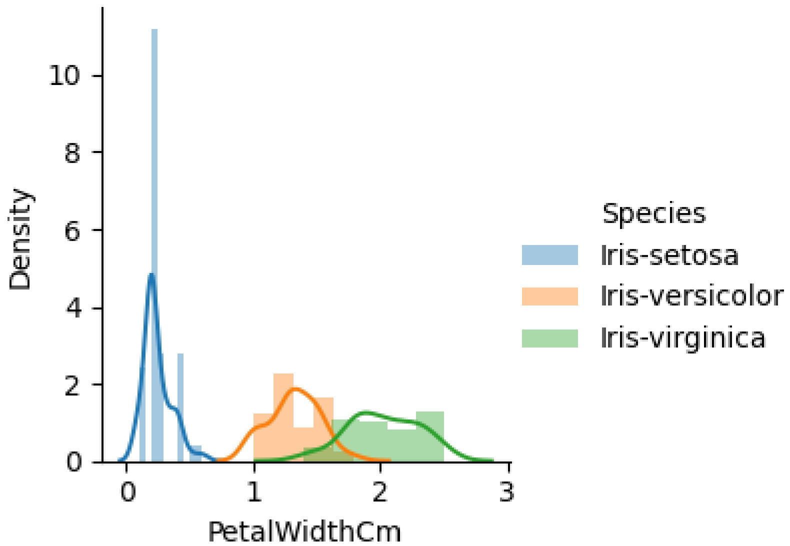
```
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
  func(*plot_args, **plot_kwargs)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
```









From the above plots, we can see that

In the case of Sepal Length, there is a huge amount of overlapping.

In the case of Sepal Width also, there is a huge amount of overlapping.

In the case of Petal Length, there is a very little amount of overlapping.

In the case of Petal Width also, there is a very little amount of overlapping

So we can use Petal Length and Petal Width as the classification feature.

In [ ]:

Handling Correlation

Pandas dataframe.corr() is used to find the pairwise correlation of all columns in the dataframe. Any NA values are automatically excluded. For any non-numeric data type columns in the dataframe it is ignored.

In [ ]:

Example:

In [54]:
```
data.select_dtypes(include=['number']).corr(method='pearson')
```

Out[54]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthC
Id	1.000000	0.624413	-0.654654	0.969909	0.9996
SepalLengthCm	0.624413	1.000000	-0.999226	0.795795	0.6438
SepalWidthCm	-0.654654	-0.999226	1.000000	-0.818999	-0.6734
PetalLengthCm	0.969909	0.795795	-0.818999	1.000000	0.9757
PetalWidthCm	0.999685	0.643817	-0.673417	0.975713	1.0000

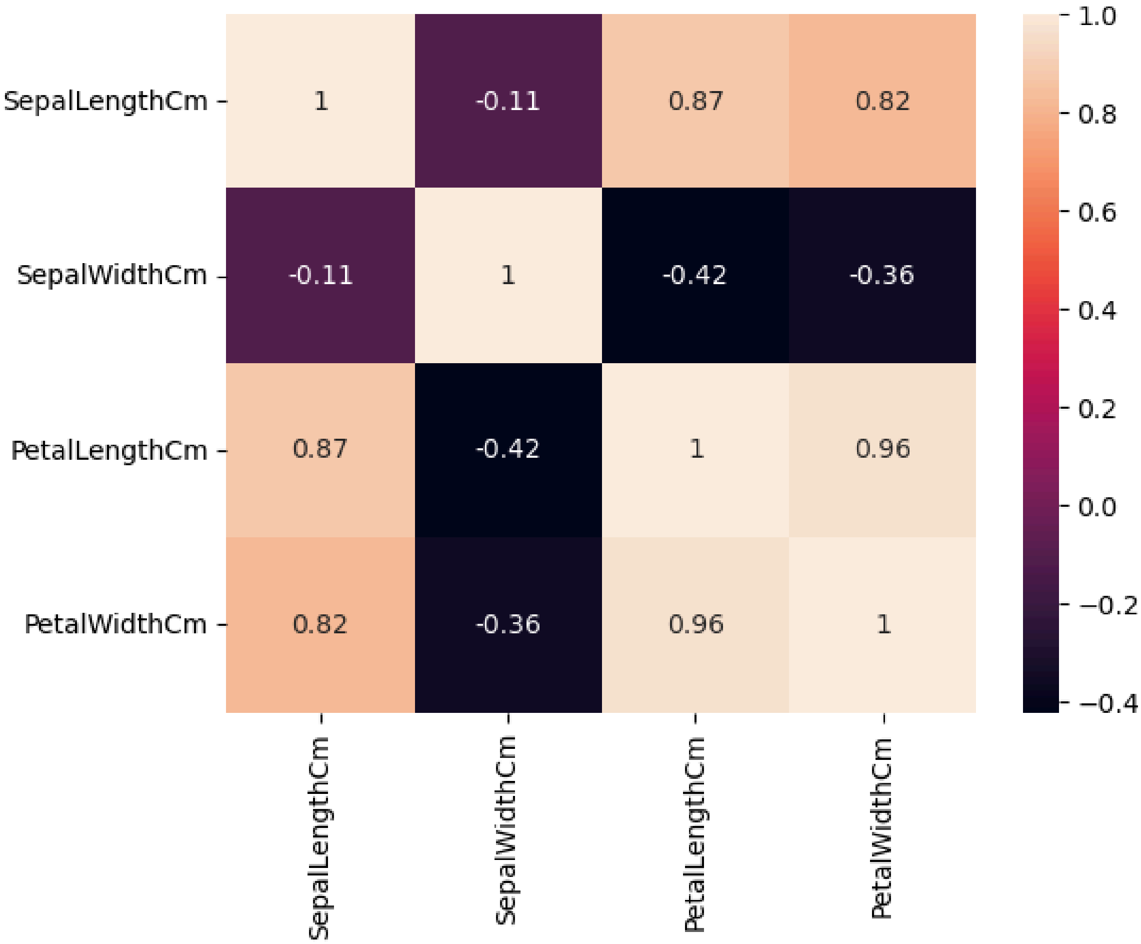
In [ ]:

Heatmaps

The heatmap is a data visualization technique that is used to analyze the dataset as colors in two dimensions.

Basically, it shows a correlation between all numerical variables in the dataset. In simpler terms, we can plot the above-found correlation using the heatmaps.

In [56]:
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.heatmap(dataset.select_dtypes(include=['number']).corr(method='pearson').drc
['Id'], axis=1).drop(['Id'], axis=0),
                        annot = True);

plt.show()

# This code is modified by Susobhan Akhuli
```



From the above graph, we can see that -

Petal width and petal length have high correlations.

Petal length and sepal width have good correlations.

Petal Width and Sepal length have good correlations.

In [ ]:

```

```

Box Plots

We can use boxplots to see how the categorical value os distributed with other numerical values.

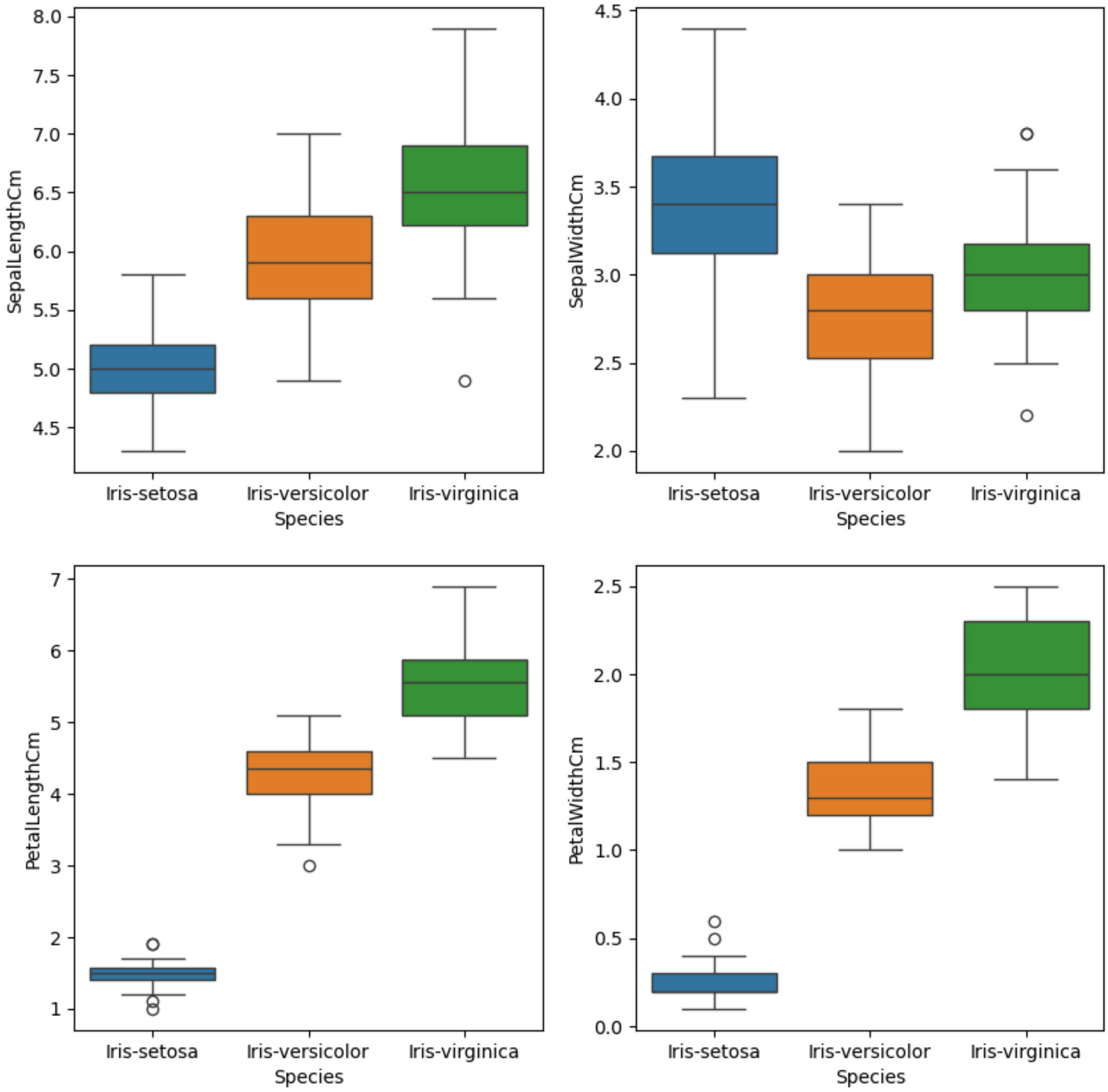
In [ ]:

```

```

Example:

In [59]:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

def graph(y):
    sns.boxplot(x="Species", y=y, data=dataset,hue="Species")

plt.figure(figsize=(10,10))

# Adding the subplot at the specified
# grid position
plt.subplot(221)
graph('SepalLengthCm')

plt.subplot(222)
graph('SepalWidthCm')

plt.subplot(223)
graph('PetalLengthCm')

plt.subplot(224)
graph('PetalWidthCm')

plt.show()
```



From the above graph, we can see that -

Species Setosa has the smallest features and less distributed with some outliers.

Species Versicolor has the average features.

Species Virginica has the highest features

```
In [ ]:
```

Handling Outliers

An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal)objects. They can be caused by measurement or execution errors. The analysis for outlier detection is referred to as outlier mining. There are many ways to detect the outliers, and the removal process is the data frame same as removing a data item from the panda's dataframe.

In [ ]:

Let's consider the iris dataset and let's plot the boxplot for the SepalWidthCm column.

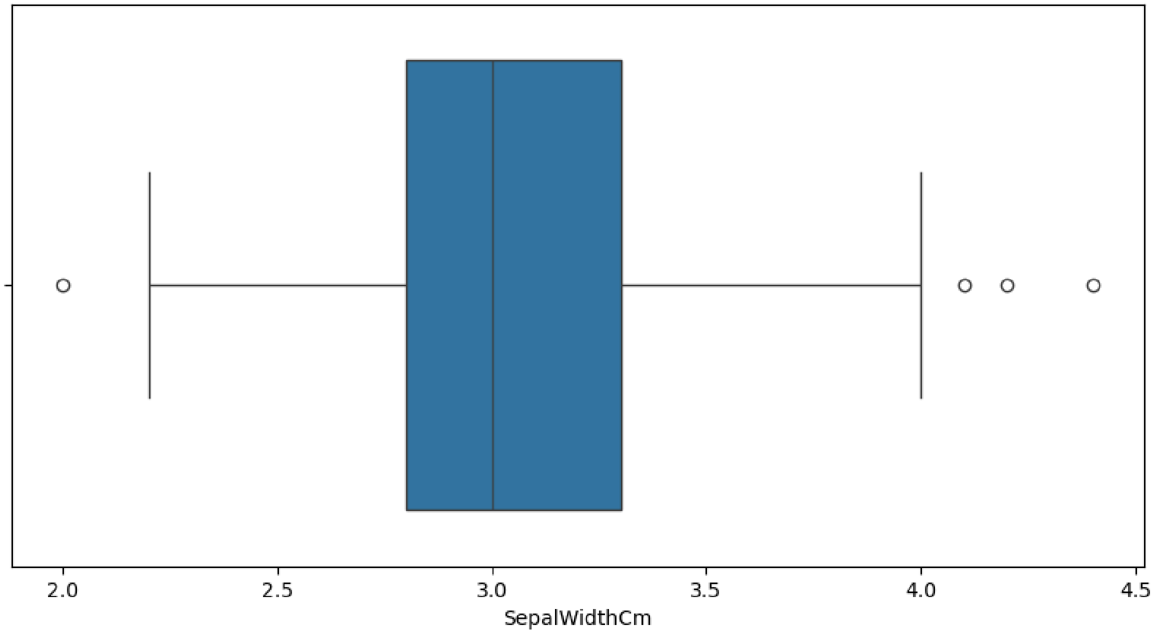
Example:

In [64]:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
plt.figure(figsize=(10,5))
df = pd.read_csv(r"C:\Users\avani\Downloads\Iris.csv")


sns.boxplot(x='SepalWidthCm', data=df)
```

Out[64]:

```
<Axes: xlabel='SepalWidthCm'>
```



In the above graph, the values above 4 and below 2 are acting as outliers.

In [ ]:

Removing Outliers

For removing the outlier, one must follow the same process of removing an entry from the dataset using its exact position in the dataset because in all the above methods of detecting the outliers end result is the list of all those data items that satisfy the outlier definition according to the method used.

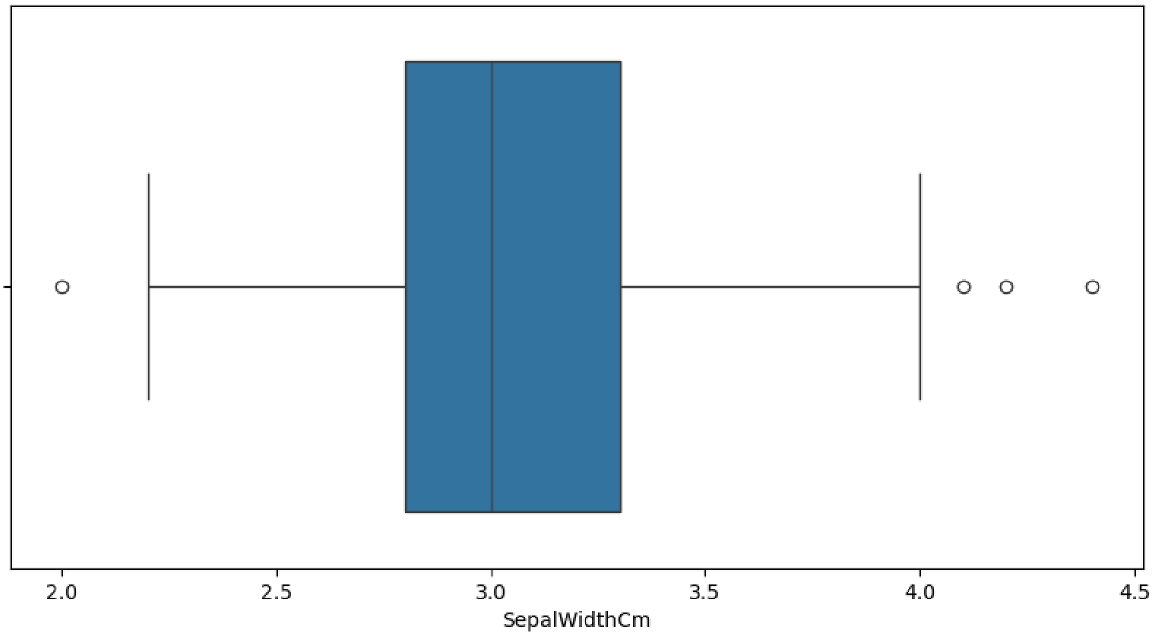
In [ ]:

Example: We will detect the outliers using IQR and then we will remove them. We will also draw the boxplot to see if the outliers are removed or not.

In [67]:
```python
# Importing
import numpy as np

# Load the dataset
plt.figure(figsize=(10,5))
df = pd.read_csv(r"C:\Users\avani\Downloads\Iris.csv")

# IQR
Q1 = np.percentile(dataset['SepalWidthCm'], 25,
                                interpolation = 'midpoint')

Q3 = np.percentile(dataset['SepalWidthCm'], 75,
                                interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", dataset.shape)

# Upper bound
upper = np.where(dataset['SepalWidthCm'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(dataset['SepalWidthCm'] <= (Q1-1.5*IQR))

# Removing the Outliers
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)

print("New Shape: ", df.shape)

sns.boxplot(x='SepalWidthCm', data=dataset)

# This code is modified by Susobhan Akhuli
```

```
Old Shape:  (150, 6)
New Shape:  (146, 6)
```

Out[67]: <Axes: xlabel='SepalWidthCm'>



Note: for more information, refer Detect and Remove the Outliers using Python.

In [ ]:

```

```