Discovery and Learning with Big Data/Machine Learning

DeAundrie Howard

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

1. What is happening in the code block below? Enter your answer by adding a new code block and use markdown.

Various python libraries and tools are being imported. Each are given alias to help distinguish from other assigned.

In [1]:

```
# Import all needed libraries

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

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

2. What is happening in the code block below? Enter your answer by adding a new code block and use markdown.

The Iris dataset is being imported into the file, read, and assigned to a pandas dataframe.

In [2]:

```
# Load the data set into a pandas dataframe
# Read the Iris data set and create the dataframe df
df = ('iris.csv')
df = pd.read_csv ('iris.csv')
```

In [3]:

```
df.head(5)
```

Out[3]:

	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 [47]:

```
df.describe()
```

Out[47]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
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

In [4]:

```
df.tail(5)
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

3. What would happen if you entered df.tail(5)? Enter your answer by adding a new code block and insert the code.

"df.tail(5)" prints out the last five (5) records within a dataframe.

4. What type of information do you get with the code below? Enter your answer by adding a new code block and use markdown.

In [5]:

```
#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
```

In [48]:

```
df.isnull()
```

Out[48]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [52]:

```
df.value_counts()
```

Out[52]:

```
SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species        
4.9            3.1           1.5            0.1           Iris-setosa        3
5.8            2.7           5.1            1.9           Iris-virginica     2
               4.0           1.2            0.2           Iris-setosa        1
5.9            3.0           4.2            1.5           Iris-versicolor    1
6.2            3.4           5.4            2.3           Iris-virginica     1
                                                                            ..
5.5            2.3           4.0            1.3           Iris-versicolor    1
               2.4           3.7            1.0           Iris-versicolor    1
                             3.8            1.1           Iris-versicolor    1
               2.5           4.0            1.3           Iris-versicolor    1
7.9            3.8           6.4            2.0           Iris-virginica     1
Name: count, Length: 147, dtype: int64
```

"df.info()" prints out crucial information about a dataframe, which may include the following: data type, columns name, column indices, total of entries within a column, and how much memory is being used. "df.isnull()" prints out any records contain null values. "df.value_counts()" prints out a count of unique values.

Univariate Data Visualization

Histograms

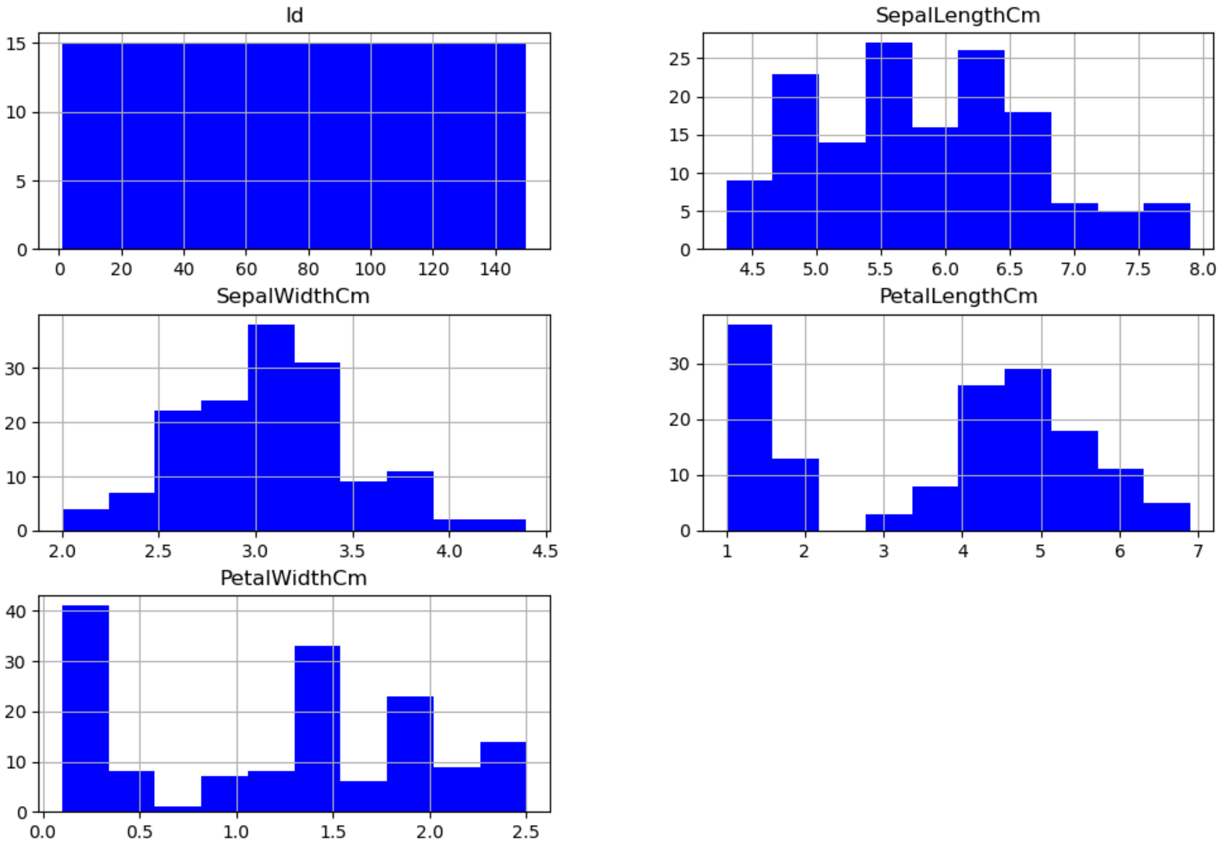
- Histograms are great when we would like to show the distribution of the data we are working with.

In [6]:

```
#create a histogram

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

Out[6]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



5. Copy the code above and change the color to 'red'. Enter your code by adding a new code block and use code instead of markdown.

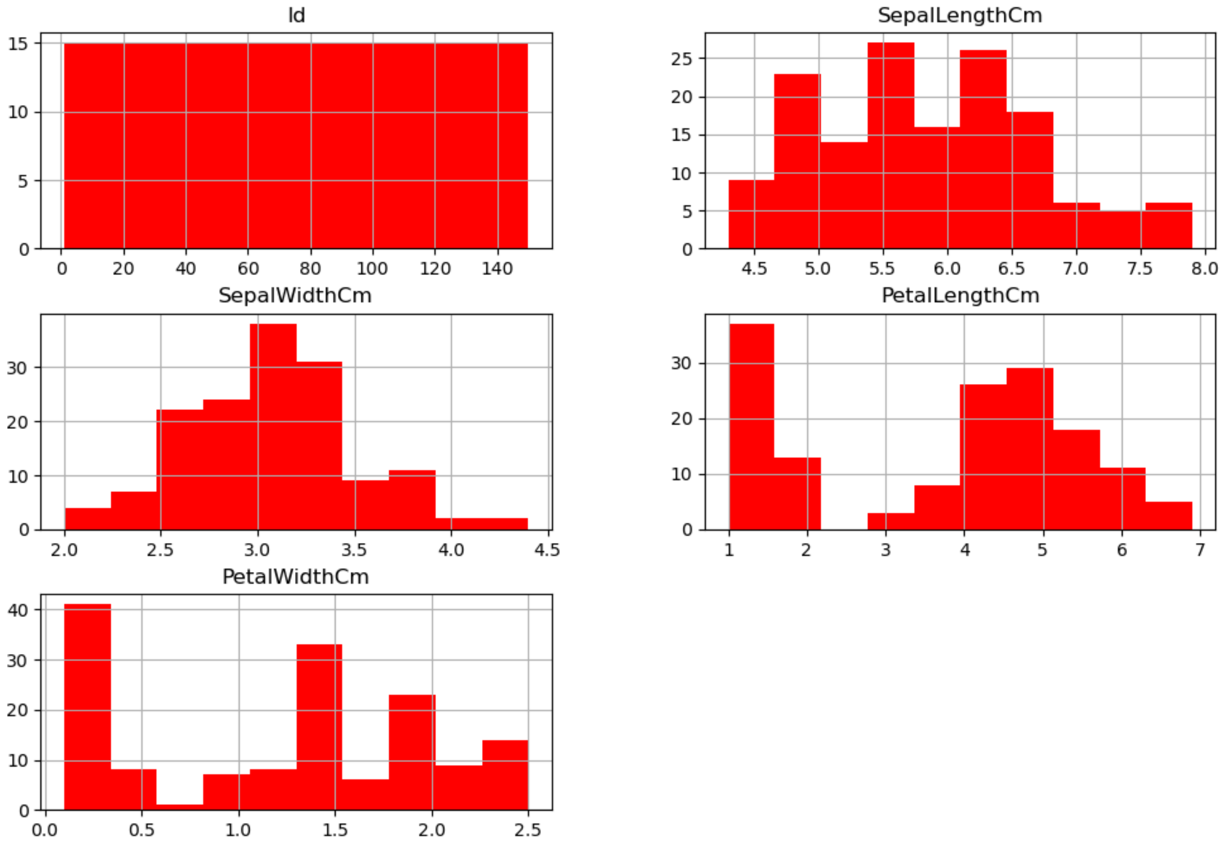
In [7]:

```
#create a histogram
```

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

Out[7]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [8]:

```
# 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
```

6. What is the count for each of the different species? Enter your answer by adding a new code block and use markdown.

Within the differents iris species, Setosa, Versicolor, and Virginica, each has a count of 50.

In [9]:

```
# in the above code, we see the variable "Id" is included in the analysis.  In order t

df.__delitem__('Id')
```

7. Why do you think we would not want the ID column? Enter your answer by adding a new code block and use markdown.

Deleting the ID column removes the unnecessary information and unnecessary graphs.

8. What code would you use to check to see if the column ID was deleted? Enter your answer by adding a new code block and add the code in the block. HINT: look at code block 3

In [10]:

```
df.head(5)
```

Out[10]:

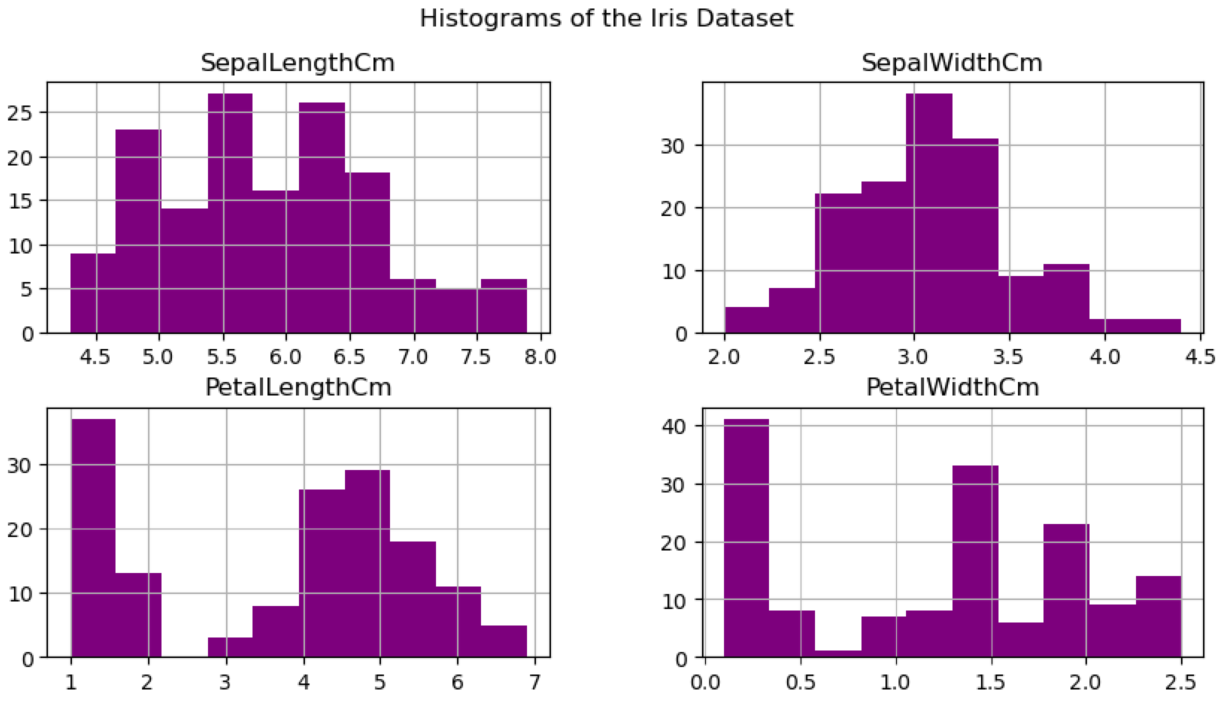
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [11]:

```
# Here we will change the color of the histogram to purple
df.hist(figsize=(10,5), color ="purple")
plt.suptitle("Histograms of the Iris Dataset")
plt.show
```

Out[11]:

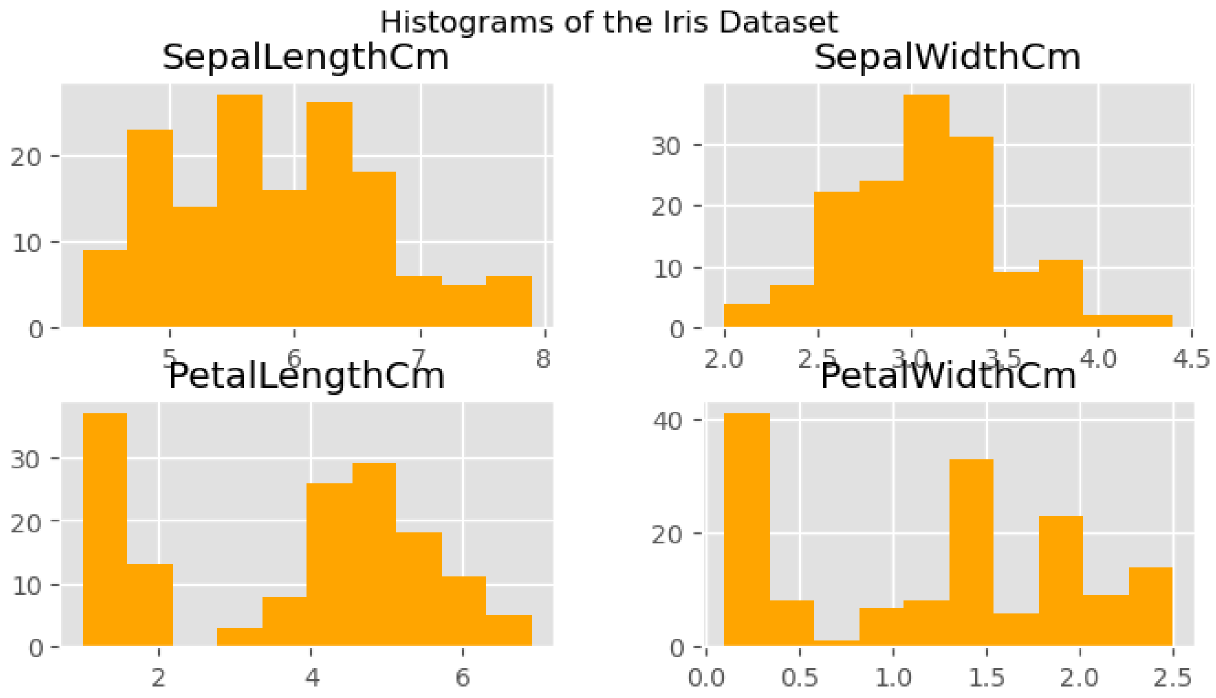
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [46]:

```
# Here we will change the color of the histogram to purple
df.hist(figsize=(8,4), color ="orange")
plt.suptitle("Histograms of the Iris Dataset")
plt.show
```

```
Out[46]: <function matplotlib.pyplot.show(close=None, block=None)>
```

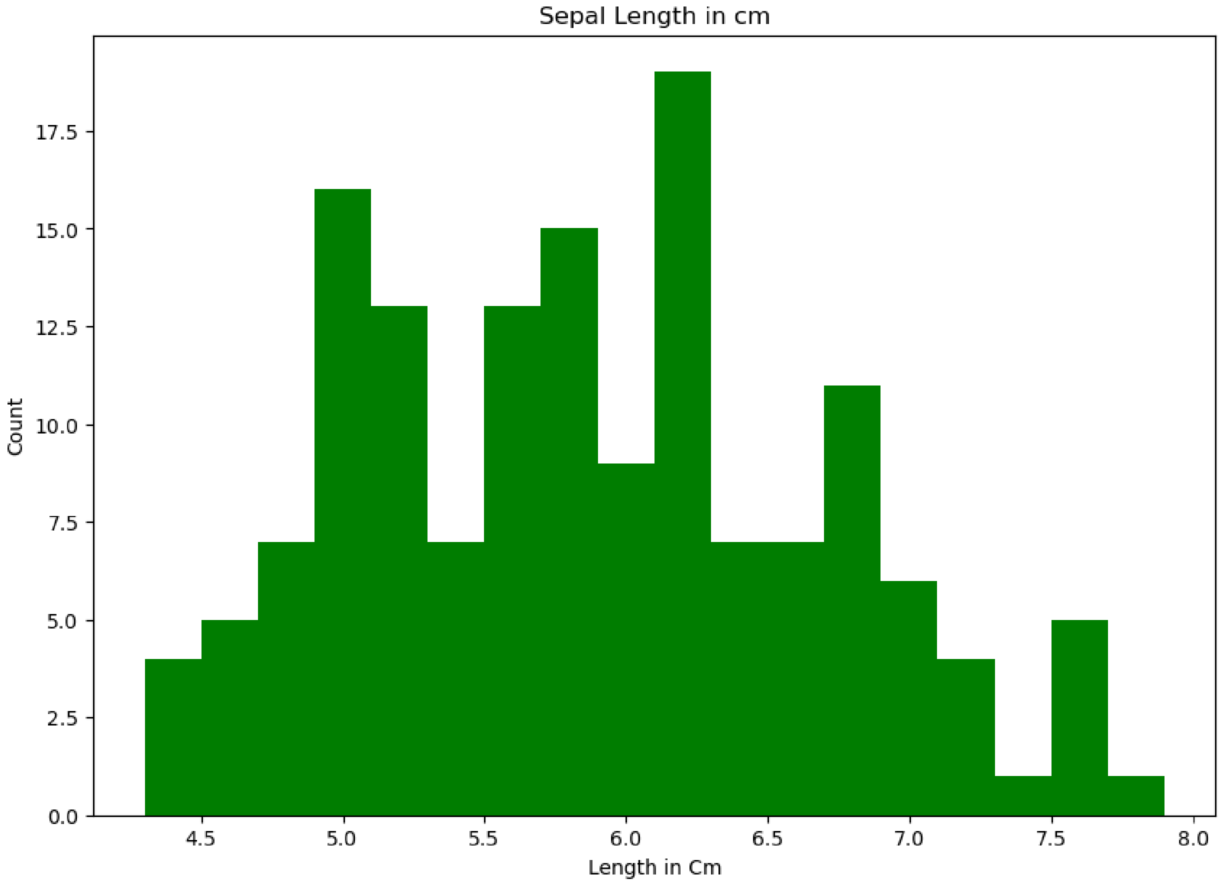


Let's say we want to make changes to the histograms.

We will look at only the PetalLengthCm variable.

```
In [12]: # histogram with just one variable - Sepal Length.  We need to isolate the one variabl

plt.figure(figsize = (10, 7))
x = df["SepalLengthCm"]
plt.hist(x, bins = 18, color = "green")
plt.title("Sepal Length in cm")
plt.xlabel("Length in Cm")
plt.ylabel("Count")
```

```
Out[12]: Text(0, 0.5, 'Count')
```

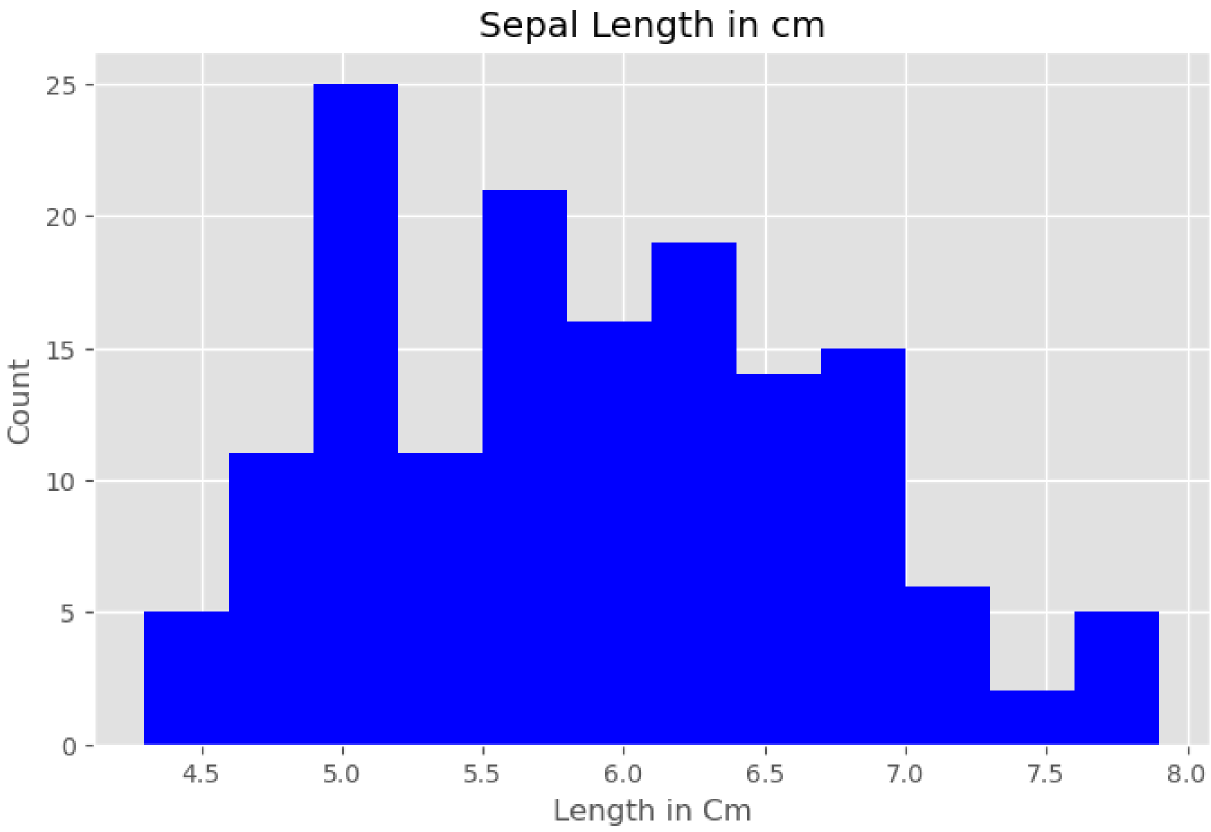


In [45]:

```
# histogram with just one variable - Sepal Length.  We need to isolate the one variabl

plt.figure(figsize = (8, 5))
x = df["SepalLengthCm"]
plt.hist(x, bins = 12, color = "blue")
plt.title("Sepal Length in cm")
plt.xlabel("Length in Cm")
plt.ylabel("Count")
```

Out[45]:

```
Text(0, 0.5, 'Count')
```



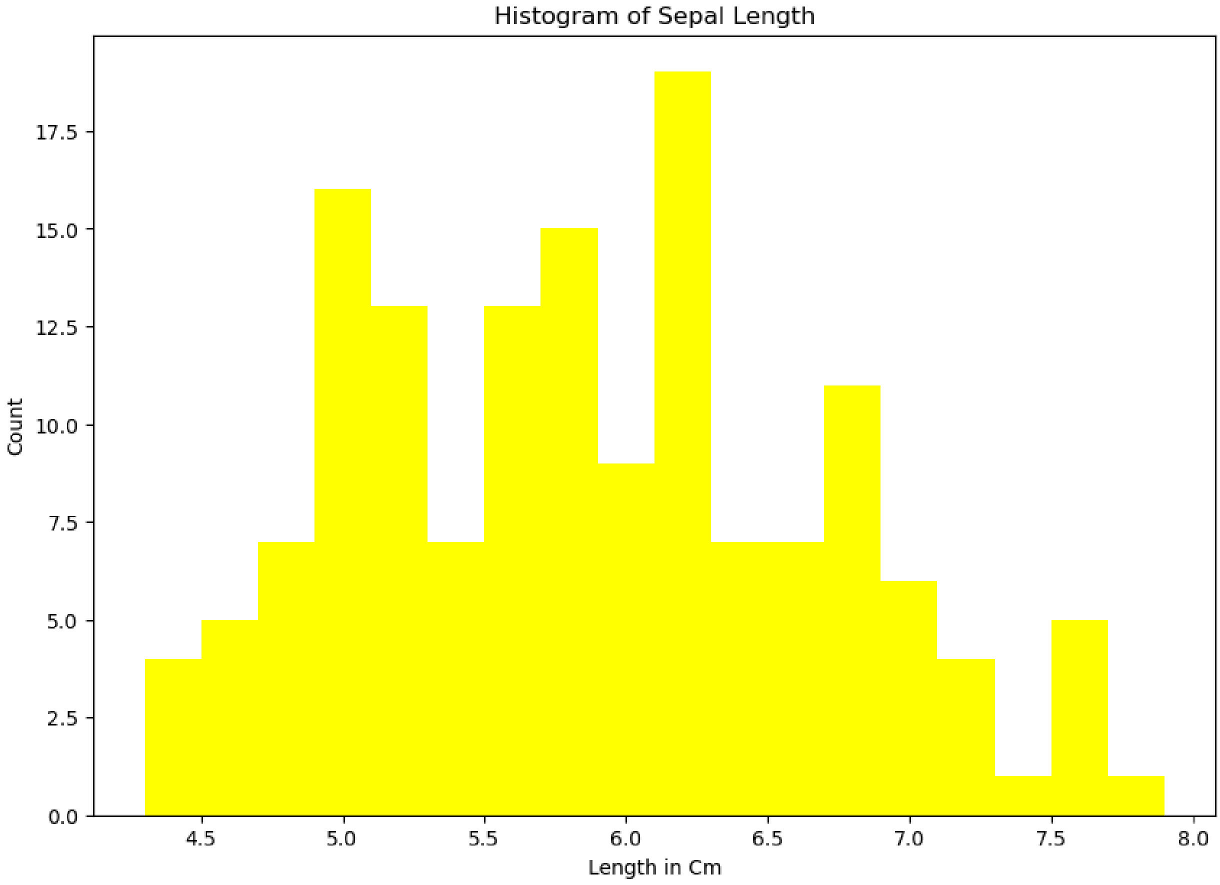
9. Change the plt.title to "Histogram of Sepal Length". Copy and past the code into a new code block, making the change, and use code.

In [13]:

```
# histogram with just one variable - Sepal Length.  We need to isolate the one variabl

plt.figure(figsize = (10, 7))
x = df["SepalLengthCm"]
plt.hist(x, bins = 18, color = "yellow")
plt.title("Histogram of Sepal Length")
plt.xlabel("Length in Cm")
plt.ylabel("Count")
```

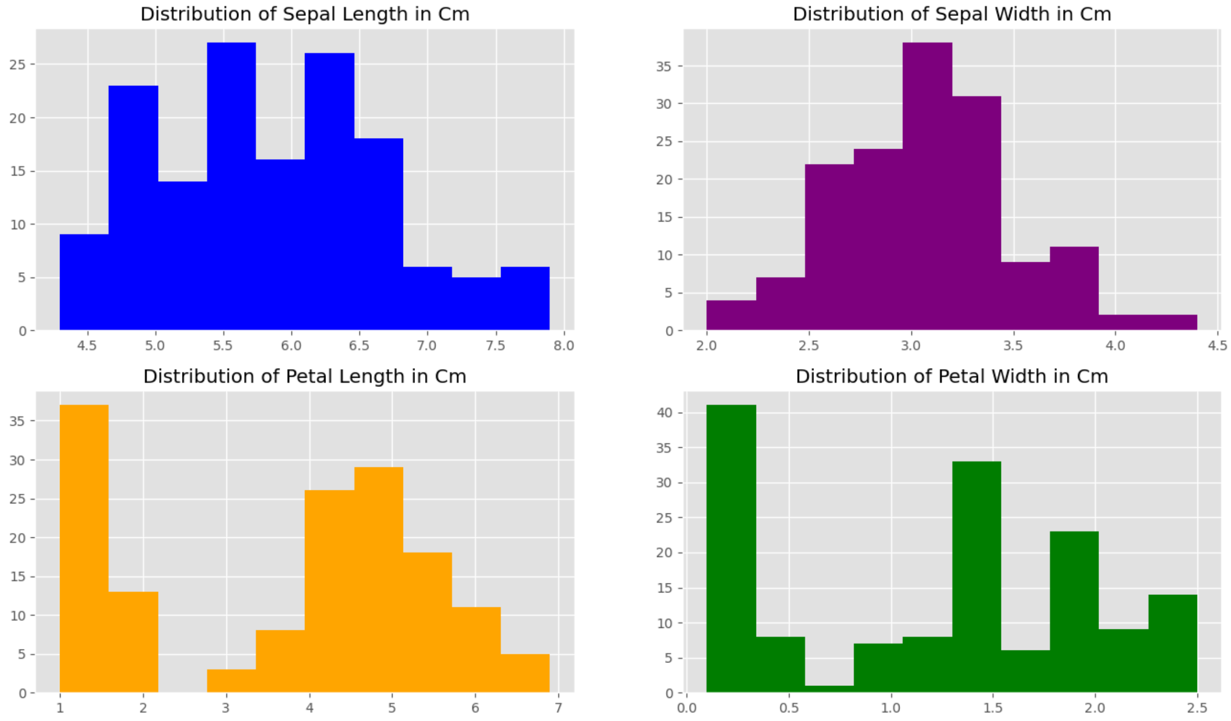
Out[13]:

```
Text(0, 0.5, 'Count')
```



In [14]:

```
# here we are isolating the variable and giving each one a color and a specific number

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');
```


Distribution of Sepal Length in Cm

Distribution of Sepal Width in Cm

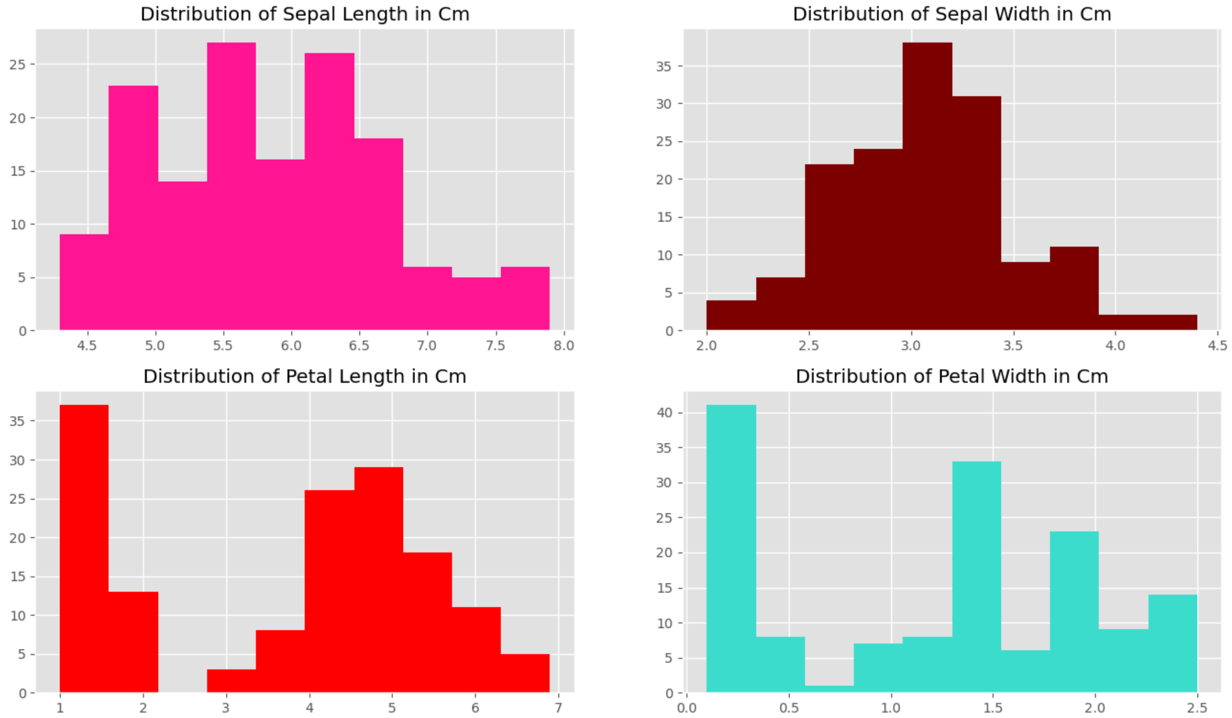
Distribution of Petal Length in Cm

Distribution of Petal Width in Cm

10. Copy and Paste the code above into a new code block and make 3 changes to color. Don't forget to run to see the changes and leave code instead of markdown.

In [19]:

```
# here we are isolating the variable and giving each one a color and a specific number

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 ='deeppink');
axes[0,1].set_title("Distribution of Sepal Width in Cm")
axes[0,1].hist(df['SepalWidthCm'], bins=10, color ='maroon');
axes[1,0].set_title("Distribution of Petal Length in Cm")
axes[1,0].hist(df['PetalLengthCm'], bins=10, color = 'red');
axes[1,1].set_title("Distribution of Petal Width in Cm")
axes[1,1].hist(df['PetalWidthCm'], bins=10, color ='turquoise');
```



Distribution of Sepal Length in Cm

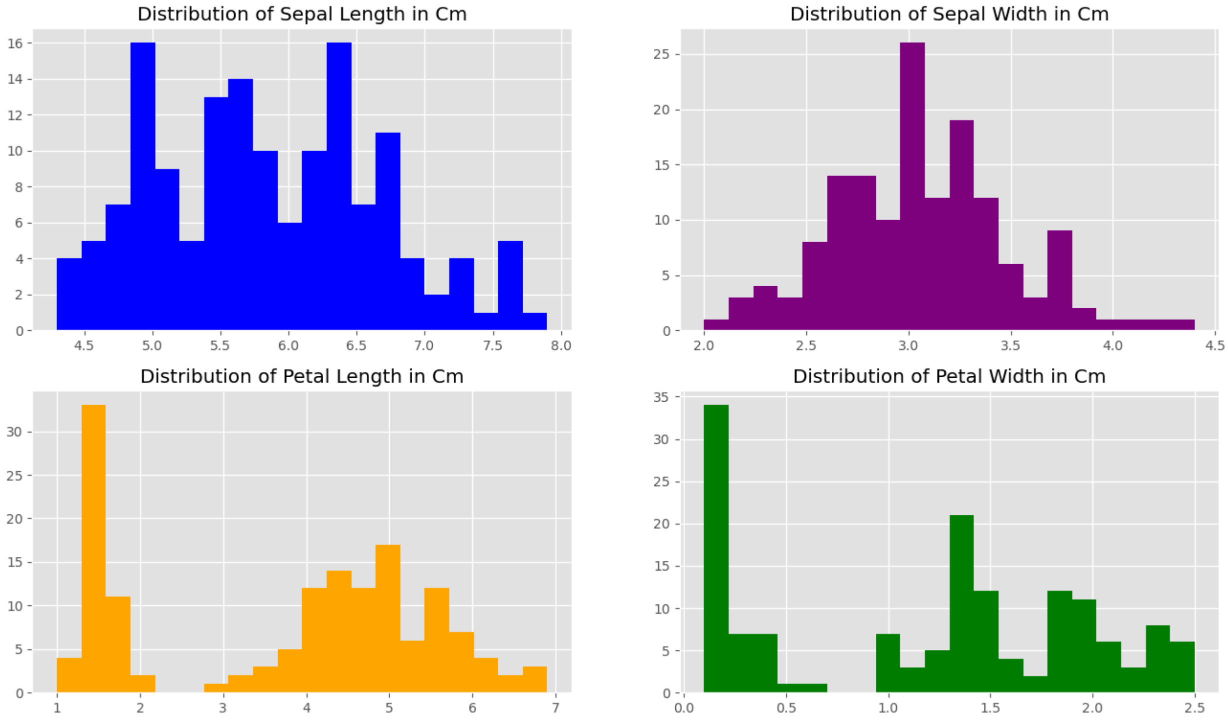
Distribution of Sepal Width in Cm

Distribution of Petal Length in Cm

Distribution of Petal Width in Cm

In [20]:

```
# 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');
```



In [ ]:

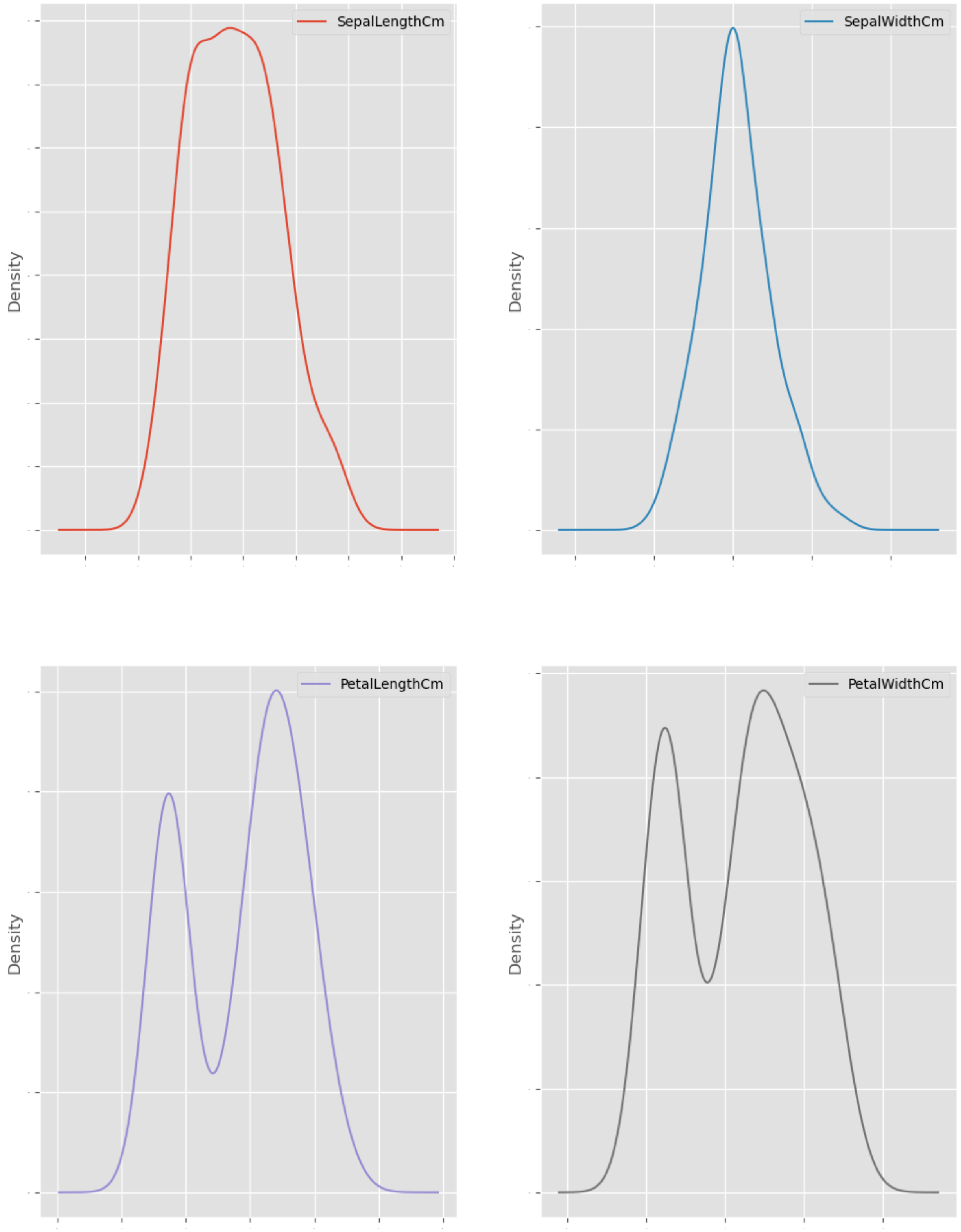
```
```

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 [21]:

```
# create the density plot

df. plot(kind='density', subplots=True, layout=(2,2), sharex=False, legend=True, fonts
plt.show()
```

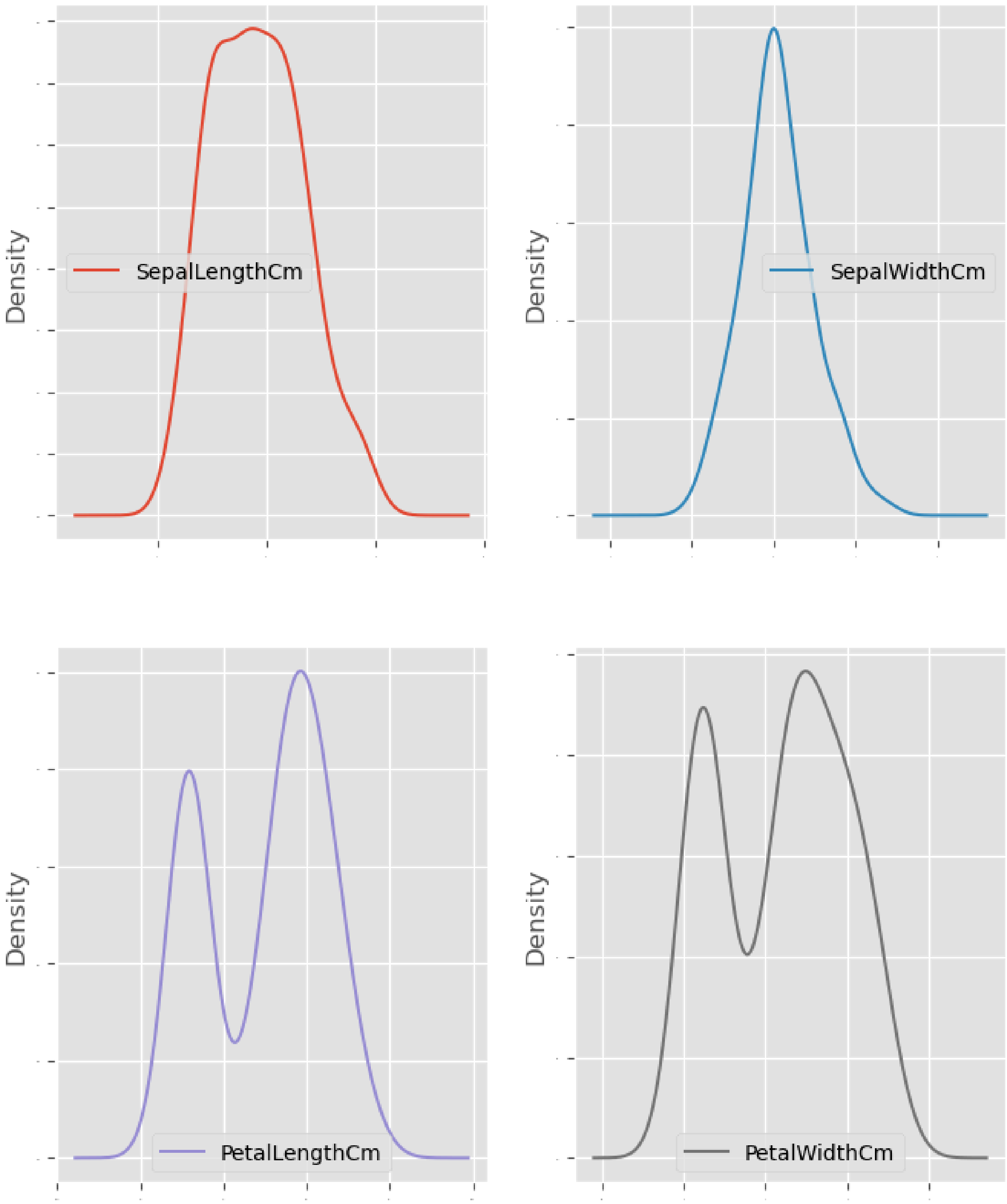


11. Copy and Paste the code above into a new code block and make the figsize smaller. Don't forget to run to see the changes and leave code instead of markdown.

In [25]:

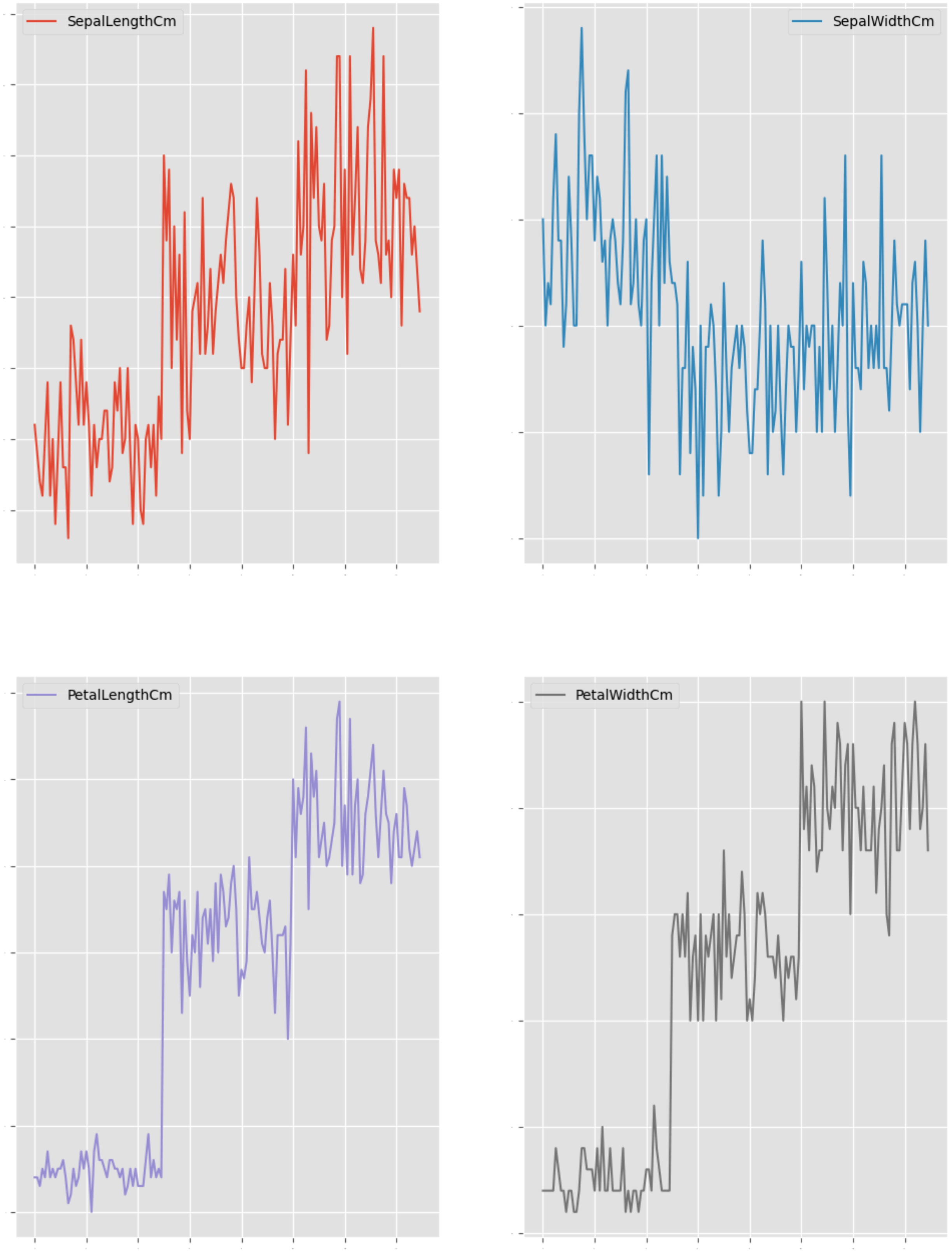
```
# create the density plot
```

```
df. plot(kind='density', subplots=True, layout=(2,2), sharex=False, legend=True, fonts
plt.show()
```



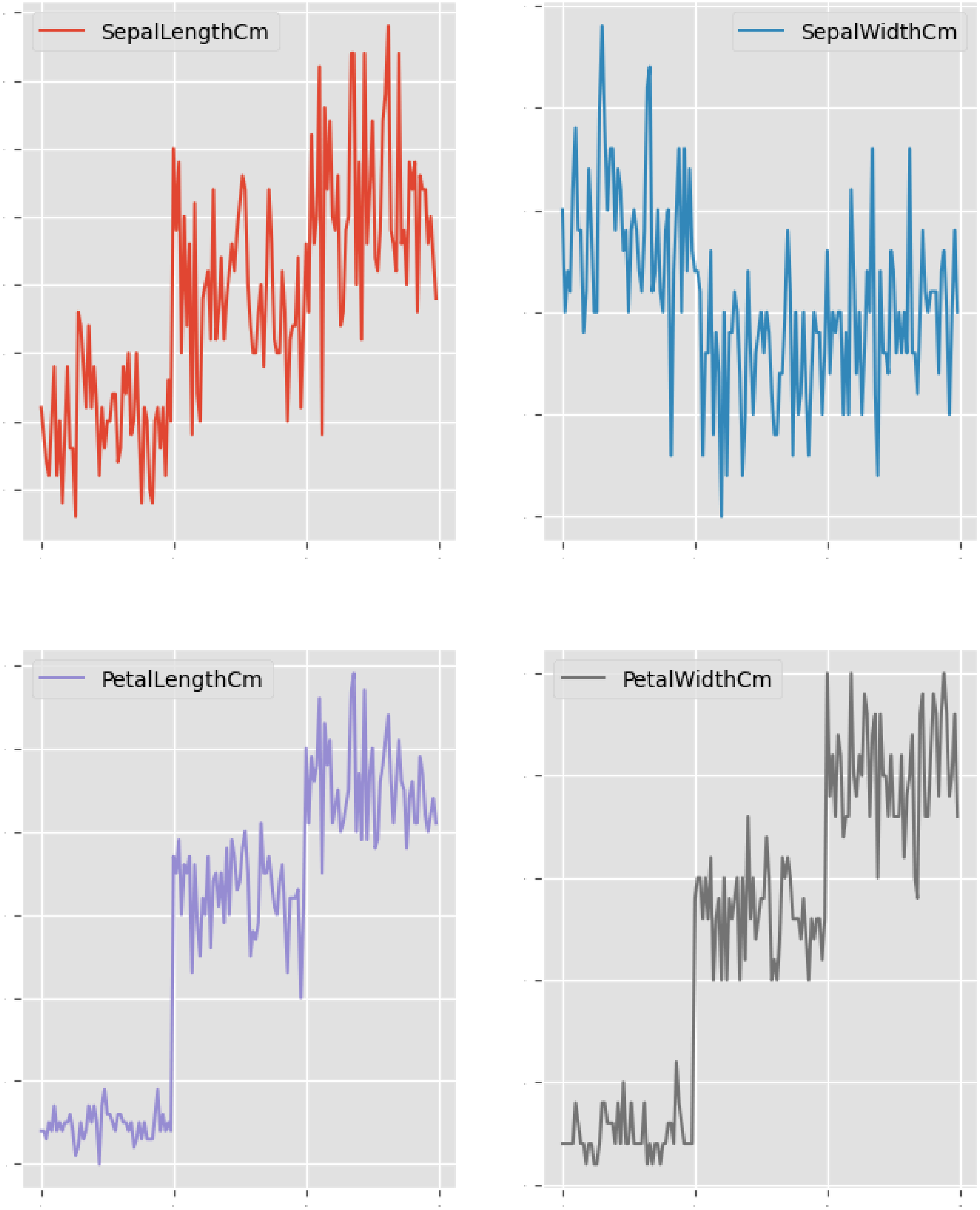
In [26]:

```
# create a line graph

df. plot(kind='line', subplots=True, layout=(2,2), sharex=False, legend=True, fontsize
plt.show()
```



In [27]:

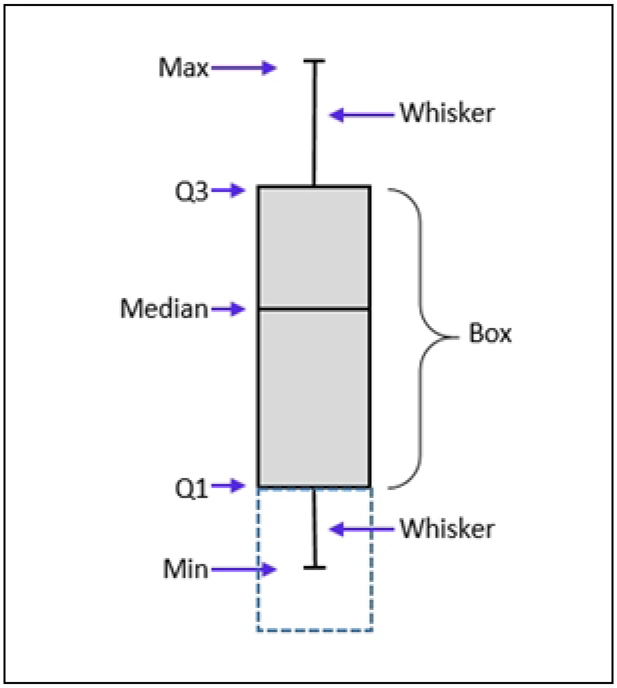
```
# create a line graph

df. plot(kind='line', subplots=True, layout=(2,2), sharex=False, legend=True, fontsize
plt.show()
```



Boxplots

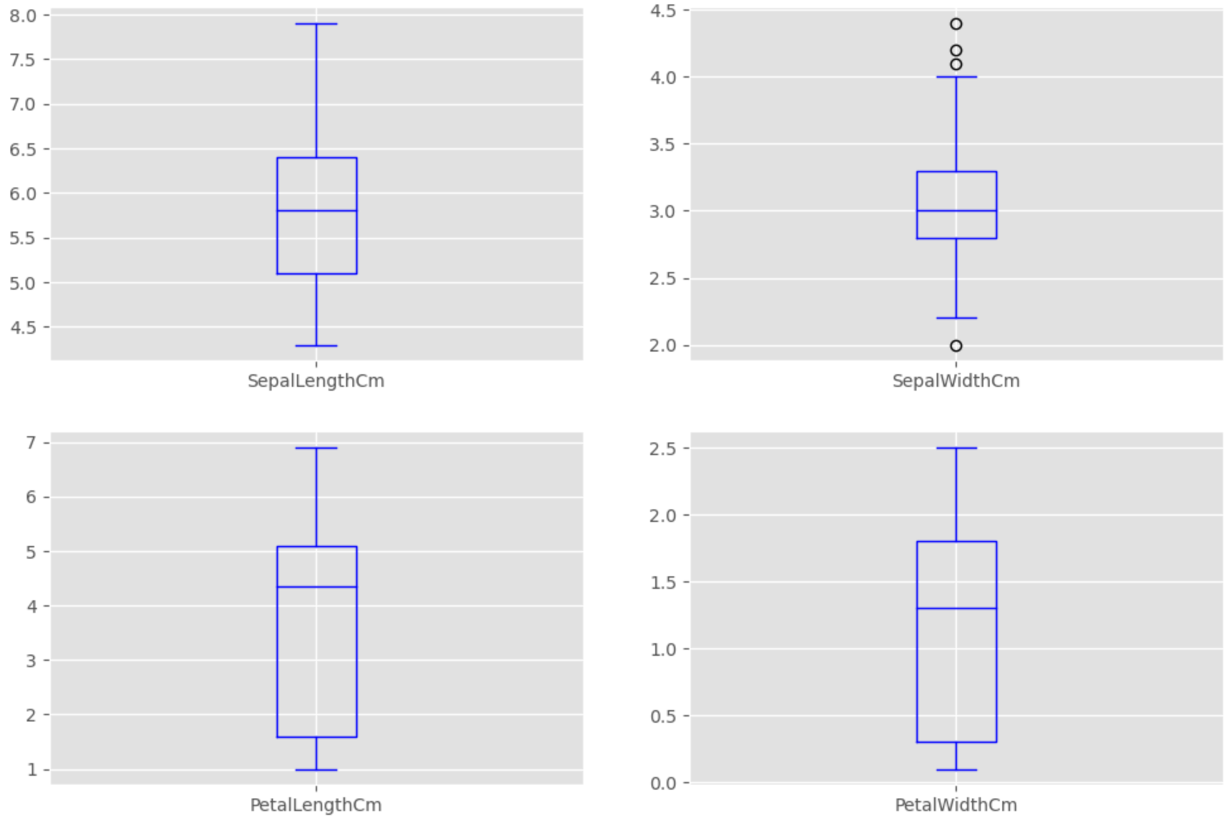
- A box plot is a very good plot to understand the spread, median, and outliers of data

- Below is an illustration of the boxplot

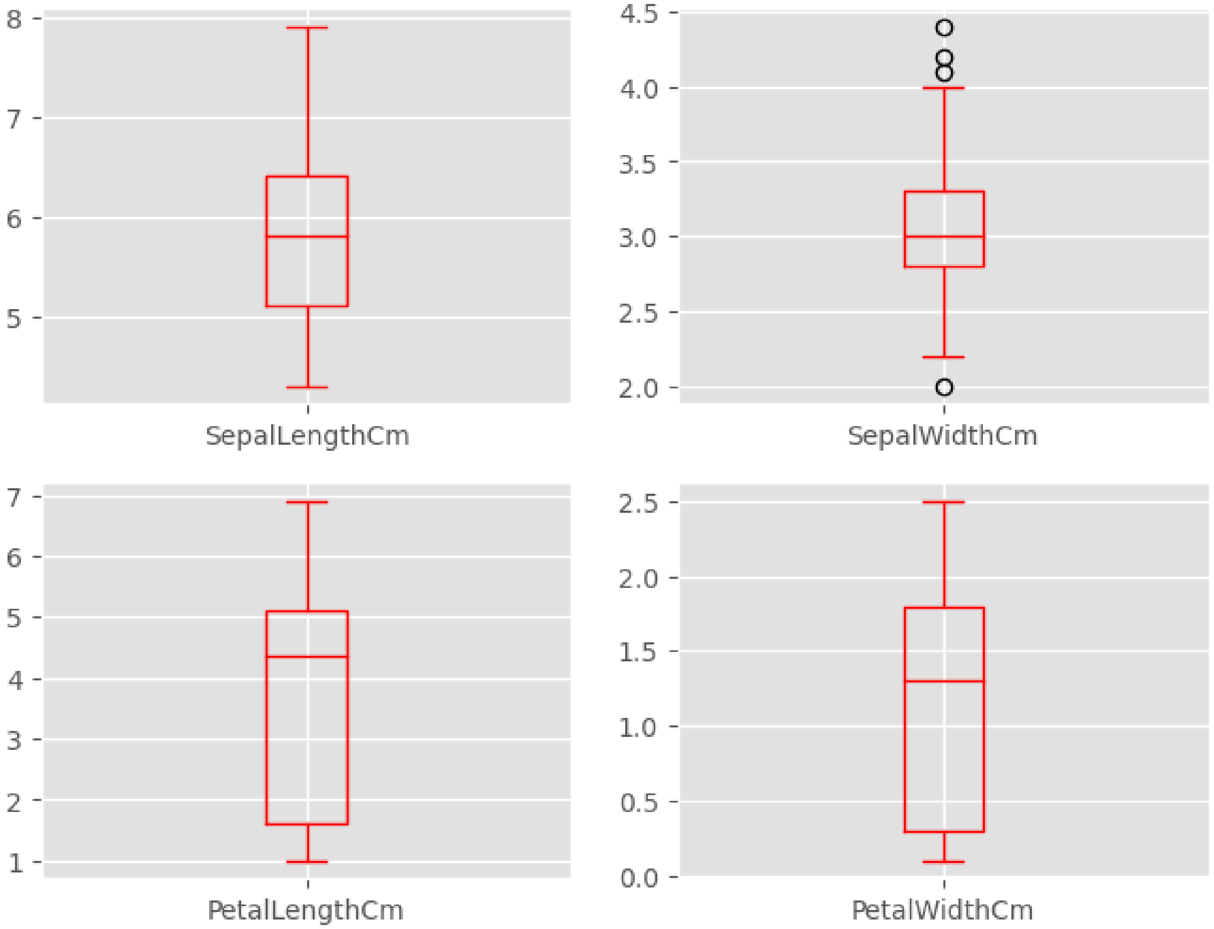


1. Q3: This is the 75th percentile value of the data. It's also called the upper hinge.
2. Q1: This is the 25th percentile value of the data. It's also called the lower hinge.
3. Box: This is also called a step. It's the difference between the upper hinge and the lower hinge.
4. Median: This is the midpoint 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 [28]:
```
# create a box plot

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

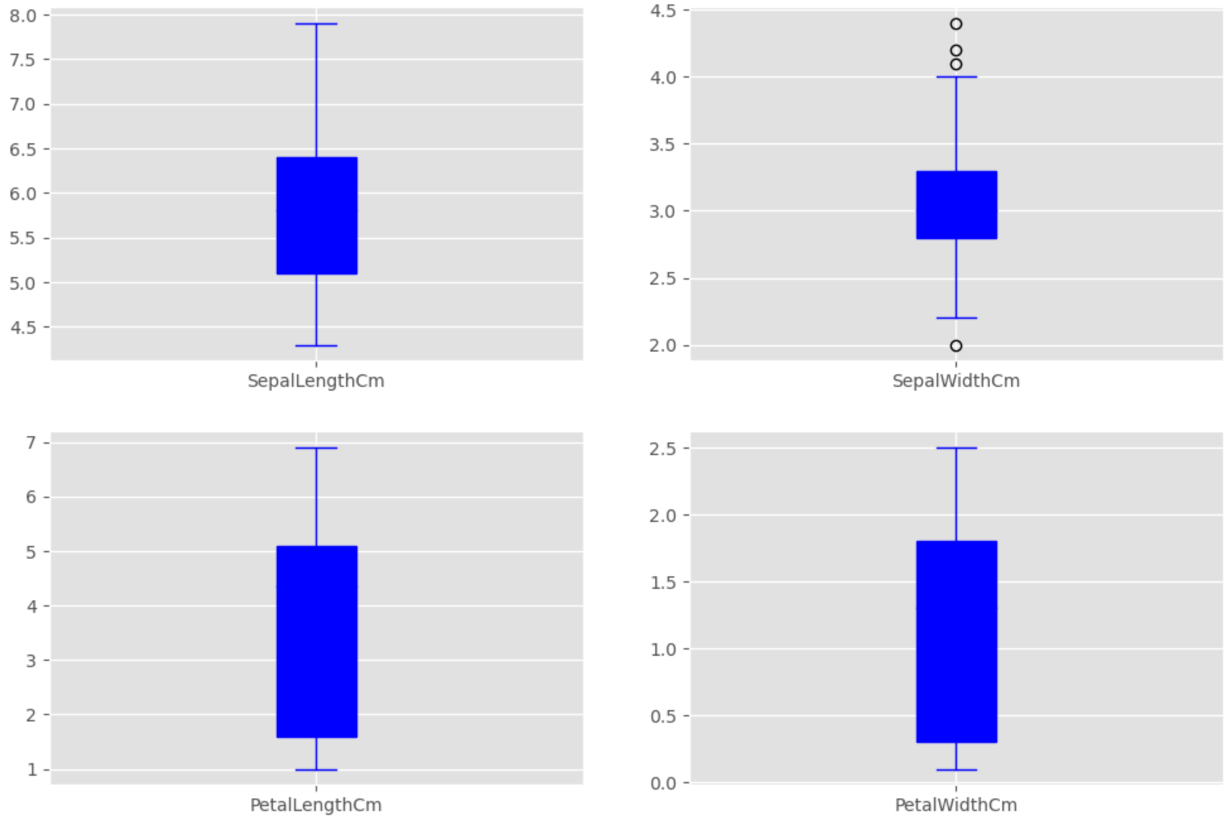


In [29]:
```
# create a box plot

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



In [30]:

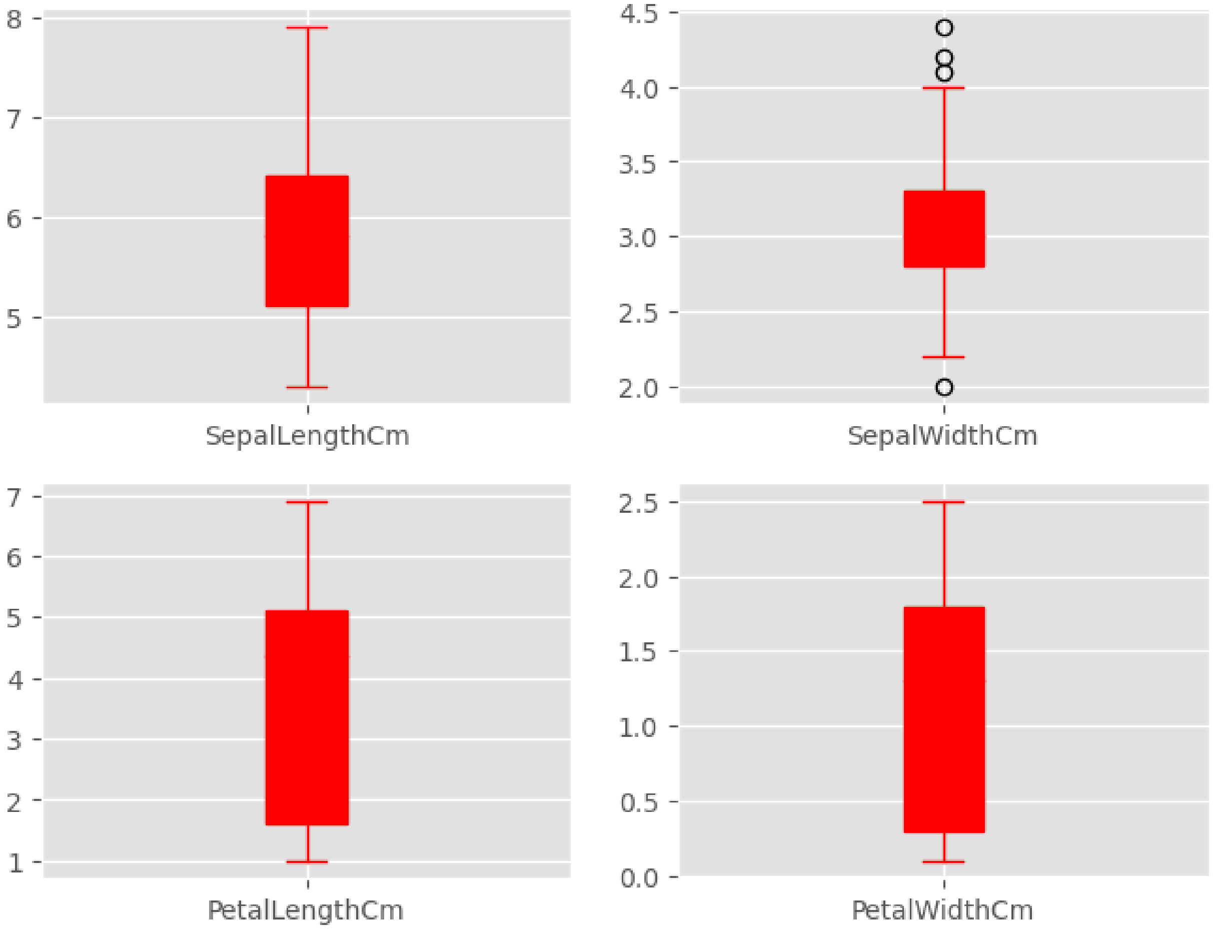
```
# fill the boxes with color, using patch_artist

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



12. Copy and Paste the code above into a new code block and change the color of the boxplots. Don't forget to run to see the changes and leave code instead of markdown.

In [32]:

```
# fill the boxes with color, using patch_artist

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

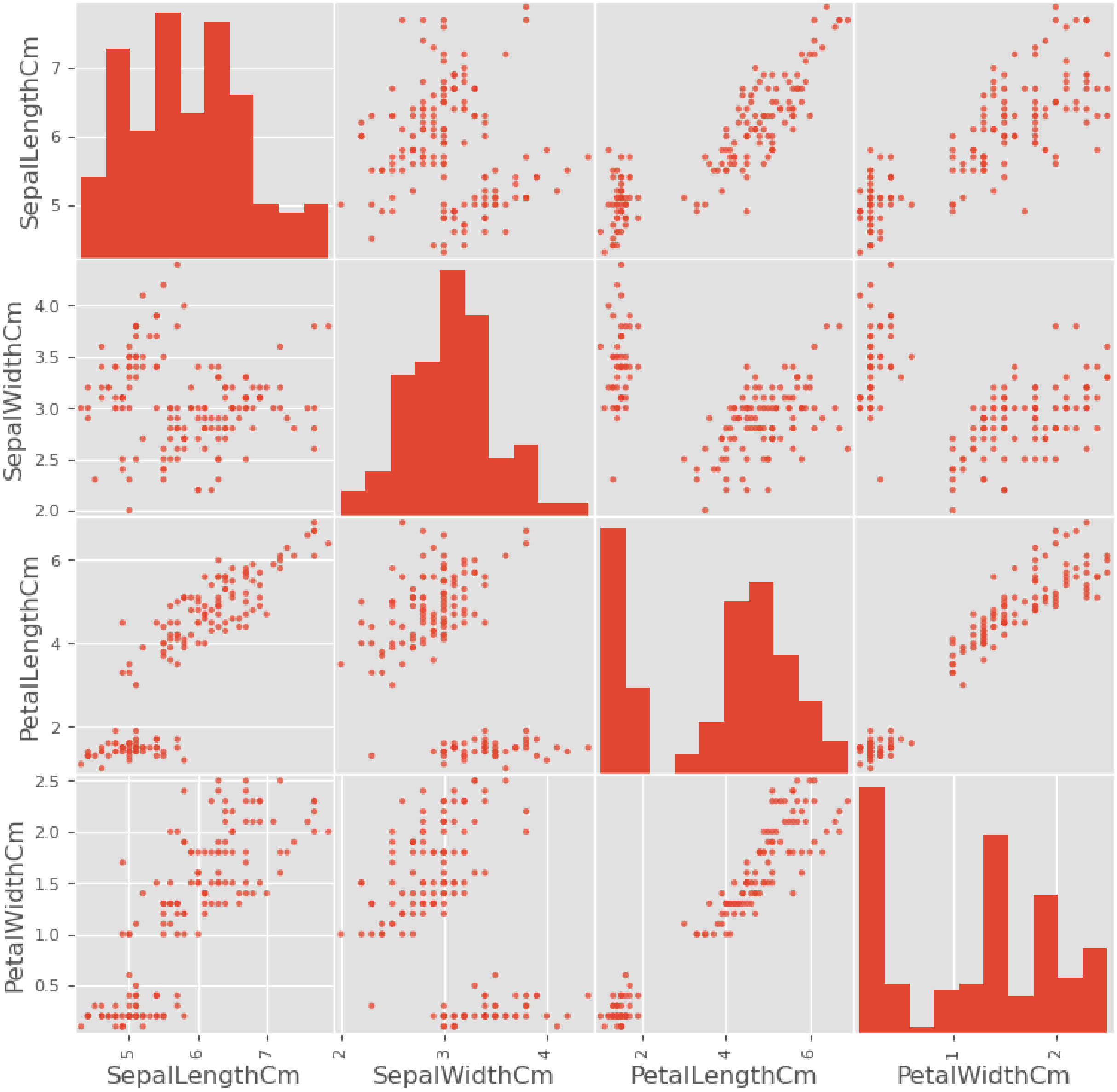


Multivariate Data Visualization

Scatter Matrix Plot

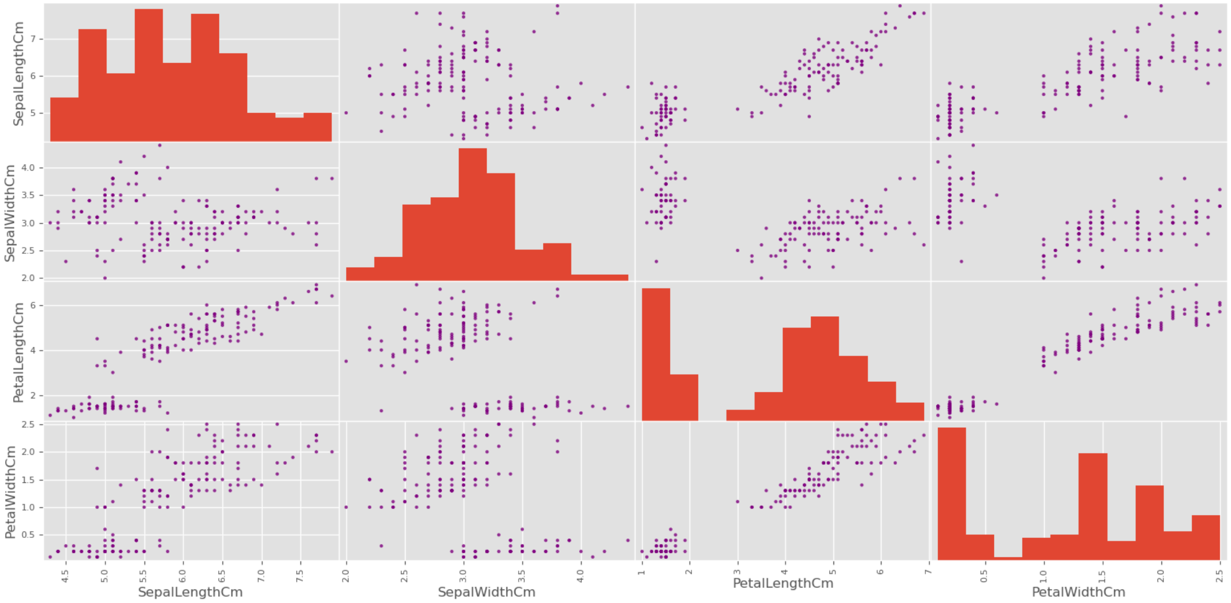
- A scatter plot matrix is a grid (or matrix) of scatter plots. This type of graph is used to visualize bivariate relationships between different combinations of variables. Each scatter plot in the matrix visualizes the relationship between a pair of variables, allowing many relationships to be explored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

In [33]:
```
# create a scatter matrix plot

scatter_matrix (df, alpha=0.8, figsize=(9,9))
plt.show()
```

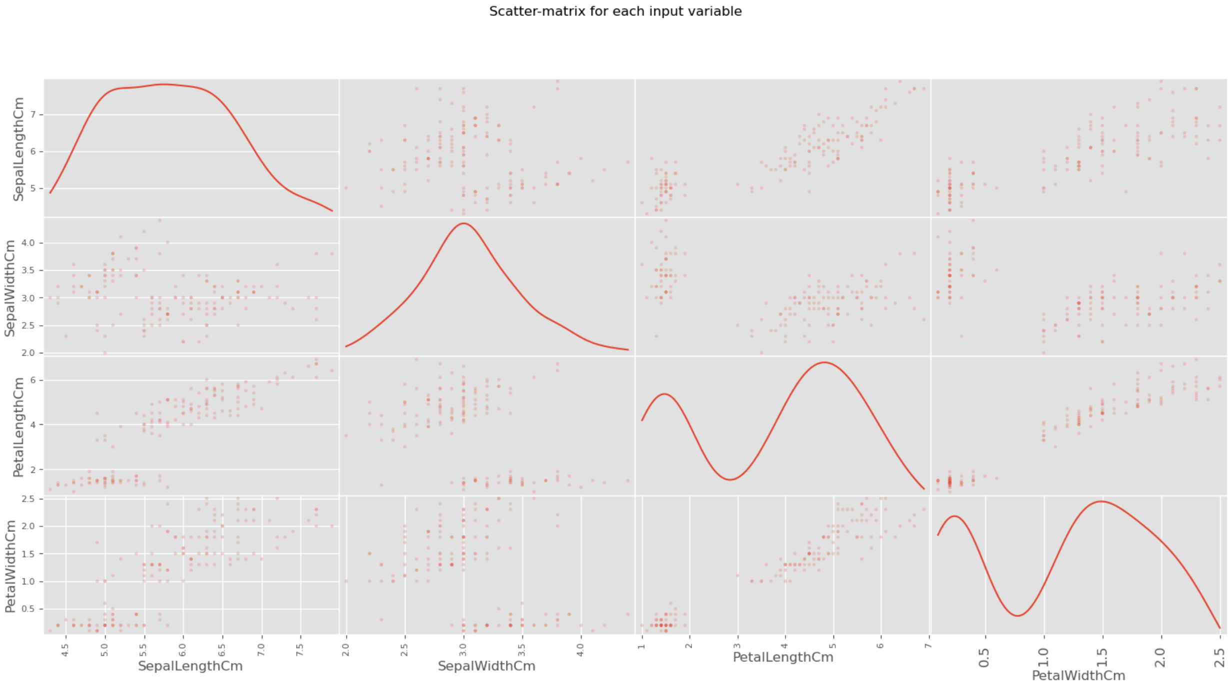


In [34]:

```
# 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()
```



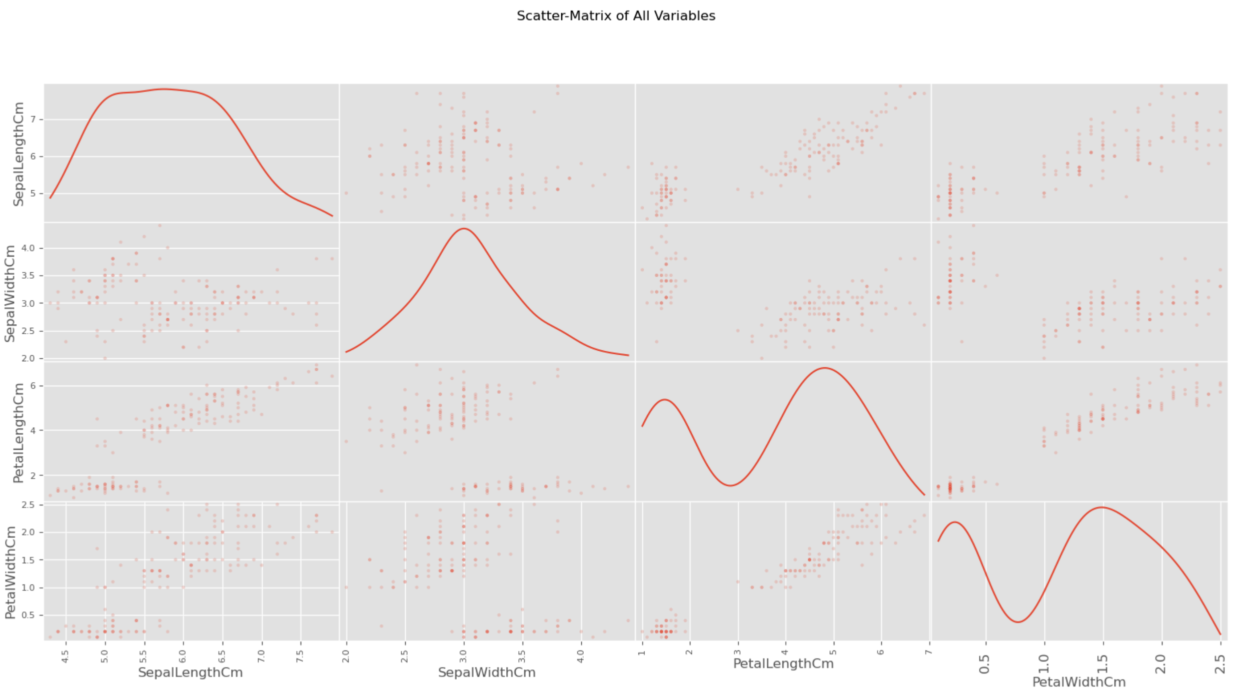
In [35]:
```
# Here you are adding a suptitle.

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)
```



13. Copy and Paste the code above into a new code block and change the suptilte to "Scatter-matrix of all variables" . Don't forget to run to see the changes and leave code instead of markdown.

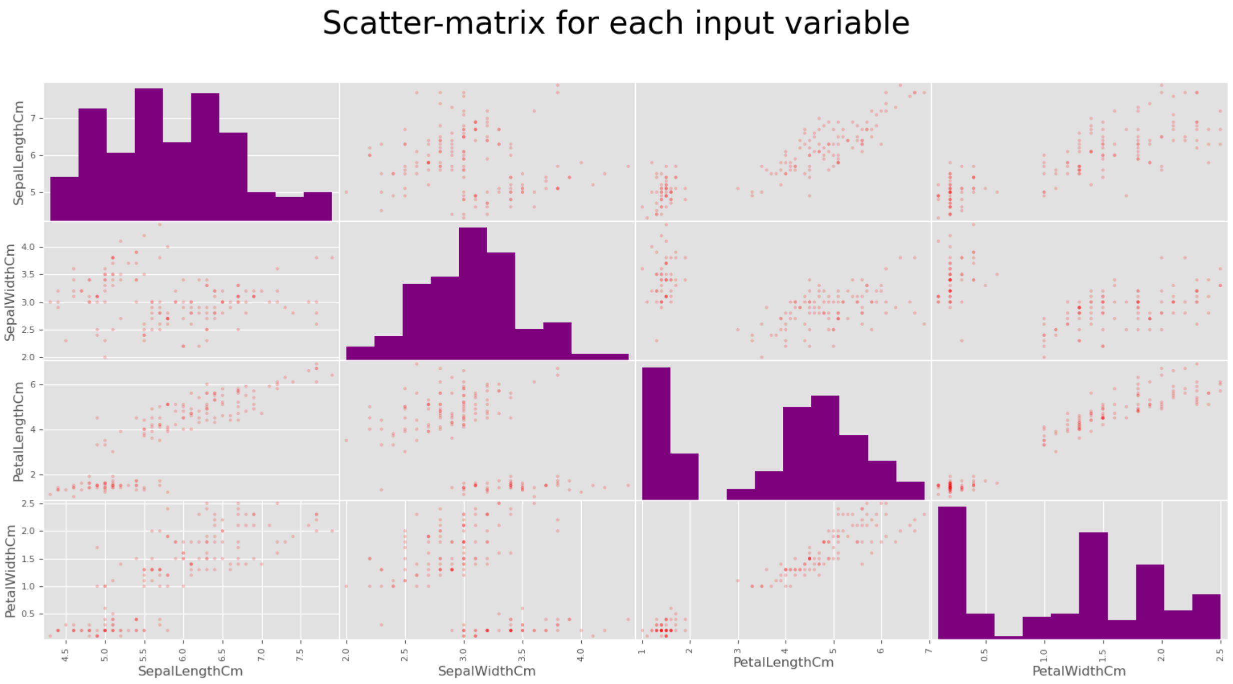
In [37]:
```
# Here you are adding a suptitle.

scatter_matrix(df,alpha=0.2, diagonal = 'kde', figsize=(19,9))
```

```
plt.suptitle('Scatter-Matrix of All Variables')
plt.tick_params(labelsize=12, pad=6)
```



In [38]:
```
# 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[38]:
```
Text(0.5, 0.98, 'Scatter-matrix for each input variable')
```



14. Copy and Paste the code above into a new code block and change the figsize to a size of your choice.

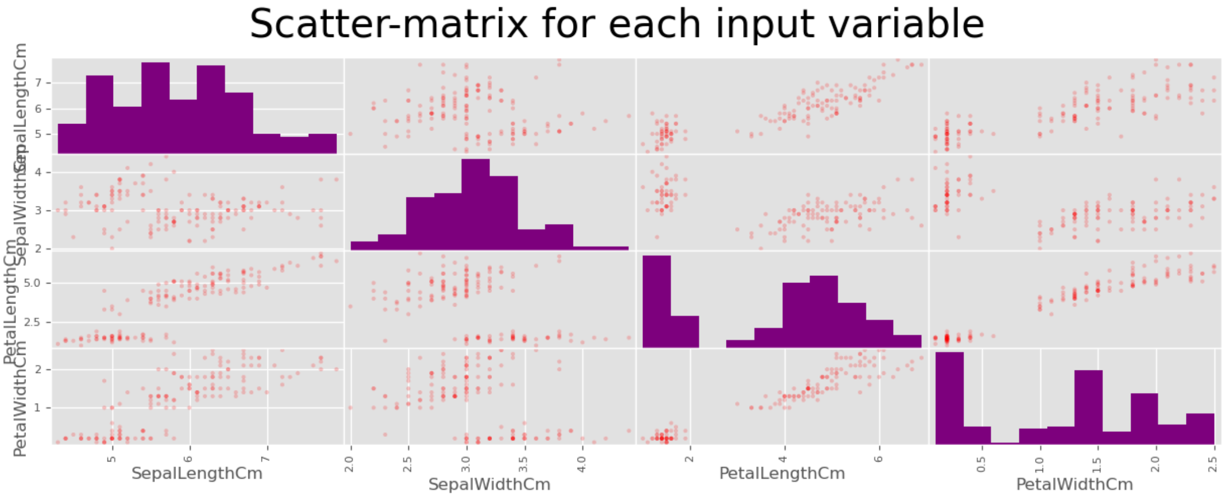
Don't forget to run to see the changes and leave code instead of markdown.

In [39]:

```
# Lastly, you are changing the color to both parts of the plot.

scatter_matrix(df, figsize= (15,5), alpha=0.2,
c='red', hist_kwds={'color':['purple']})
plt.suptitle('Scatter-matrix for each input variable', fontsize=28)
```

Out[39]:

```
Text(0.5, 0.98, 'Scatter-matrix for each input variable')
```



15. Copy and Paste the code above into a new code block and change the fontsize in the subtitle to a size of your choice. Don't forget to run to see the changes and leave code instead of markdown.

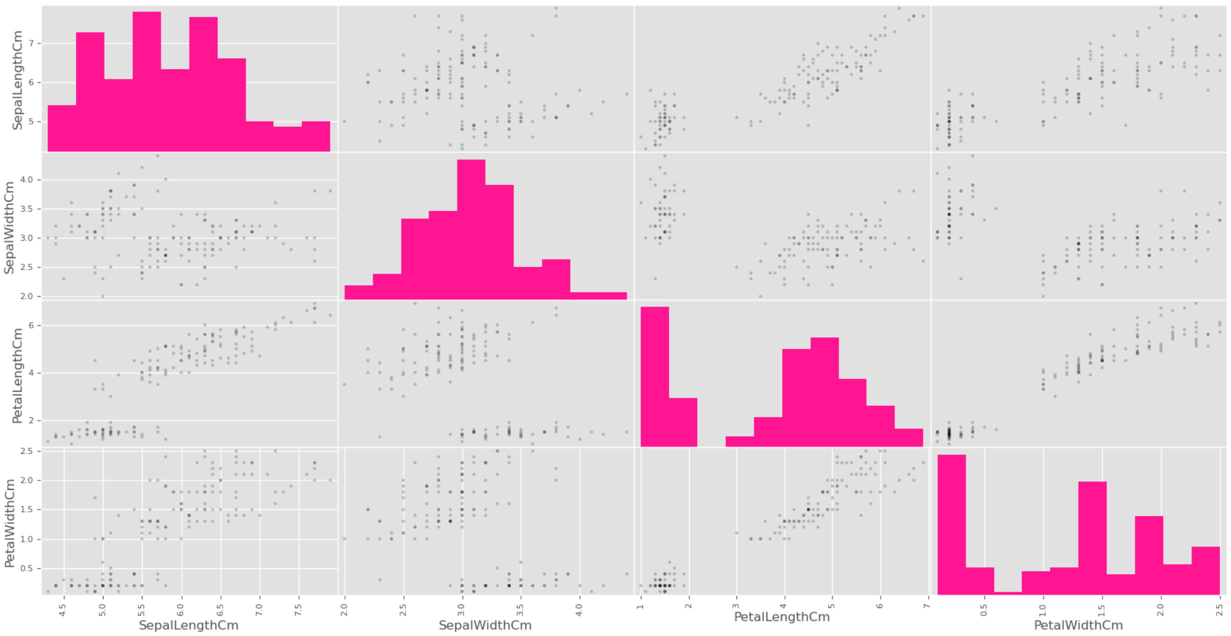
In [44]:

```
# Lastly, you are changing the color to both parts of the plot.

scatter_matrix(df, figsize= (20,10), alpha=0.2,
c='black', hist_kwds={'color':['deeppink']})
plt.suptitle('Scatter-matrix for each input variable', fontsize=16)
```

Out[44]:

```
Text(0.5, 0.98, 'Scatter-matrix for each input variable')
```

Scatter-matrix for each input variable



16. If you have any questions or comments, please create a new code block and enter that information. Don't forget to make this a markdown box and hit run.