Assignment No. 2

Problem Statement: Write a python script to find basic descriptive statistics using summary, quartile function, etc on iris datasets.

Objective: The objective of this assignment is to create a Python script that computes and displays basic descriptive statistics for the iris dataset, such as the summary statistics, quartiles, and other relevant metrics. The dataset should be loaded into memory, processed, and analyzed to understand the distribution of the features.

Prerequisite :

1. A Python setup with pandas, numpy, and sklearn libraries.
2. Basic knowledge of statistics (mean, median, mode, quartiles, etc.).
3. Familiarity with pandas for data handling.
4. Iris dataset access (sklearn.datasets or external source).
5. A text editor or IDE to run the script.

Theory :

Descriptive statistics summarize the key features of a dataset in a quantitative way, simplifying data analysis to identify patterns and trends without needing to examine the entire dataset. Central tendency indicates a central value in a probability distribution, represented by the mean, median, and mode.

1. Mean : -

- The mean is the most common measure of central tendency, known as the average.
- It is represented as μ for populations and x̄ for samples.
- The mean is calculated by dividing the sum of all values by the count of values.
- It is sensitive to outliers, which can skew results.
- Thus, relying solely on the mean may not offer sufficient insight for decisions.

2. Median : -

- The median is the value that splits a dataset into two equal parts.
- To determine the median, first sort the data in ascending order.
- The median of this dataset is the number at (n+1)/2 th position, if n is odd.
- If n is even, then the median is the average of the (n/2) th number and (n+2)/2 th number.

- The median is not significantly affected by extreme values, making it a robust measure.
- It is especially useful in skewed distributions or when outliers may distort other measures of central tendency.

3. Mode : -

- The mode is the most frequently occurring value in a dataset.
- A dataset can be unimodal (one mode), bimodal (two modes), or multimodal (multiple modes).
- It is useful for identifying the most common category in categorical data.
- The mode can apply to both numerical and non-numerical data.
- It is not affected by extreme values like the mean and median.

4. Quartile :-

- Quartiles divide a dataset into four equal parts.
- The first quartile (Q1) marks the 25th percentile, while the second quartile (Q2) is the median (50th percentile).
- The third quartile (Q3) indicates the 75th percentile.
- Quartiles help in understanding data distribution and identifying outliers.

5. Standard Deviation :-

- Standard deviation quantifies the amount of variation or dispersion in a dataset.
- It indicates how much individual data points deviate from the mean of the dataset.
- A low standard deviation means the values are close to the mean, while a high standard deviation indicates greater spread.
- It is calculated by taking the square root of the variance.
- Standard deviation is useful for comparing the variability of different datasets.

Algorithm (if any to achieve the objective)

- **Import Libraries:** Use import pandas as pd, import numpy as np, and visualization libraries like import matplotlib.pyplot as plt and import seaborn as sns.
- **Load Dataset:** Load the iris dataset with pd.read_csv('path_or_url').
- **Inspect Data:** Display the first few rows using iris_data.head().

Compute Basic Statistics:

- **Mean**: Calculate the average using iris_data.mean().
- **Median**: Find the middle value using iris_data.median().
- **Mode**: Identify the most common value using iris_data.mode().
- **Minimum**: Get the smallest value using iris_data.min().

- **Maximum**: Get the largest value using iris_data.max().
- **Variance**: Measure how spread out the numbers are using iris_data.var().
- **Standard Deviation**: Calculate the average distance of each number from the mean using iris_data.std().
- **Quartiles**: Find the 25th, 50th (median), and 75th percentiles using iris_data.quantile([0.25, 0.5, 0.75]).
- **Skewness**: Measure the asymmetry of the data distribution using iris_data.skew().
- **Kurtosis**: Assess the "tailedness" of the data distribution using iris_data.kurt().
- **Distribution Plot:** Use sns.distplot(iris_data['feature_name']) for selected features.
- **Box Plot:** Use sns.boxplot(data=iris_data) to visualize data distribution and detect outliers.

Code & OutPut :-

```
# Importing necessary libraries for data analysis and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# Importing the warnings library and suppressing all warnings to avoid clutter in the output
import warnings
warnings.filterwarnings('ignore')
```

```
# Loading the Iris dataset from a specified CSV file path into a DataFrame
data = pd.read_csv("C:/Users/ML/Desktop/123B2F148/Datasets/Iris.csv")
```

```
data.shape          #150 instances and 6 variables
```

```
(150, 6)
```

```python
# Displaying the first five rows of the DataFrame
data.head()
```

	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

```python
# Displaying a summary of the DataFrame, including data types and non-null counts
data.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
```

```python
# Checking for missing values in each column of the DataFrame by summing the null entries
data.isnull().sum()
```

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

```python
# Calculating the mean of the 'PetalLengthCm' column and printing the result
mean = data['PetalLengthCm'].mean()
print(mean)
```

```
3.758666666666666
```

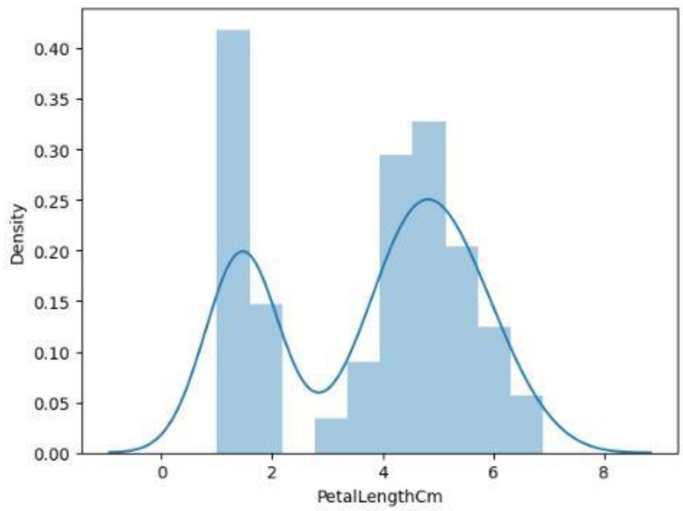
```python
# Calculating the median of the 'PetalLengthCm' column and printing the result
median = data['PetalLengthCm'].median()
print(median)
```

```
4.35
```

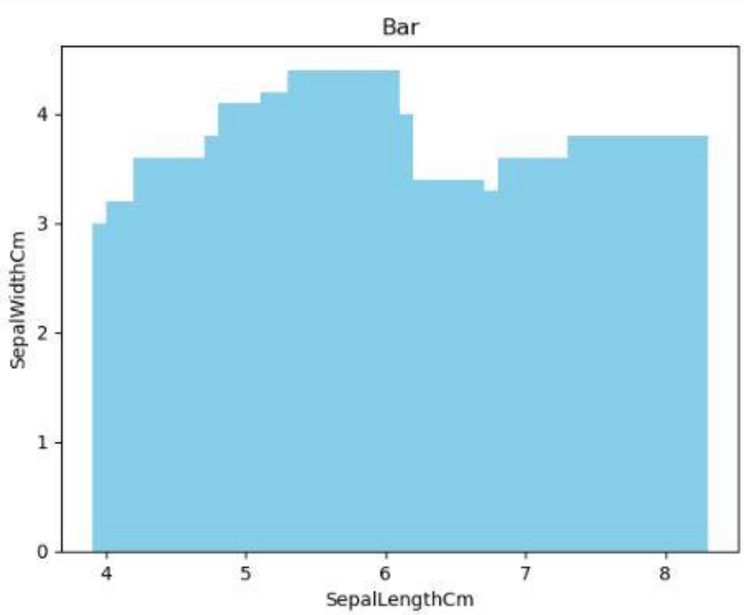
```python
# Calculating the mode of the 'PetalLengthCm' column and printing the result
mode = data['PetalLengthCm'].mode()
print(mode)
```

```
# Plotting the distribution of the 'PetalLengthCm' column using a histogram and kernel density estimate (KDE) with 10 bins
ans = data['PetalLengthCm']
ans = data['PetalLengthCm']
sns.distplot(ans,bins=10,hist=True,kde=True,label = 'PetalLengthCm')
```

```
<Axes: xlabel='PetalLengthCm', ylabel='Density'>
```

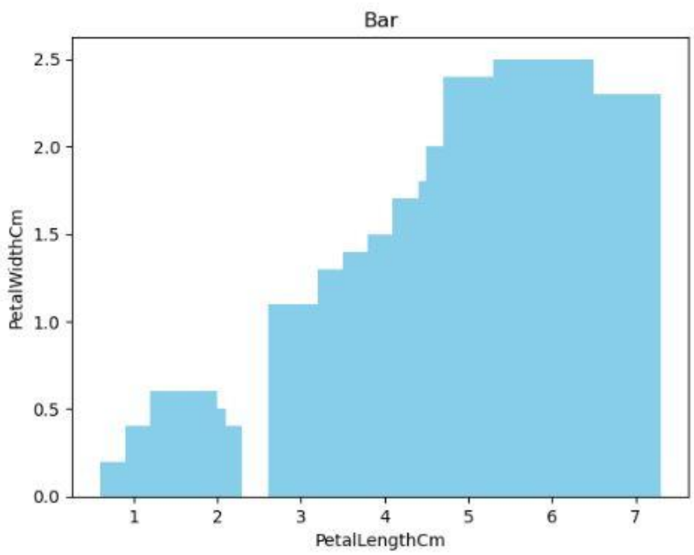



```
# Creating a bar chart with 'd1' on the x-axis and 'd2' on the y-axis, and customizing the appearance
plt.bar(d1, d2, color='skyblue')
plt.xlabel("SepalLengthCm")  # Labeling the x-axis
plt.ylabel("SepalWidthCm")    # Labeling the y-axis
plt.title("Bar")               # Setting the title of the chart
plt.show()                     # Displaying the bar chart
```

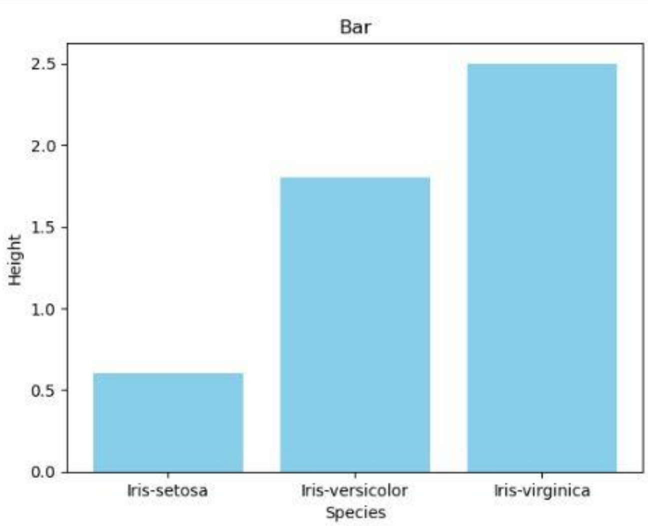


```python
d1 = data['PetalLengthCm']
d2 = data['PetalWidthCm']
```

```python
# Creating a bar chart with 'd1' representing Petal Length on the x-axis and 'd2' representing Petal Width on the y-axis
plt.bar(d1, d2, color='skyblue')
plt.xlabel("PetalLengthCm")  # Labeling the x-axis with 'PetalLengthCm'
plt.ylabel("PetalWidthCm")    # Labeling the y-axis with 'PetalWidthCm'
plt.title("Bar")               # Setting the title of the chart
plt.show()                     # Displaying the bar chart
```




```python
# Creating a bar chart with 'd1' representing species on the x-axis and 'd2' representing height on the y-axis
plt.bar(d1, d2, color='skyblue')
plt.xlabel("Species")  # Labeling the x-axis with 'Species'
plt.ylabel("Height")    # Labeling the y-axis with 'Height'
plt.title("Bar")        # Setting the title of the chart
plt.show()              # Displaying the bar chart
```



```python
# Finding the minimum value in the 'SepalLengthCm' column of the DataFrame
data['SepalLengthCm'].min()
```

```
4.3
```

```python
# Finding the maximum value in the 'SepalLengthCm' column of the DataFrame
data['SepalLengthCm'].max()
```

```
7.9
```

```python
# Calculating the range of the 'SepalLengthCm' column by subtracting the minimum value from the maximum value
data['SepalLengthCm'].max() - data['SepalLengthCm'].min()
```

```
3.6000000000000005
```

```python
# Calculating the variance of the 'SepalLengthCm' column in the DataFrame
data['SepalLengthCm'].var()
```

```
0.6856935123042505
```

```python
# Calculating the standard deviation of the 'SepalLengthCm' column in the DataFrame
data['SepalLengthCm'].std()
```

```
0.8280661279778629
```

```python
# Calculating the 75th percentile (Q3) of the 'SepalLengthCm' column in the DataFrame
Q2 = data['SepalLengthCm'].quantile(0.75)
```

```
6.4
```

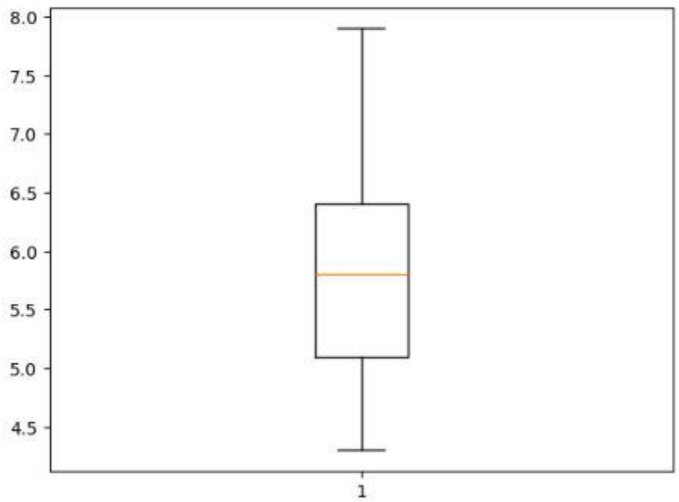
```python
# Calculating the 25th percentile (Q1) of the 'SepalLengthCm' column in the DataFrame
Q2 = data['SepalLengthCm'].quantile(0.25)
```

```
5.1
```

```python
# Calculating the 50th percentile (median) of the 'SepalLengthCm' column in the DataFrame
Q2 = data['SepalLengthCm'].quantile(0.5)
```

```
5.8
```

```
# Creating a box plot for the 'SepalLengthCm' column to visualize its distribution and identify potential outliers
plt.boxplot(data['SepalLengthCm'])
plt.show()  # Displaying the box plot
```




```
# Calculating the skewness of the 'SepalLengthCm' column to assess the asymmetry of its distribution
data['SepalLengthCm'].skew()
```

```
0.3149109566369728
```

References :

https://gist.github.com/pb111/512c840affb32593d28573fbb764045b

Conclusion :

In summary, descriptive statistics are vital for effectively summarizing and interpreting data, providing essential insights into central tendencies, variability, and distribution shapes. They facilitate the quick identification of key patterns and trends, aiding in informed decision-making. For a more comprehensive analysis, it’s beneficial to combine them with inferential statistics.