Assignment-II: Iris Datasets

1. **Exploratory Data Analysis (EDA)**
   - **Describe the dataset: number of rows, columns, and data types.**
     - **Code:**

```
In [15]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.preprocessing import MinMaxScaler
```

Here in the above code, we have imported different libraries of python for the basic data analysis of the data set.

```
In [16]: df = pd.read_csv('iris.csv')
```

In the 16 no line code we have loaded the ‘titanic.csv’ which is the titanic data set containing numerical data.

```
In [17]: df.shape #number of rows and column
Out[17]: (150, 5)

In [18]: df.shape[0] #number of rows
Out[18]: 150

In [19]: df.shape[1] #number of columns
Out[19]: 5
```

In the line no 17,18,19 we can see that the number of rows and column present in the data-set are displayed as output.

```
In [20]: df.dtypes #datatypes

Out[20]: sepal_length    float64
         sepal_width     float64
         petal_length    float64
         petal_width     float64
         species          object
         dtype: object
```

In line no 9, the data types of the attributes or columns in the data -set are displayed using the python function

- **Summarize numerical features: mean, median, mode, range. Standard deviation**
  - **Code:**

```
In [21]: numeric_df = df.select_dtypes(include='number')
```

The data set contains string values as well as numeric values so to calculate the numerical features we have included only the numerical values in line no 21.

```
In [22]: mean_values = numeric_df.mean()
         mean_values

Out[22]: sepal_length    5.843333
         sepal_width     3.054000
         petal_length    3.758667
         petal_width     1.198667
         dtype: float64
```

In line no 22, we have calculated mean value from the numerical data extracted from the data-set.

```
In [23]: median_values = numeric_df.median()
         median_values

Out[23]: sepal_length    5.80
         sepal_width     3.00
         petal_length    4.35
         petal_width     1.30
         dtype: float64
```

In line no 23, we have calculated median value from the numerical data of the iris data-set.

```
In [24]: #mode calculation
         #In case there are multiple modes, this
         mode_values = numeric_df.mode().iloc[0]
         mode_values

Out[24]: sepal_length    5.0
         sepal_width     3.0
         petal_length    1.5
         petal_width     0.2
         Name: 0, dtype: float64
```

Here in line no 24, we have calculated mode value but if there are multiple modes then ‘df.mode().iloc[0]’ selects the first one value.

```
In [25]: range_values = numeric_df.max() - numeric_df.min()
         range_values

Out[25]: sepal_length    3.6
         sepal_width     2.4
         petal_length    5.9
         petal_width     2.4
         dtype: float64
```

Here in line no 25, we have calculated range value of the data set. This part of code calculates the maximum value and minimum value for each column in the Data Frame. The ‘max()’ function in pandas returns the maximum value along a specified axis. Similarly ‘min()’ calculates the minimum value for each column in the Data Frame.

```
In [26]: std_deviation = numeric_df.std()
         std_deviation

Out[26]: sepal_length    0.828066
         sepal_width     0.433594
         petal_length    1.764420
         petal_width     0.763161
         dtype: float64
```

Here in line no 21, the standard deviation is calculated using ‘std()’ function.

- **Explore categorical features: frequency distribution, unique values.**
  - **Code:**

```
In [27]: categorical_columns = df.select_dtypes(include='object').columns

         for column in categorical_columns:
             frequency_distribution = df[column].value_counts()
             print(f"Frequency distribution for {column}:\n{frequency_distribution}\n")
```

```
Frequency distribution for species:
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

Here in line no 27, we have selected the columns from Data Frame that have data types classified as 'object' and stored in the categorical_columns. Then we have initiated a loop in categorical_columns where in each categorical columns, the frequency distribution of unique values is calculated, and then printed or displayed.

```
In [28]: for column in categorical_columns:
             unique_values = df[column].unique()
             print(f"Unique values for{column}:\n{unique_values}\n")
```

```
Unique values forspecies:
['setosa' 'versicolor' 'virginica']
```

Here in line no 28, we have initiated a loop in categorical_columns where unique values are calculated using 'unique ()' function.

- **Visualize distributions: histograms, box plots.**
  - **Code:**

```
In [29]: #set style for seaborn
         sns.set(style="whitegrid")
```

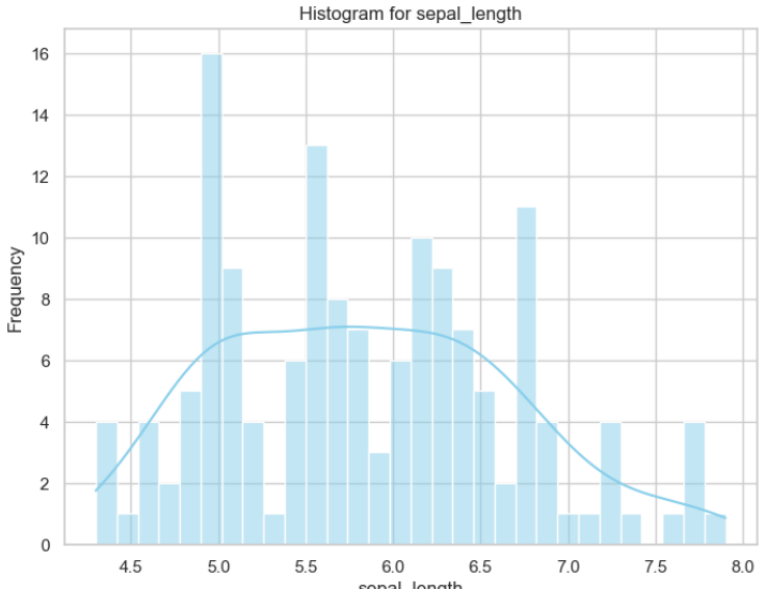
Here in line 29, we have set the overall style of seaborn plots as color white so that there will be more clarity.

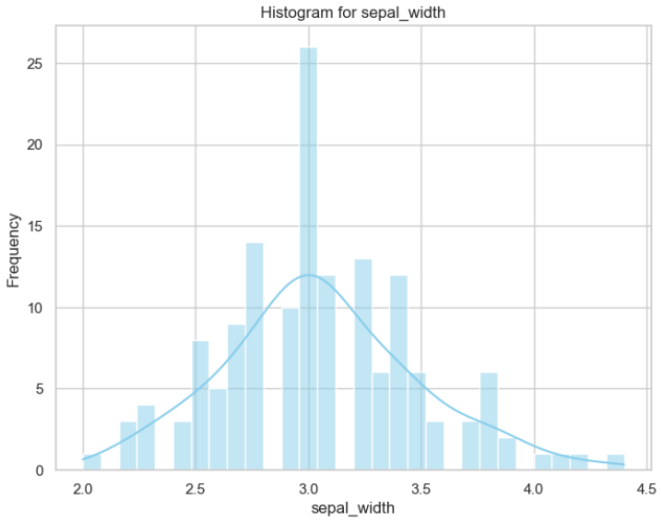
```
In [30]: numeric_columns = df.select_dtypes(include='number').columns
```

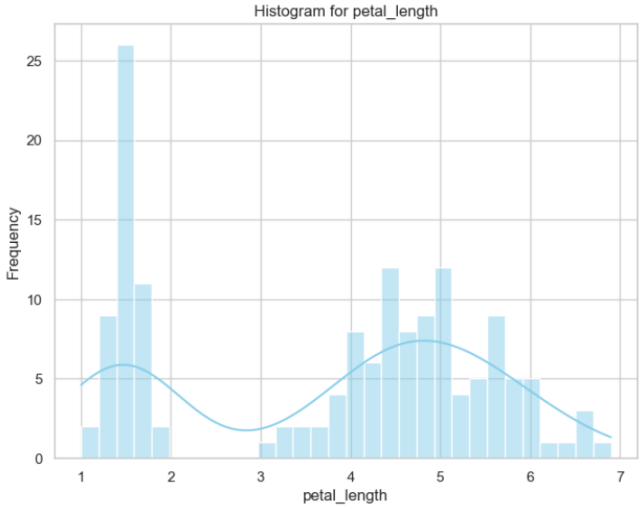
```
In [31]: for column in numeric_columns:
             plt.figure(figsize=(8,6))
             sns.histplot(data=df, x=column,kde=True,bins=30,color='skyblue')
             plt.title(f'Histogram for {column}')
             plt.xlabel(column)
             plt.ylabel('Frequency')
             plt.show()
```

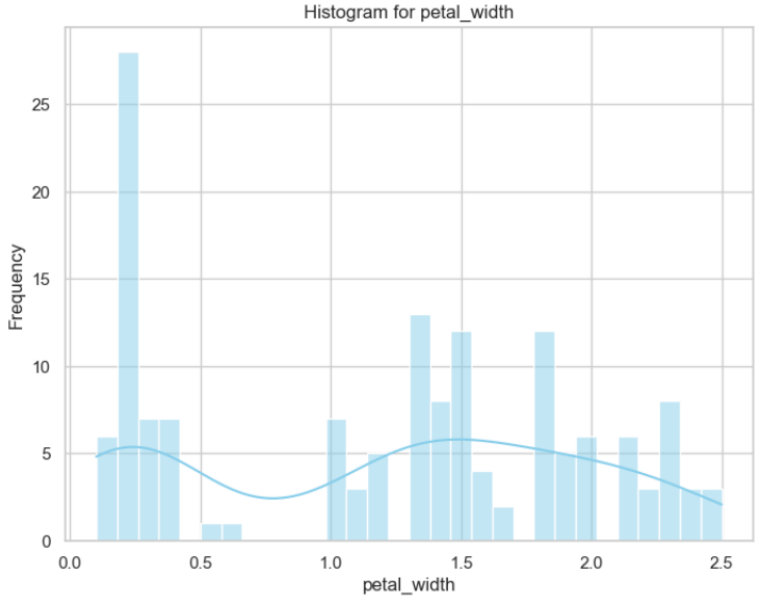
Here in line no 30, the numerical columns are chosen which is essential for the plots.

Here in the line no 31, we have initiated a loop in numeric_columns where the figure size is (8,6) and so on. The histogram visualization is created then. It also includes title, xlabel, ylabel and so on. The output is:





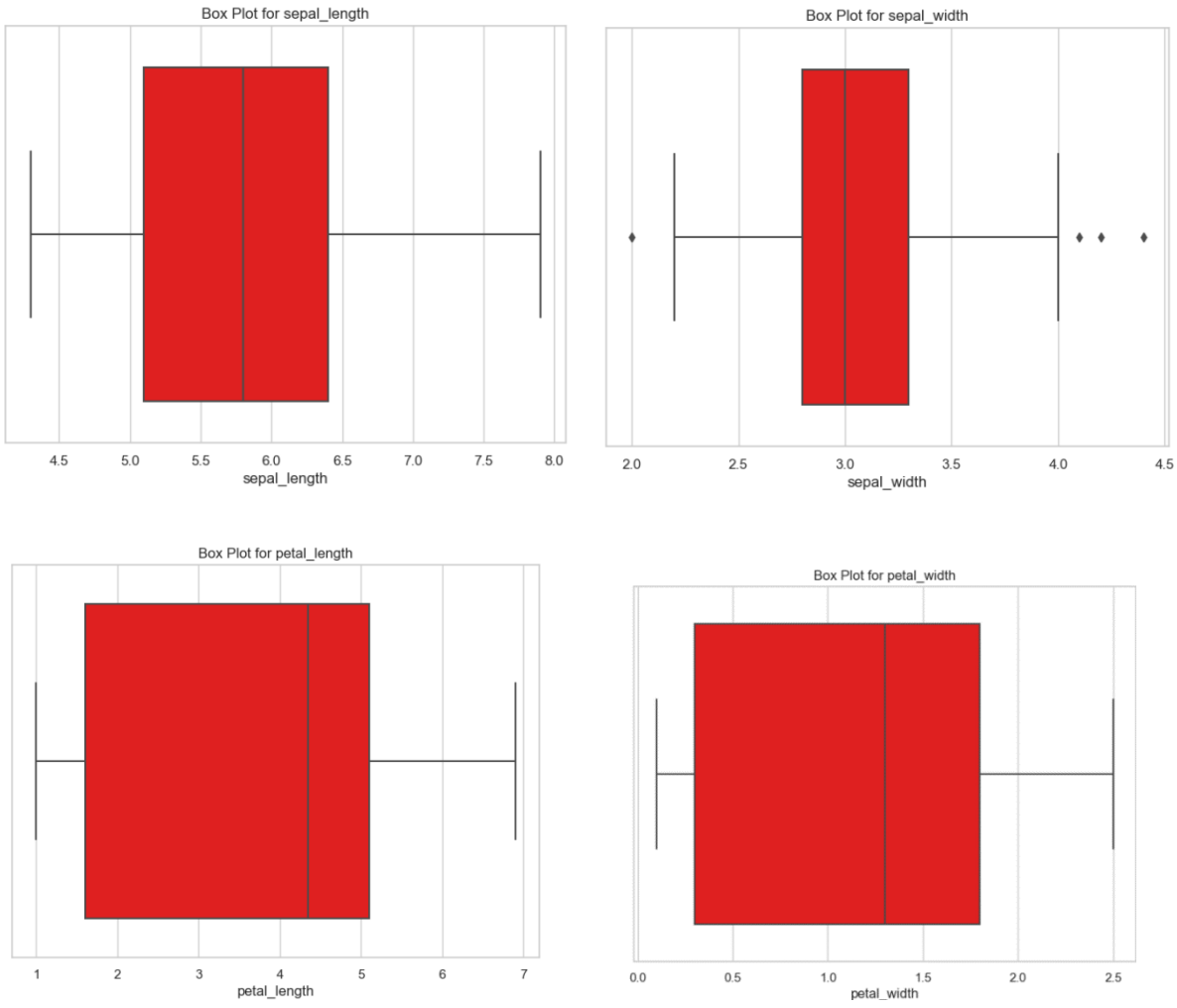




These are the histogram plots of the given data from the data-set.

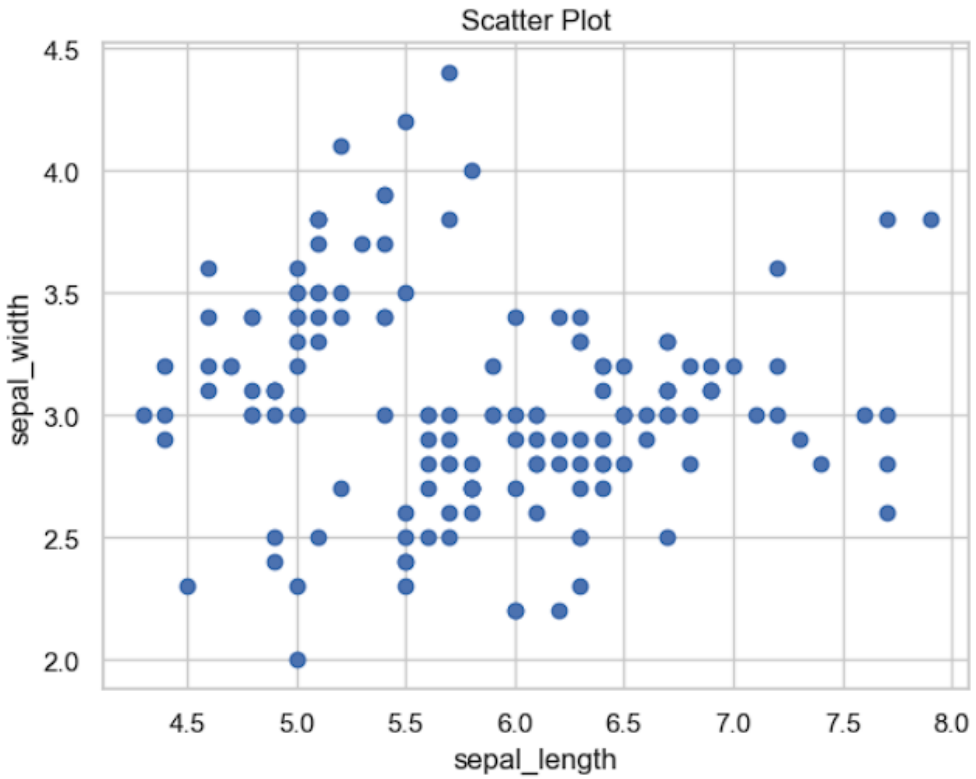
```
In [32]: for column in numeric_columns:
             plt.figure(figsize =(8,6))
             sns.boxplot(x=df[column],color='red')
             plt.title(f'Box Plot for {column}')
             plt.xlabel(column)
             plt.show()
```

Here in the line no 32 we have initiated a loop in numeric_columns where the figure size is (8,6) and so on. The Box Plot visualization is created then. It also includes title, xlabel, and so on. The output is:



These are the Box plots of the given data from the data-sets

In [34]:
```
plt.scatter(numeric_df['sepal_length'], numeric_df['sepal_width'])
plt.title('Scatter Plot')
plt.xlabel('sepal_length')
plt.ylabel('sepal_width')
plt.show()
```



Here in line 34, the scatter plot of the data set is shown.

2. Handling Missing Values

- **Identify missing values: count missing values per column**
  - **Code:**

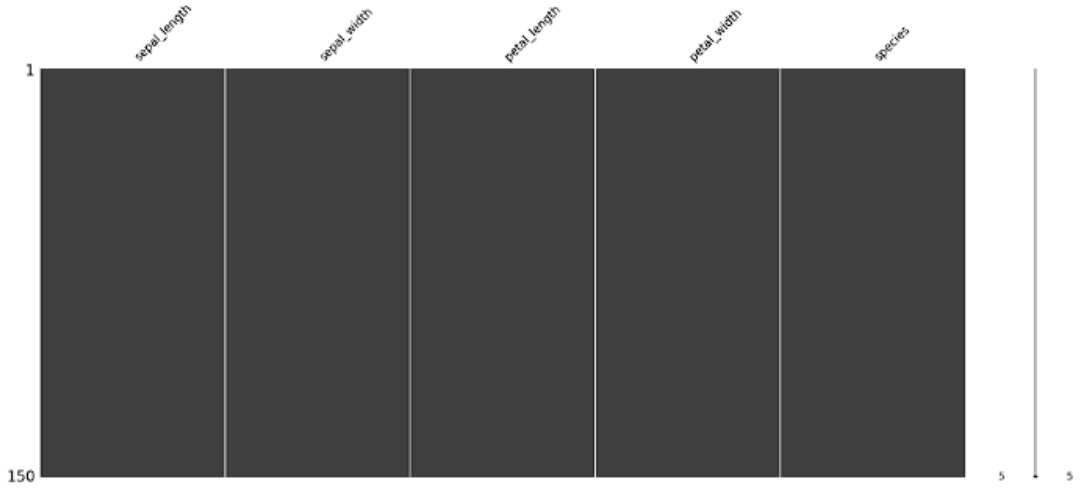
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import missingno as msno
        from sklearn.preprocessing import MinMaxScaler
```

Here in the above code, we have imported different libraries of python for the basic data analysis of the data set.

```
In [2]: df = pd.read_csv('iris.csv') #load data-set
```

In the 2 no line code we have loaded the ‘iris.csv’ which is the iris data set containing numerical data.

```
In [3]: msno.matrix(df)
Out[3]: <Axes: >
```



Here in line no 3, the line generates a matrix plot using ‘missingno’ to visualize missing values in the DataFrame. The matrix function displays a grid where each row corresponds to a row in the DataFrame, and each column corresponds to a variable(feature) in the DataFrame. The white cells represent the non-missing values.

```
In [4]: missing_values = df.isnull().sum() #missing values per column
        missing_values

Out[4]: sepal_length    0
        sepal_width     0
        petal_length    0
        petal_width     0
        species         0
        dtype: int64
```

In the line no 9, we can see that missing_values series provide a summary of the count of missing values for each column in the Data Frame. The line 'df.isnull().sum()' states that Boolean DataFrame is obtained which can be True or False then the 'sum()' is applied. Since 'True' is equivalent to 1 and 'False' is equivalent to 0 when summing, this operation counts the number of 'True' values (missing values in each column. The result is a Series where the index corresponds to column names, and the values represent the count of missing values in each column.

- **Decide on the treatment strategy: impute missing values or remove them**
  - **Explain:**
    Here the decision I Have chosen is null, as there are no missing values present.

- **Implement chosen strategies: impute missing values using mean, median, mode, or advanced methods**
  - **Explain:**
    As the decision chosen is null there will be no implementation of strategies as there are no missing values present.

- **Validate the effectiveness of the chosen strategy: compare before and after results.**
  - **Code:**

```
In [5]: numeric_df = df.select_dtypes(include = 'number')
        numeric_df.fillna(numeric_df.mean() , inplace=True)
```

```
In [6]: numeric_df.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
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

Here in line no 5,6, the before version of the data set is shown.

```
In [7]: numeric_df_imputed = numeric_df.copy()
        numeric_df_imputed.fillna(numeric_df.mean(),inplace=True)
```

```
In [8]: numeric_df_imputed.describe()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width
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

Here in line no 23, the after version of the data is shown, which shows the effectiveness

3. **Dealing with Duplicate Values**
    - **Detect duplicate rows: identify rows with identical values across all columns.**
        - **Code:**

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import MinMaxScaler
```

Here in the above code, we have imported different libraries of python

```
In [2]: df = pd.read_csv('iris.csv') #load data-set
```

In the 2 no line code we have loaded the ‘iris.csv’ which is the titanic data set containing numerical data.

```
In [3]: df_no_duplicates = df.drop_duplicates()
        df_no_duplicates
```

Out[3]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

147 rows × 5 columns

In line no 3, we have deleted the duplicated rows from the Data Frame.

- **Decide on the treatment strategy: remove duplicated or keep one instance**
  - **Explain:**
    The decision taken for the treatment strategy is to remove the duplicated.

- **Implement chosen strategies: remove duplicated based on specific criteria.**
  - **Code:**

```
In [4]: duplicate_rows = df.duplicated(subset=['petal_length','petal_width'])
        df_no_duplicates = df[~duplicate_rows]
        df.duplicated().sum()
        df_no_duplicates.shape

Out[4]: (102, 5)
```

Here in line 4, the duplicated data in ‘petal_length’ and ‘petal_width’ column which are removed and the output is displayed.

- **Validate the effectiveness of the duplicate removal: assess the impact on data integrity.**
  - **Code:**

```
In [5]: summary_before = df.describe()
        summary_before
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width
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

Here in line no 5, the before version of the data set is shown.

```
In [7]: summary_after = df_no_duplicates.describe()
        summary_after
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width
count	102.000000	102.000000	102.000000	102.000000
mean	6.056863	3.025490	4.248039	1.406863
std	0.833876	0.434111	1.639532	0.710913
min	4.300000	2.000000	1.000000	0.100000
25%	5.500000	2.725000	3.625000	1.000000
50%	6.050000	3.000000	4.700000	1.500000
75%	6.700000	3.300000	5.475000	2.000000
max	7.900000	4.400000	6.900000	2.500000

Here in line no 7, the after version of the data is shown, which shows the effectiveness

- **Conclusion**
  - At last, from this assignment I have learned many things such as basic commands to calculate the numerical data from the given data-set. The frequency distribution and unique values also give an information about the data-set. The visual representation gives more clarity to the study.