

Kotha Karthik Goud 06/25/2023

Assignment 3 Hands-on

EDA: Python Data Visualization with Matplotlib, Pandas, and NumPy

Import Libraries and Load Dataset

In [25]: *# Importing the libraries which are needed to run the code.*

```
import pandas as pd
import numpy as np

# filter warnings
import warnings
warnings.filterwarnings("ignore")

import matplotlib.pyplot as plt

from pandas.plotting import scatter_matrix
from pandas import DataFrame, read_csv
```

In [26]: *# Import os module which is a miscellaneous operating system interface*

```
import os
```

```
#Use a command in the os module to show your current working directory for ref
cwd = os.getcwd()
cwd
```

Out[26]: 'C:\\Users\\sudhe\\Downloads'

```
In [27]: # Load the data set 'iris.csv' into a pandas dataframe
# Read the iris data set and create the dataframe df

df = ('iris.csv')
df = pd.read_csv ('iris.csv')
df.head(5)
```

Out[27]:

	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

```
In [28]: #print the information about the dataset

print(df.info())
```

```
<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
None
```

Univariate Data Visualization

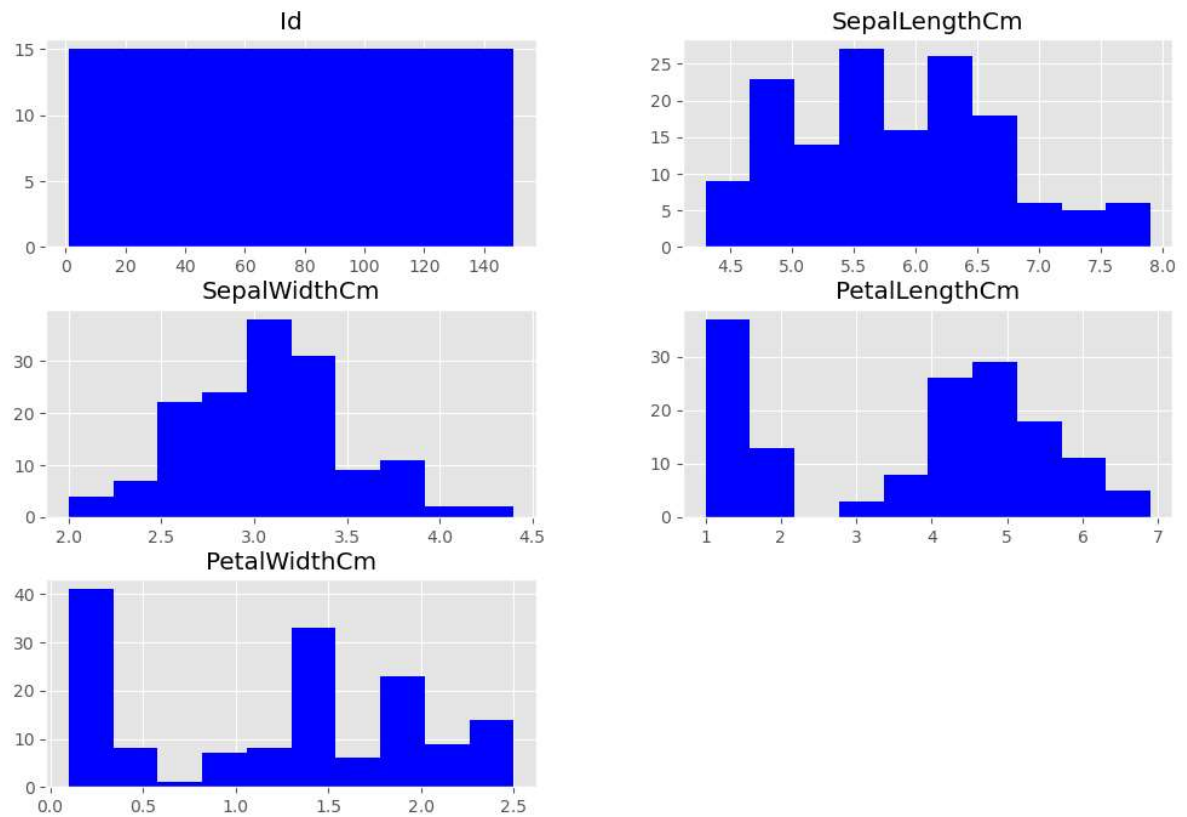
Histograms

- Histograms are great when we would like to show the distribution of the data we are working with.

```
In [29]: # Create a histogram
# SepalWidth - normal dist, PetalLength - bimodal
# The normal distribution is so important easier for mathematical statistician
# Many kinds of statistical tests can be derived from normal distributions.

df.hist(figsize=(12,8), color='blue')
plt.show
```

```
Out[29]: <function matplotlib.pyplot.show(close=None, block=None)>
```



```
In [30]: # Here we want to see the different Species

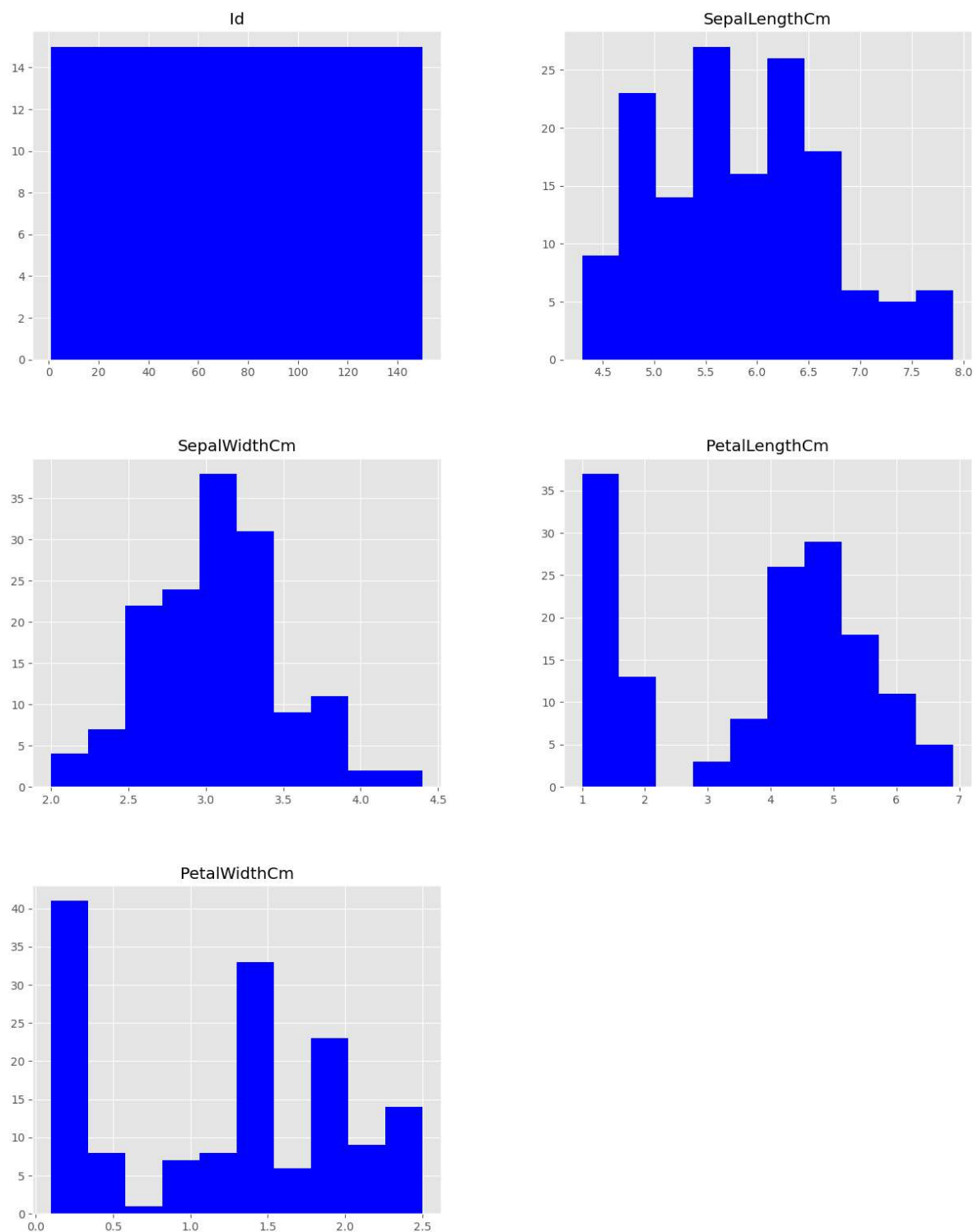
print(df.groupby('Species').size())
```

```
Species
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64
```

In [31]: *# Histograms are great when we would like to show the distribution of the data*

```
df.hist(figsize=(15,19), color='blue')  
plt.show
```

Out[31]: <function matplotlib.pyplot.show(close=None, block=None)>

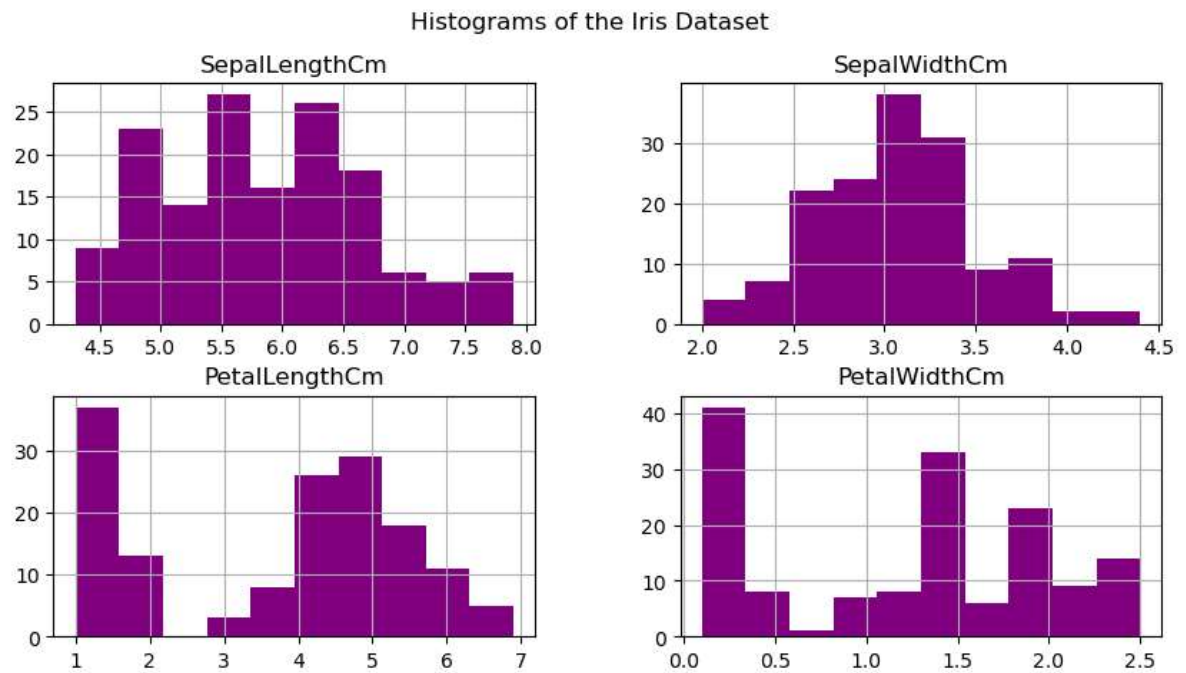


```
In [32]: # In the above code, we see the variable "Id" is included in the analysis.  
# In order to get rid of this variable we use the following code.  
# It doesn't provide any output.
```

```
df.__delitem__('Id')
```

```
In [12]: df.hist(figsize=(10,5), color="purple")  
plt.suptitle("Histograms of the Iris Dataset")  
plt.show  
  
# After this run, we see that "Id" is gone. you can also see we changed the co
```

```
Out[12]: <function matplotlib.pyplot.show(close=None, block=None)>
```

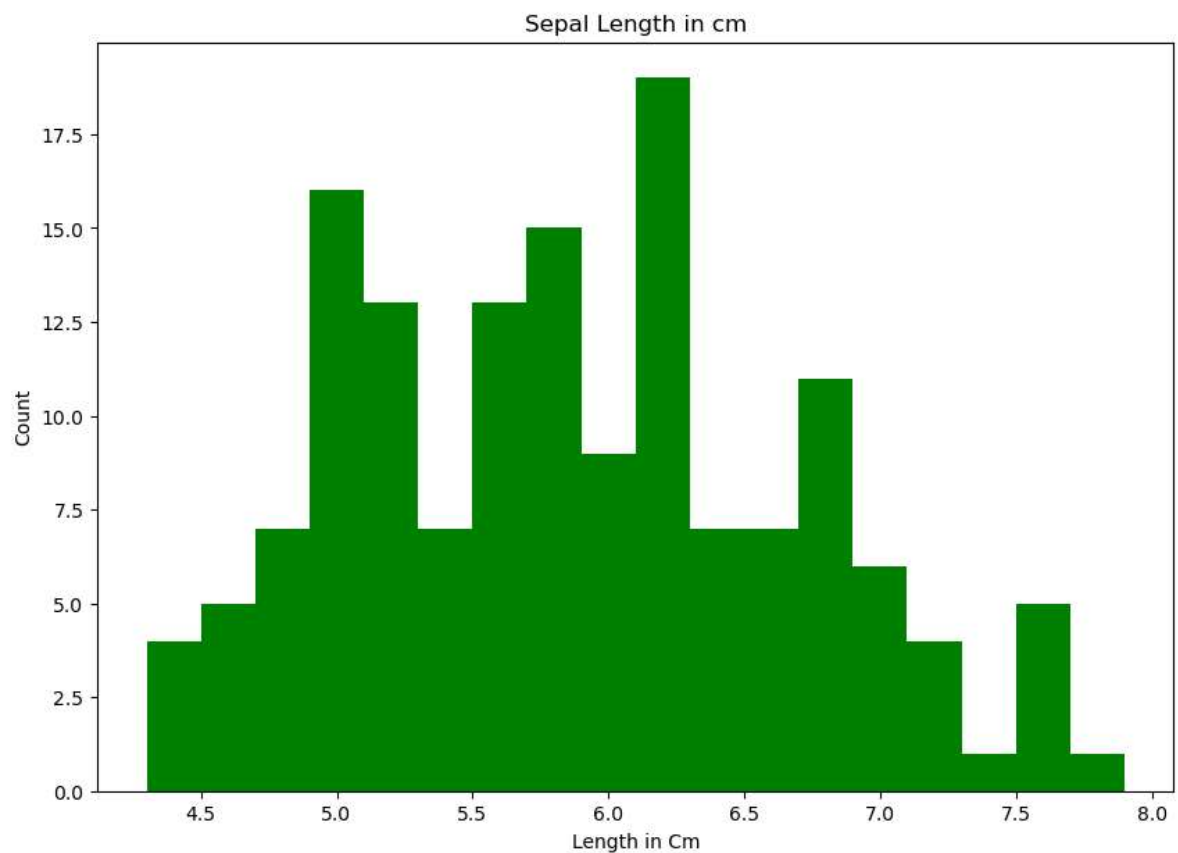


Let's say we want to make changes to the histograms.

We will look at only the PetalLengthCm variable.

```
In [13]: # A histogram with just one variable - Sepal Length.  
# We need to isolate the one variable using square brackets [].  
  
plt.figure(figsize = (10, 7))  
x = df["SepalLengthCm"]  
  
# bins is an integer, it defines the number of equal-width bins in the range  
plt.hist(x, bins = 18, color = "green")  
plt.title("Sepal Length in cm")  
plt.xlabel("Length in Cm")  
plt.ylabel("Count")
```

Out[13]: Text(0, 0.5, 'Count')

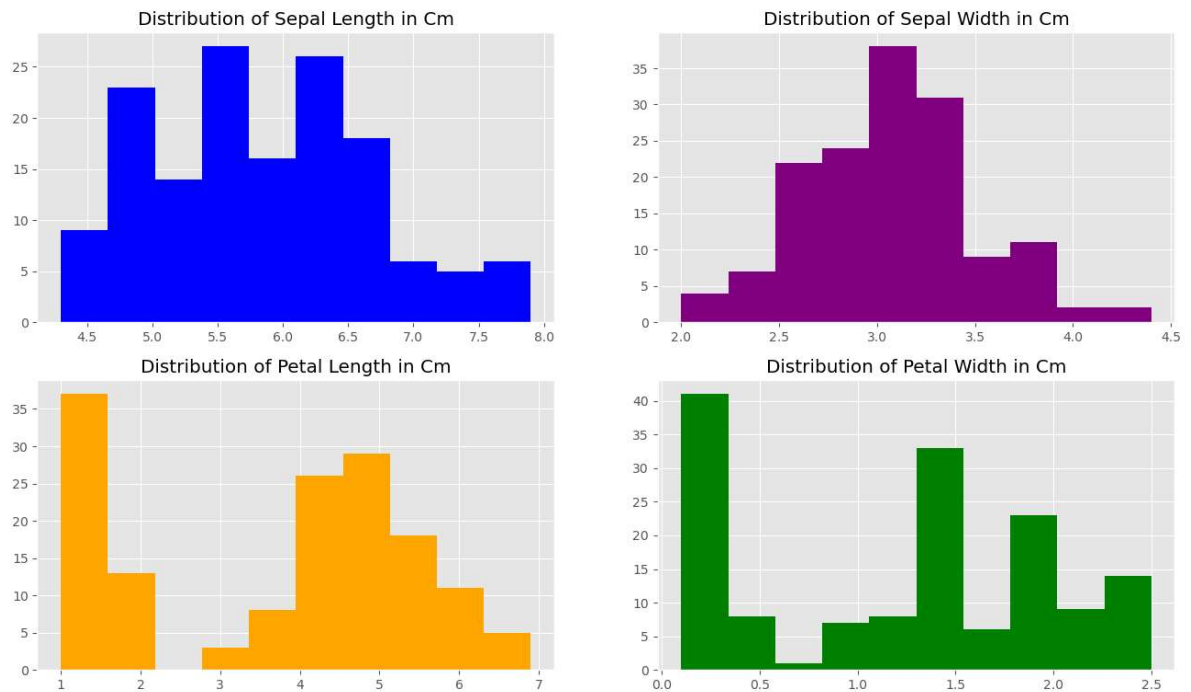


In [14]: *# Here we are isolating the variable and giving each one a color and a specific grammar of graphics plot*

```
plt.style.use("ggplot")

fig, axes = plt.subplots(2, 2, figsize=(16,9))

axes[0,0].set_title("Distribution of Sepal Length in Cm")
axes[0,0].hist(df['SepalLengthCm'], bins=10, color='blue');
axes[0,1].set_title("Distribution of Sepal Width in Cm")
axes[0,1].hist(df['SepalWidthCm'], bins=10, color='purple');
axes[1,0].set_title("Distribution of Petal Length in Cm")
axes[1,0].hist(df['PetalLengthCm'], bins=10, color='orange');
axes[1,1].set_title("Distribution of Petal Width in Cm")
axes[1,1].hist(df['PetalWidthCm'], bins=10, color='green');
```

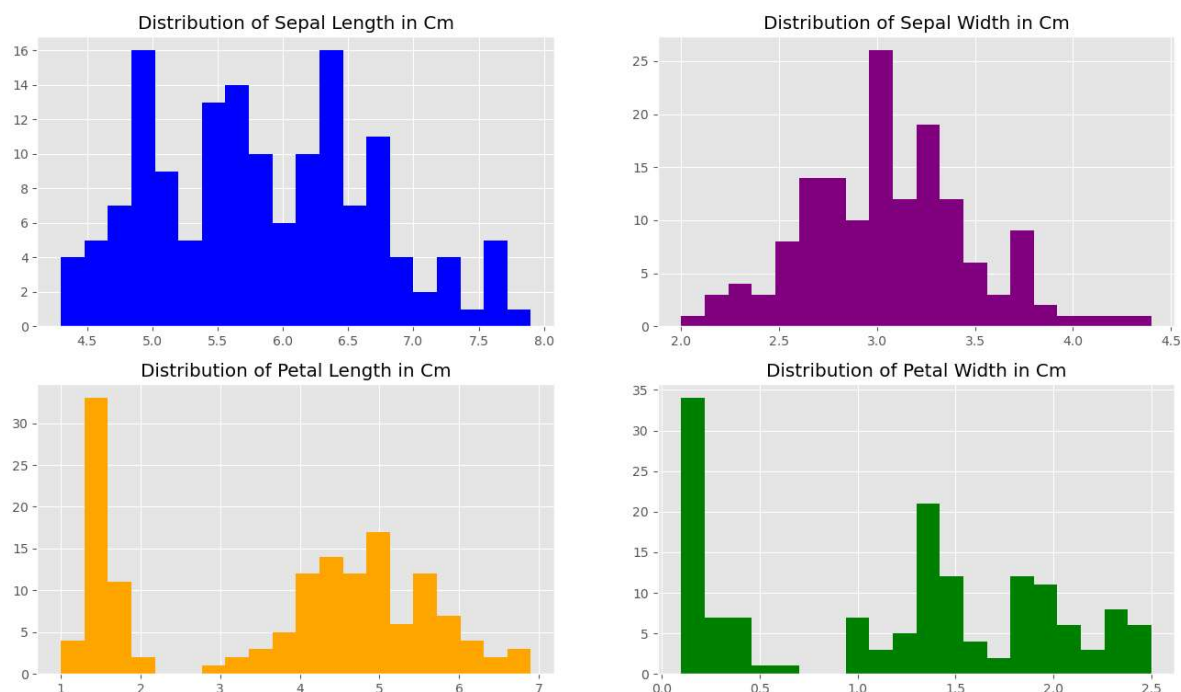


In [15]: *# Lastly, let's change the bin size*

```
plt.style.use("ggplot")

fig, axes = plt.subplots(2, 2, figsize=(16,9))

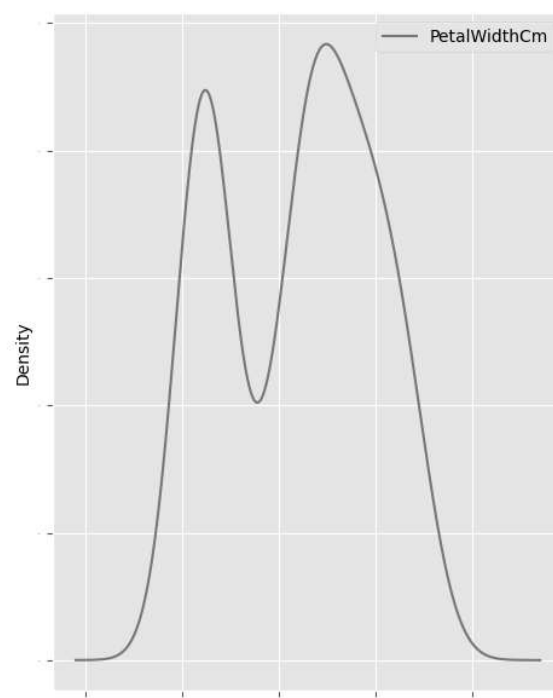
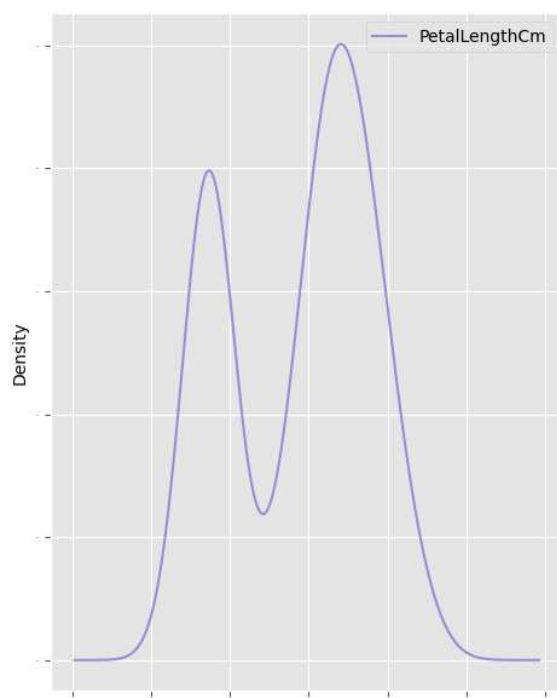
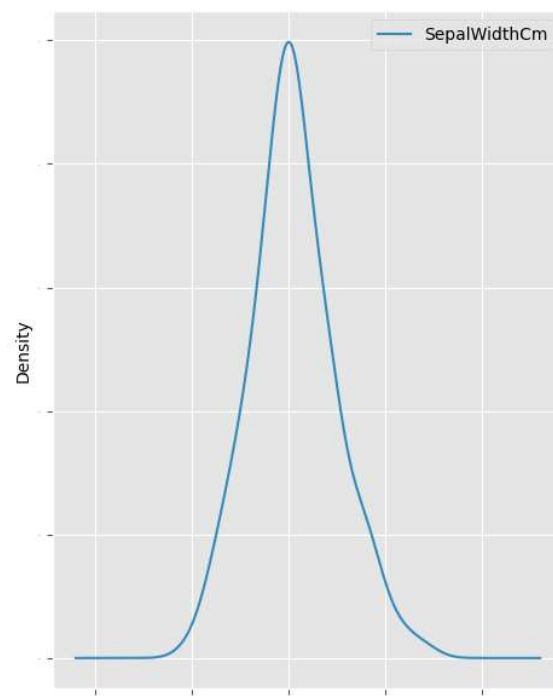
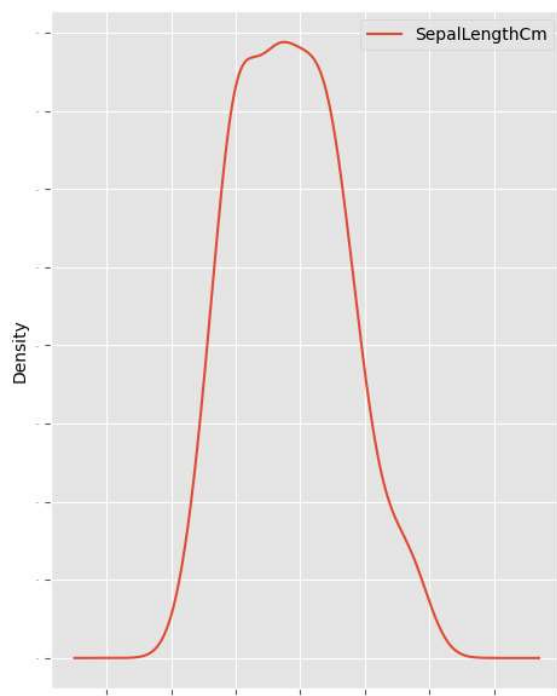
axes[0,0].set_title("Distribution of Sepal Length in Cm")
axes[0,0].hist(df['SepalLengthCm'], bins=20, color='blue');
axes[0,1].set_title("Distribution of Sepal Width in Cm")
axes[0,1].hist(df['SepalWidthCm'], bins=20, color='purple');
axes[1,0].set_title("Distribution of Petal Length in Cm")
axes[1,0].hist(df['PetalLengthCm'], bins=20, color='orange');
axes[1,1].set_title("Distribution of Petal Width in Cm")
axes[1,1].hist(df['PetalWidthCm'], bins=20, color='green');
```



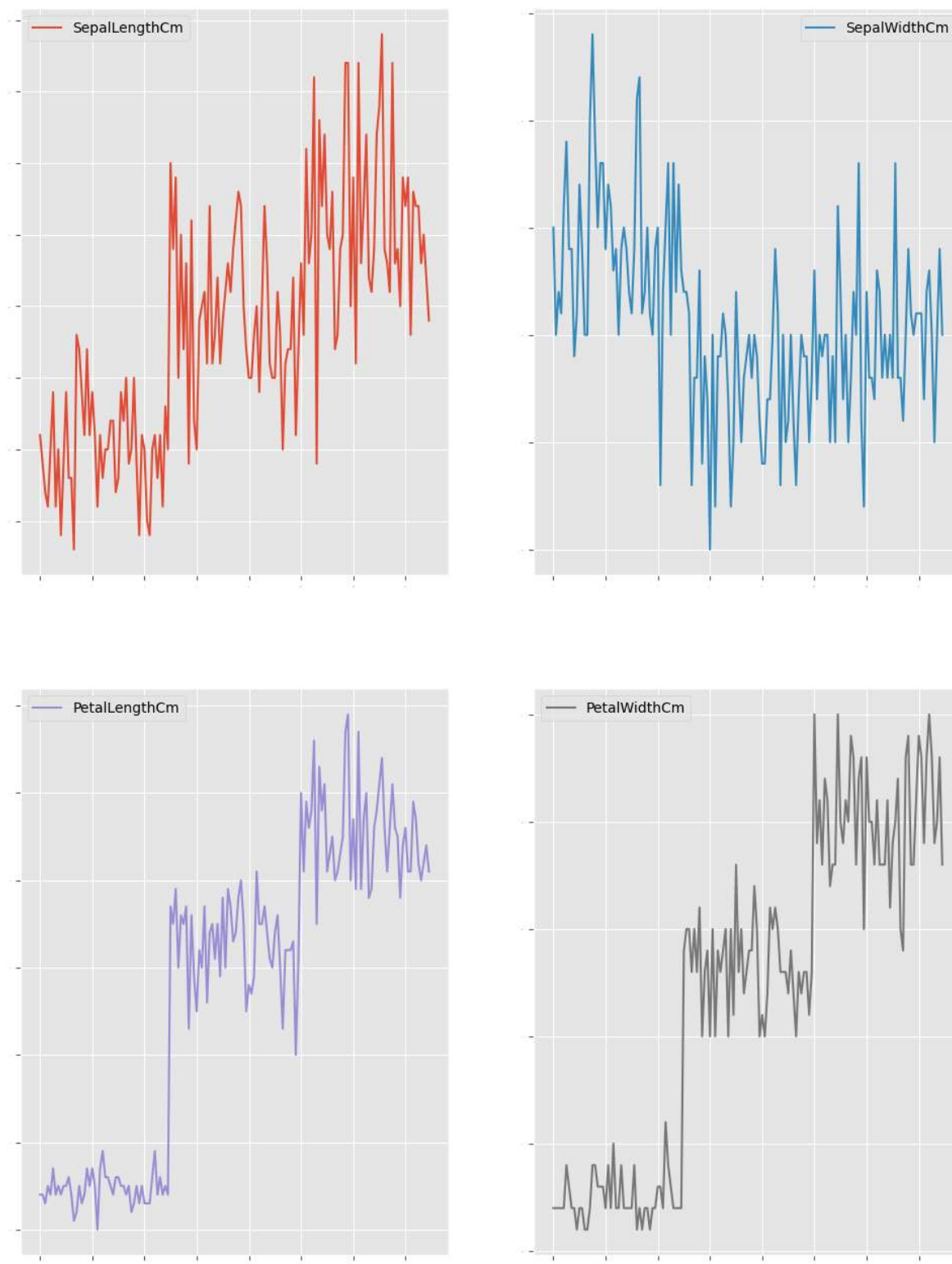
Density Plots

- A Density Plot visualizes the distribution of data over a time period or a continuous interval. This chart is a variation of a Histogram but it smooths out the noise made by binning.
- Density Plots have a slight advantage over Histograms since they're better at determining the distribution shape and, as mentioned above, they are not affected by the number of bins used (each bar used in a typical histogram). As we saw above a Histogram with only 10 bins wouldn't produce a distinguishable enough shape of distribution as a 20-bin Histogram would. With Density Plots, this isn't an issue.


```
In [16]: # Create the density plot  
# If subplots=True is specified, plots for each column are drawn as subplots  
  
df.plot(kind='density', subplots=True, layout=(2,2), sharex=False, legend=True,  
plt.show())
```



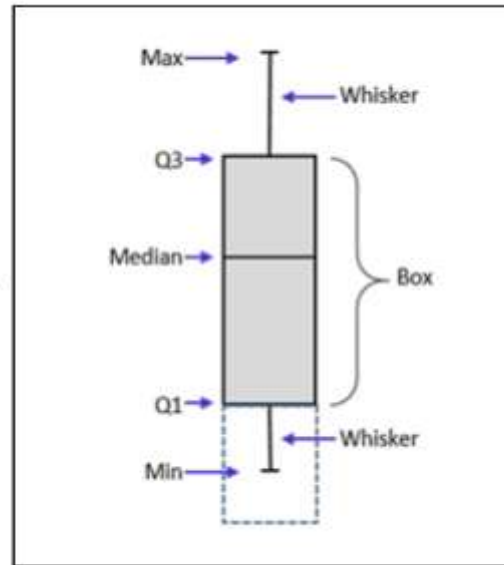
```
In [17]: # Create a Line graph
df.plot(kind='line', subplots=True, layout=(2,2), sharex=False, legend=True, f
plt.show())
```



Boxplots

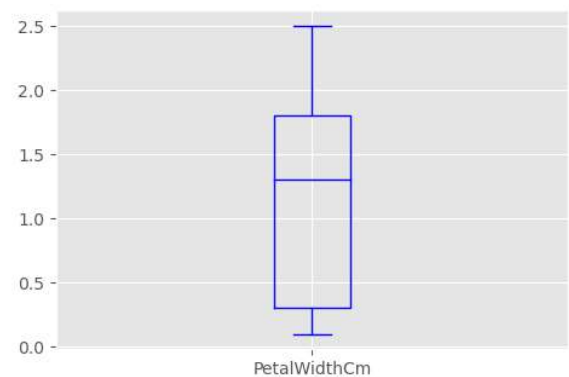
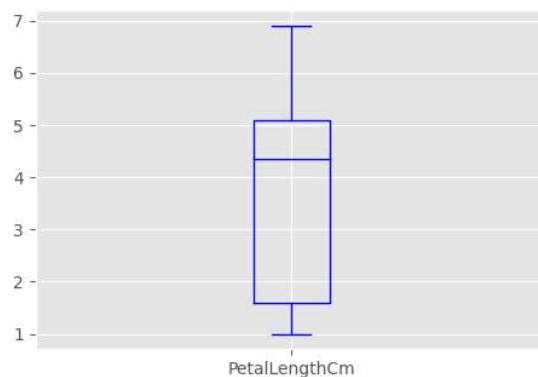
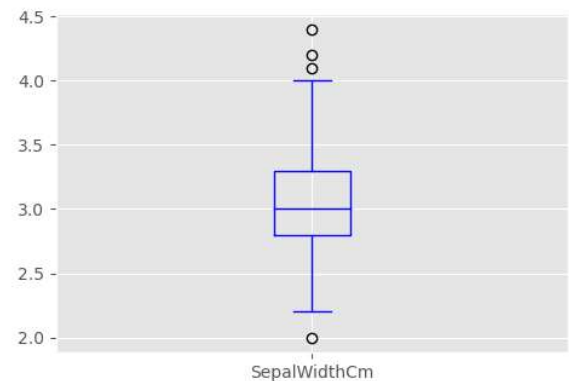
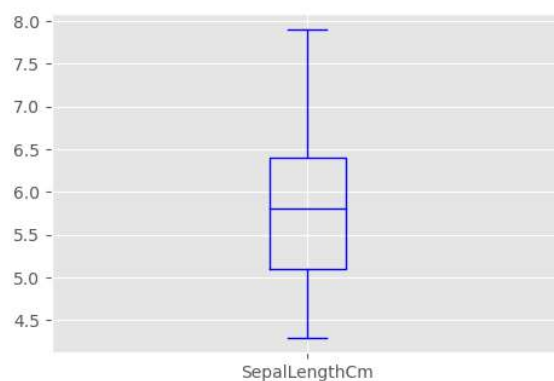
- A box plot is a very good plot to understand the spread, median, and outliers of data

1. Q3: This is the 75th percentile value of the data. It's also called the upper hinge.
2. Q1: This is the 25th percentile value of the data. It's also called the lower hinge.
3. Box: This is also called a step. It's the difference between the upper hinge and the lower hinge.
4. Median: This is the midpoint of the data.
5. Max: This is the upper inner fence. It is 1.5 times the step above Q3.
6. Min: This is the lower inner fence. It is 1.5 times the step below Q1.



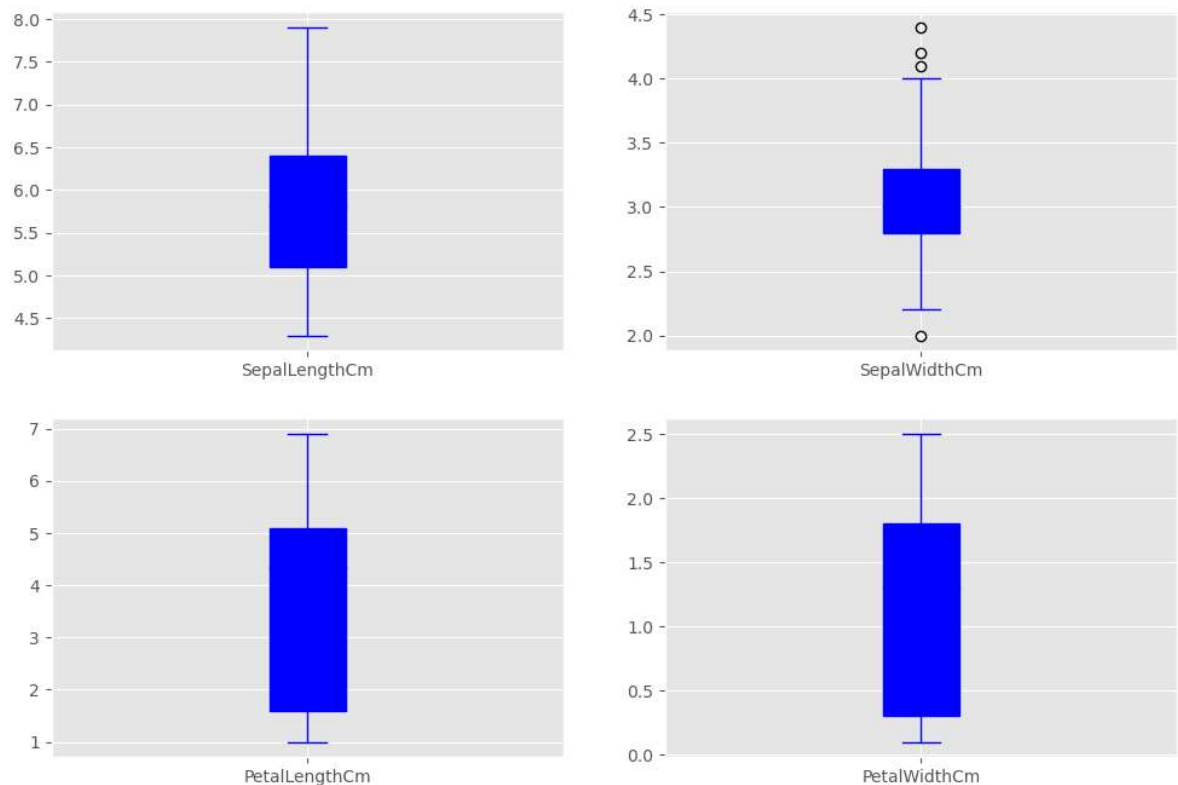
In [18]: *# Create a box plot*

```
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False, c
plt.show())
```



```
In [19]: # Fill the boxes with color, using patch_artist
```

```
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False, color='blue',  
plt.show())
```

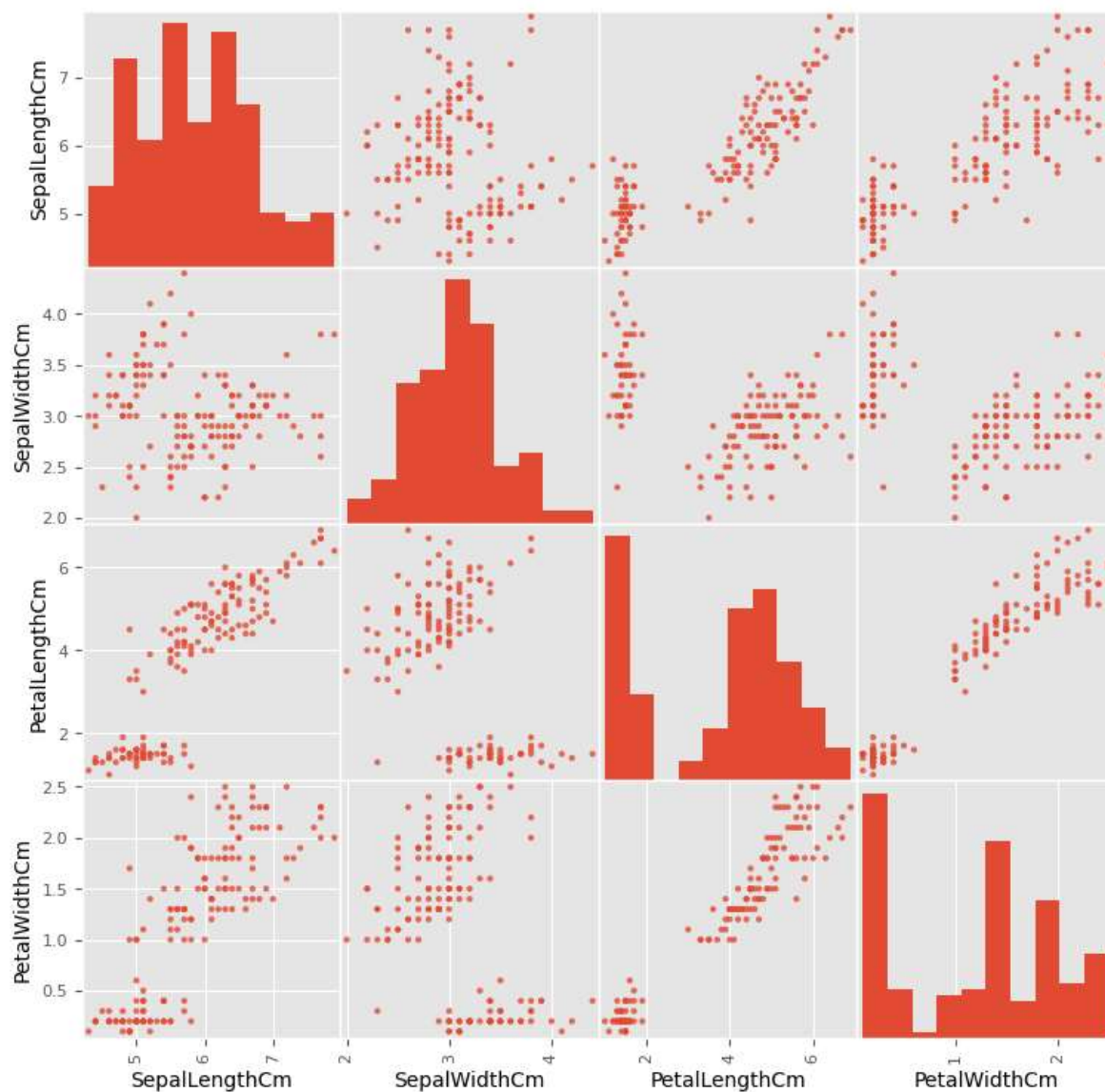


Multivariate Data Visualization

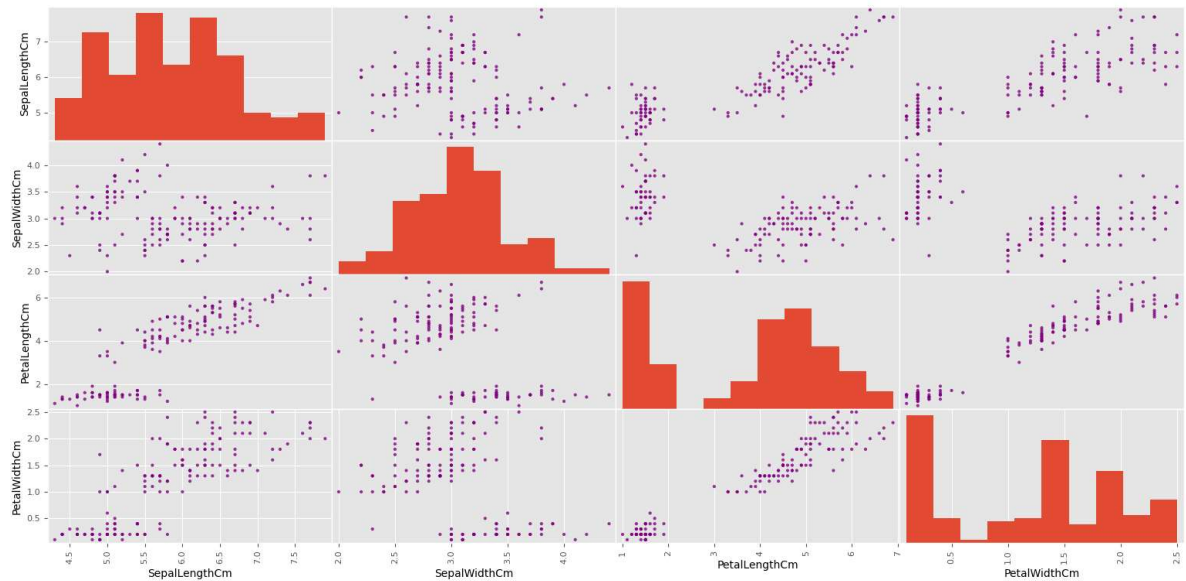
Scatter Matrix Plot

- A scatter plot matrix is a grid (or matrix) of scatter plots. This type of graph is used to visualize bivariate relationships between different combinations of variables. Each scatter plot in the matrix visualizes the relationship between a pair of variables, allowing many relationships to be explored in one chart. For example, the first row shows the relationship between SepalLength and the other 3 variables.

```
In [20]: # Create a scatter matrix plot  
# alpha = Amount of transparency applied  
  
scatter_matrix(df, alpha=0.8, figsize=(9,9))  
plt.show()
```

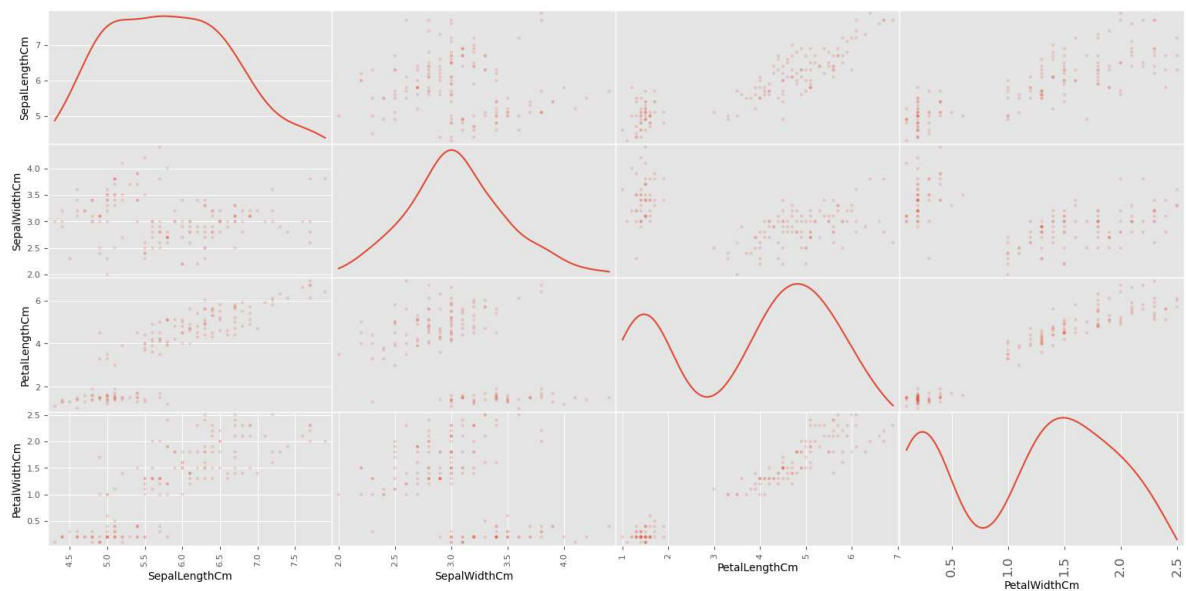


```
In [21]: # change the color to purple. Notice only one part of the plot changes.
scatter_matrix(df, alpha=0.8, figsize=(19,9), color = 'purple')
plt.show()
```



```
In [22]: # Here you are adding a subtitle.
# Pick between 'kde' and 'hist' for either Kernel Density Estimation or Histogram
# KDE = a density estimator is an algorithm which seeks to model the probability
scatter_matrix(df, alpha=0.2, diagonal = 'kde', figsize=(19,9))
plt.suptitle('Scatter-matrix for each input variable')
plt.tick_params(labelsize=12, pad=6)
```

Scatter-matrix for each input variable

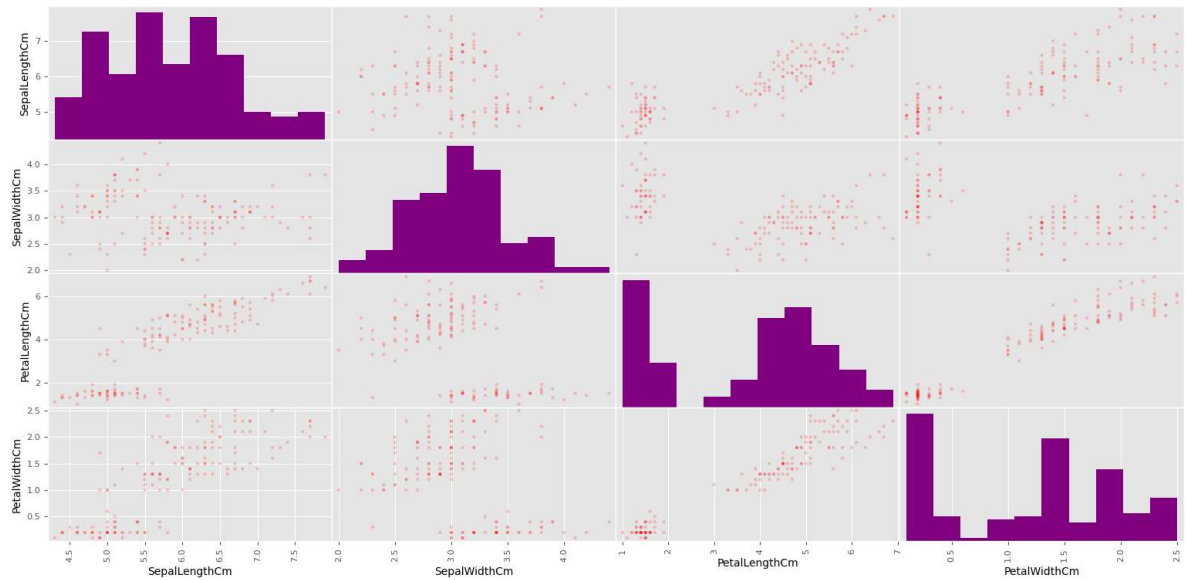


In [23]: *# Lastly, you are changing the color to both parts of the plot.*

```
scatter_matrix(df, figsize= (19,9), alpha=0.2,  
c='red', hist_kwds={'color':['purple']})  
plt.suptitle('Scatter-matrix for each input variable', fontsize=28)
```

Out[23]: Text(0.5, 0.98, 'Scatter-matrix for each input variable')

Scatter-matrix for each input variable



You are done. Great job!