Assignment No. 3

Problem Statement: CORRELATION AND COVARIANCE

Objective: The objective of this assignment is to compute the correlation matrix of the Iris dataset, visualize the correlations among different features, and provide insights into the relationships between the dataset variables.

Prerequisite :

- A Python environment set up with libraries such as pandas, seaborn, and matplotlib for data manipulation and visualization.
- Basic knowledge of data analysis and visualization concepts.
- Familiarity with the Iris dataset and its features.

Theory :

Covariance : Covariance is a statistical measure that indicates the extent to which two random variables change together. It quantifies the relationship between the deviations of two variables from their respective means.

- It quantifies how the deviation of one variable (X) from its mean relates to the deviation of another variable (Y) from its mean.
- Covariance can range from $-\infty$ infinity to $+\infty$

Types of Covariance:

- **Positive Covariance:** Variables move in the same direction (e.g., both increase).
- **Negative Covariance:** Variables move in opposite directions (e.g., one increases while the other decreases).

Formula:

$$Cov_{x,y} = \frac{\Sigma(x_i - \overline{x})(y_i - \overline{y})}{N-1}$$

- xi, yi: individual data points
- $\bar{x}$ $\bar{y}$ means of variables
- N: number of data points

Correlation : Correlation measures the strength and direction of the linear relationship between two variables. It quantifies how changes in one variable are associated with changes in another.

- Correlation helps us to determine whether or not, and how strongly, changes in various variables relate to each other.
- Correlation coefficients range from -1 to +1

Types of Correlation:

- **Positive Correlation:** Indicates that as one variable increases, the other also increases (e.g., height and weight).
- **Negative Correlation:** Indicates that as one variable increases, the other decreases (e.g., exercise and body fat).

Formula :

$$r = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

where:

- r: correlation coefficient
- Cov(X,Y) : covariance between variables X and Y
- σX, σY: standard deviations of X and Y

Correlation Matrix :

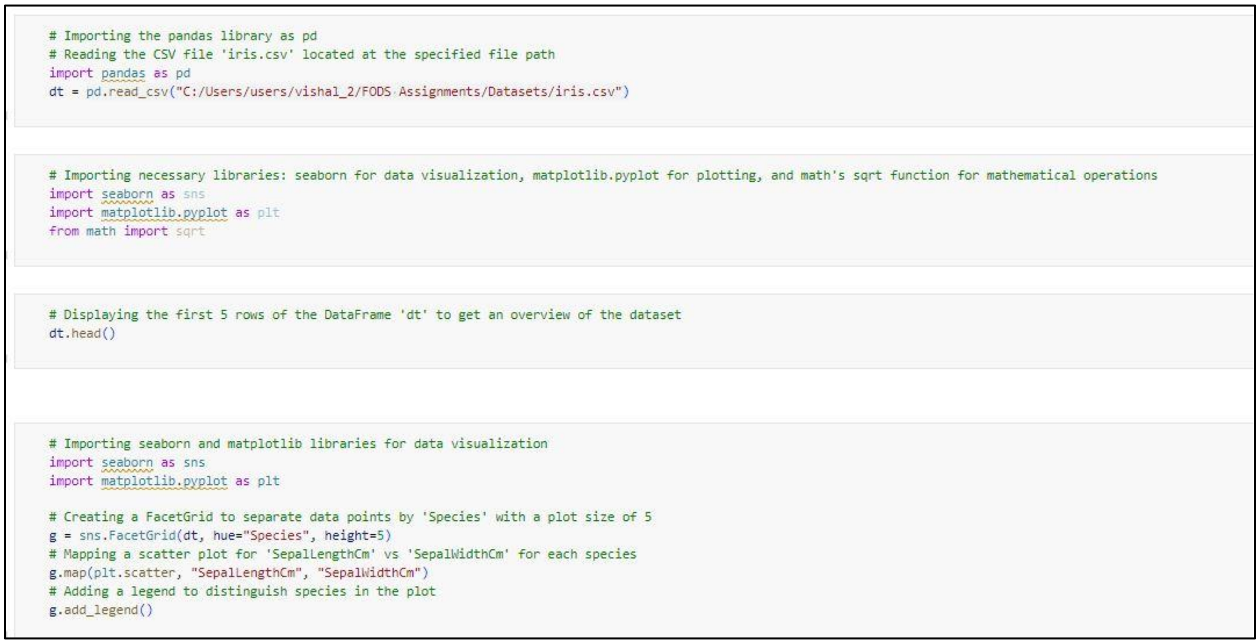
A correlation matrix is a table that displays the correlation coefficients between multiple variables in a dataset. It shows how strongly pairs of variables are related to one another.

Techniques for Measuring Correlation :

- **Scatter Diagram:** A graphical tool that displays the relationship between two variables by plotting data points, helping to visually assess the relationship's form.
- **Karl Pearson's Coefficient of Correlation:** A numerical measure that quantifies the degree of linear relationship between two variables X and Y. It is suitable only for linear correlations.

- **Spearman's Rank Correlation:** This method evaluates the correlation between variables when the relationship is clear in direction but may not be linear, using ranks instead of actual values.

Code & output :



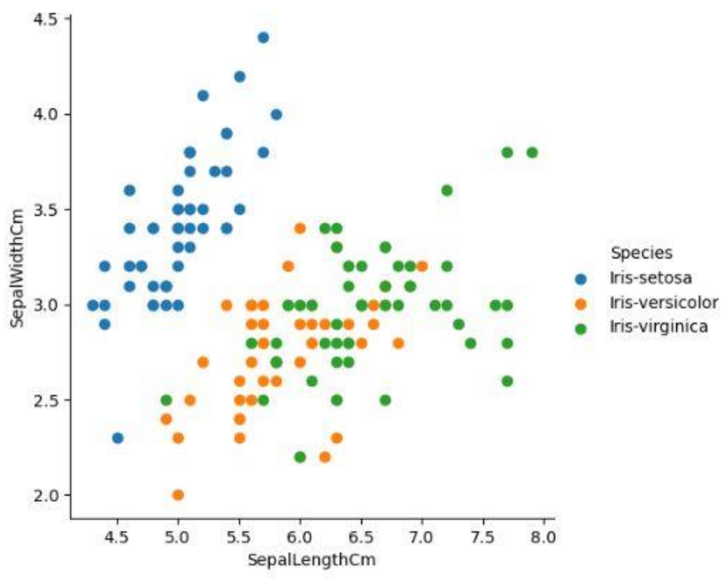
```
# Importing the pandas library as pd
# Reading the CSV file 'iris.csv' located at the specified file path
import pandas as pd
dt = pd.read_csv("C:/Users/users/vishal_2/FODS Assignments/Datasets/iris.csv")
```

```
# Importing necessary libraries: seaborn for data visualization, matplotlib.pyplot for plotting, and math's sqrt function for mathematical operations
import seaborn as sns
import matplotlib.pyplot as plt
from math import sqrt
```

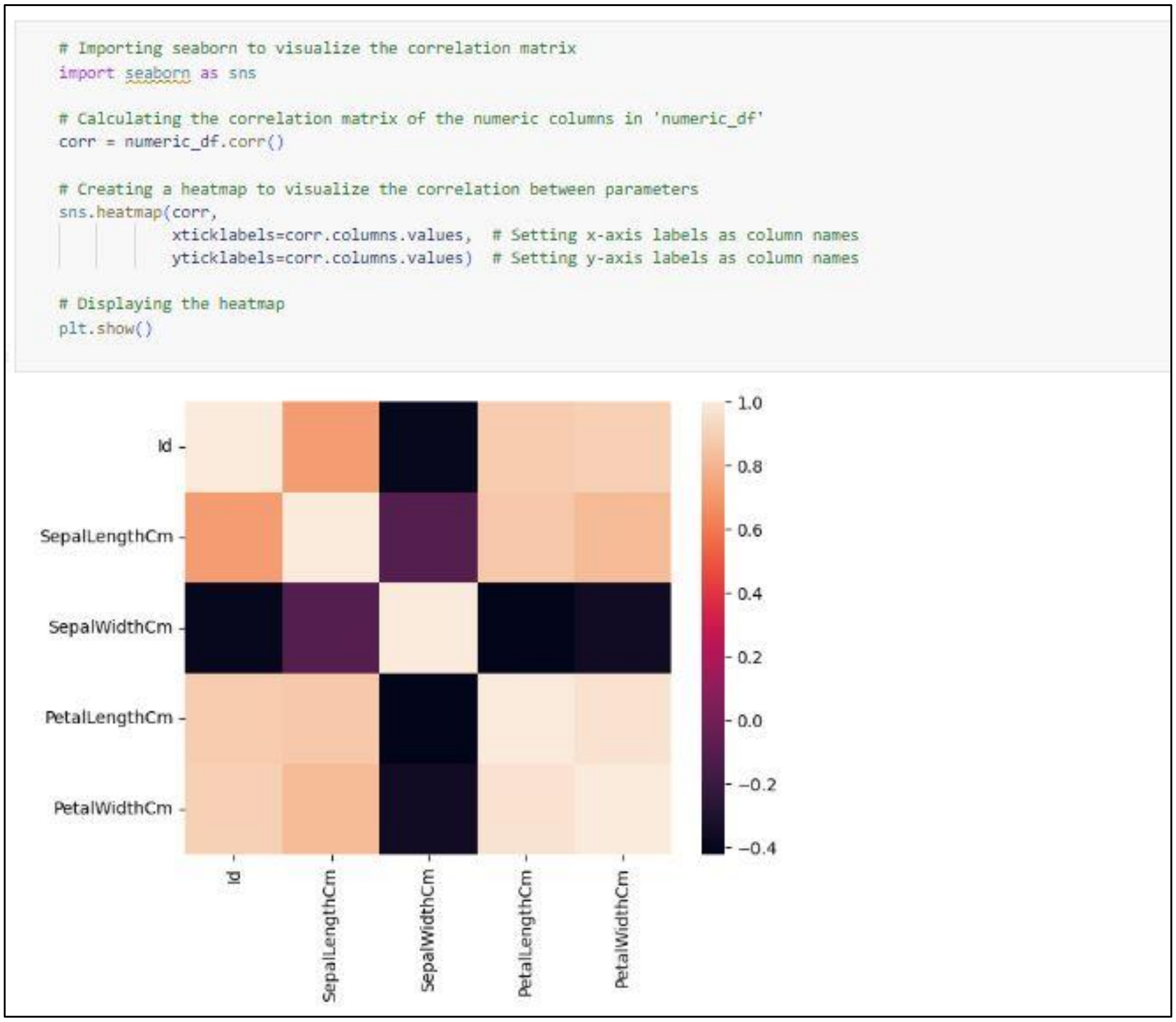
```
# Displaying the first 5 rows of the DataFrame 'dt' to get an overview of the dataset
dt.head()
```

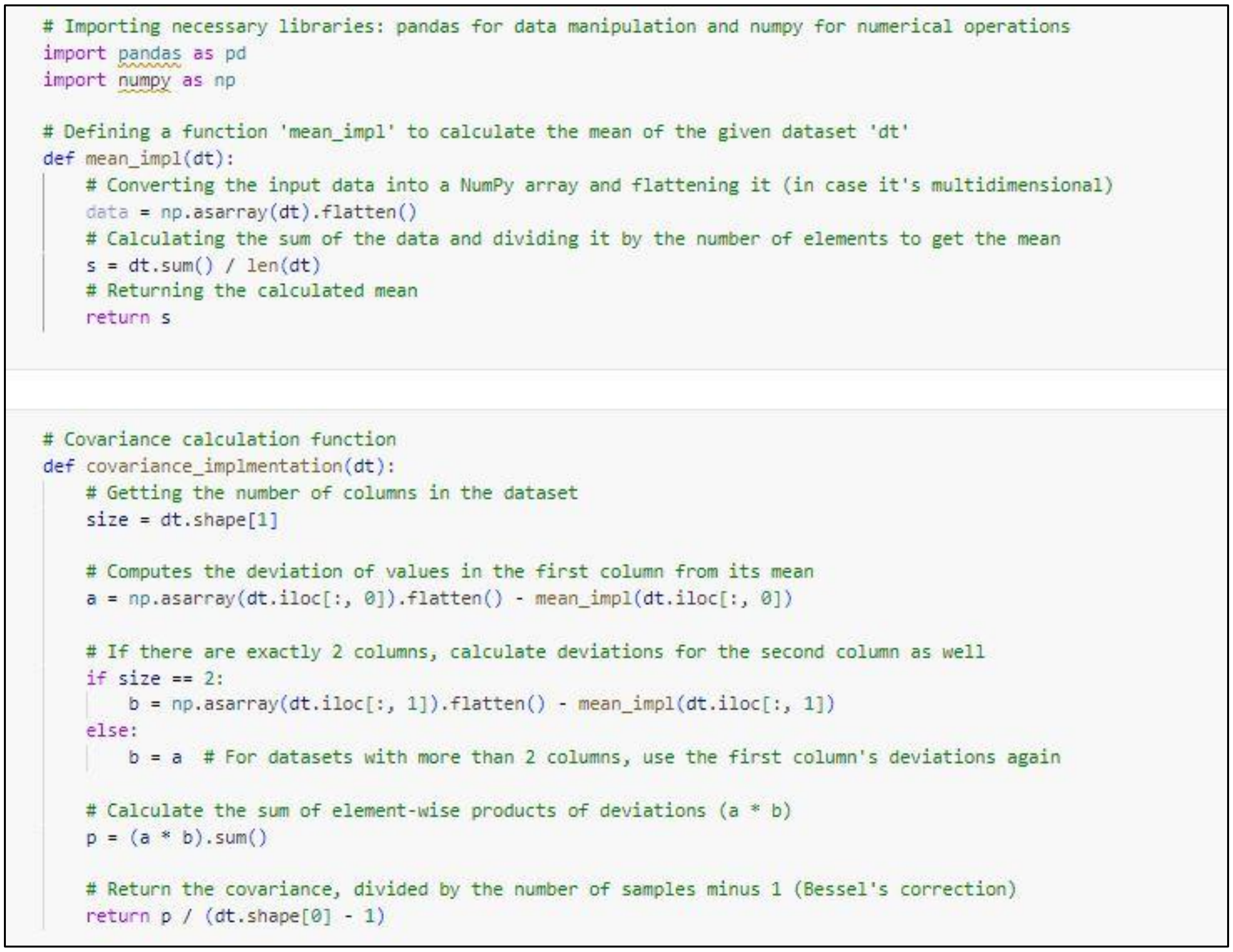
```
# Importing seaborn and matplotlib libraries for data visualization
import seaborn as sns
import matplotlib.pyplot as plt

# Creating a FacetGrid to separate data points by 'Species' with a plot size of 5
g = sns.FacetGrid(dt, hue="Species", height=5)
# Mapping a scatter plot for 'SepalLengthCm' vs 'SepalWidthCm' for each species
g.map(plt.scatter, "SepalLengthCm", "SepalWidthCm")
# Adding a legend to distinguish species in the plot
g.add_legend()
```

```
<seaborn.axisgrid.FacetGrid at 0x26a28fda5d0>
```



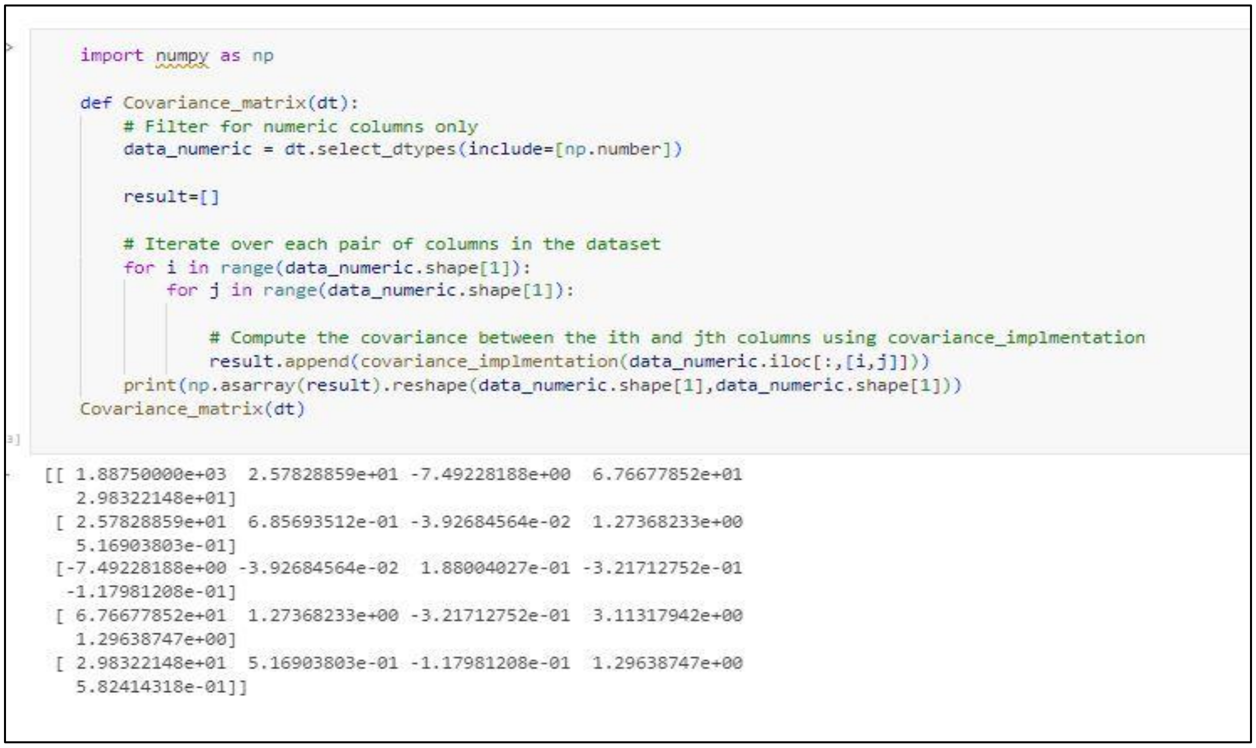
```
# Selecting only the columns with numeric data types (float and int) from the DataFrame 'dt' and storing them in 'numeric_df'
numeric_df = dt.select_dtypes(include = [float, int])
```


```
# Importing seaborn to visualize the correlation matrix
import seaborn as sns

# Calculating the correlation matrix of the numeric columns in 'numeric_df'
corr = numeric_df.corr()

# Creating a heatmap to visualize the correlation between parameters
sns.heatmap(corr,
            xticklabels=corr.columns.values,  # Setting x-axis labels as column names
            yticklabels=corr.columns.values)  # Setting y-axis labels as column names

# Displaying the heatmap
plt.show()
```



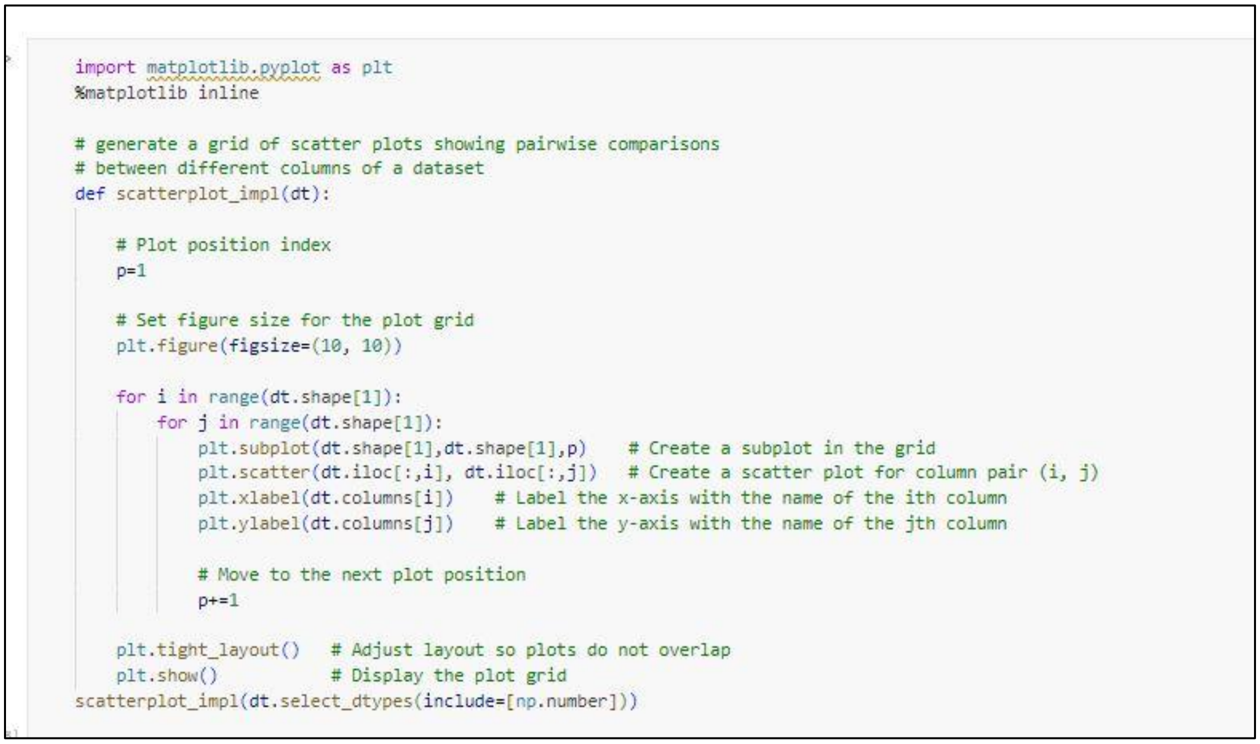
```python
# Importing necessary libraries: pandas for data manipulation and numpy for numerical operations
import pandas as pd
import numpy as np

# Defining a function 'mean_impl' to calculate the mean of the given dataset 'dt'
def mean_impl(dt):
    # Converting the input data into a NumPy array and flattening it (in case it's multidimensional)
    data = np.asarray(dt).flatten()
    # Calculating the sum of the data and dividing it by the number of elements to get the mean
    s = dt.sum() / len(dt)
    # Returning the calculated mean
    return s
```

```python
# Covariance calculation function
def covariance_implmentation(dt):
    # Getting the number of columns in the dataset
    size = dt.shape[1]

    # Computes the deviation of values in the first column from its mean
    a = np.asarray(dt.iloc[:, 0]).flatten() - mean_impl(dt.iloc[:, 0])

    # If there are exactly 2 columns, calculate deviations for the second column as well
    if size == 2:
        b = np.asarray(dt.iloc[:, 1]).flatten() - mean_impl(dt.iloc[:, 1])
    else:
        b = a  # For datasets with more than 2 columns, use the first column's deviations again

    # Calculate the sum of element-wise products of deviations (a * b)
    p = (a * b).sum()

    # Return the covariance, divided by the number of samples minus 1 (Bessel's correction)
    return p / (dt.shape[0] - 1)
```

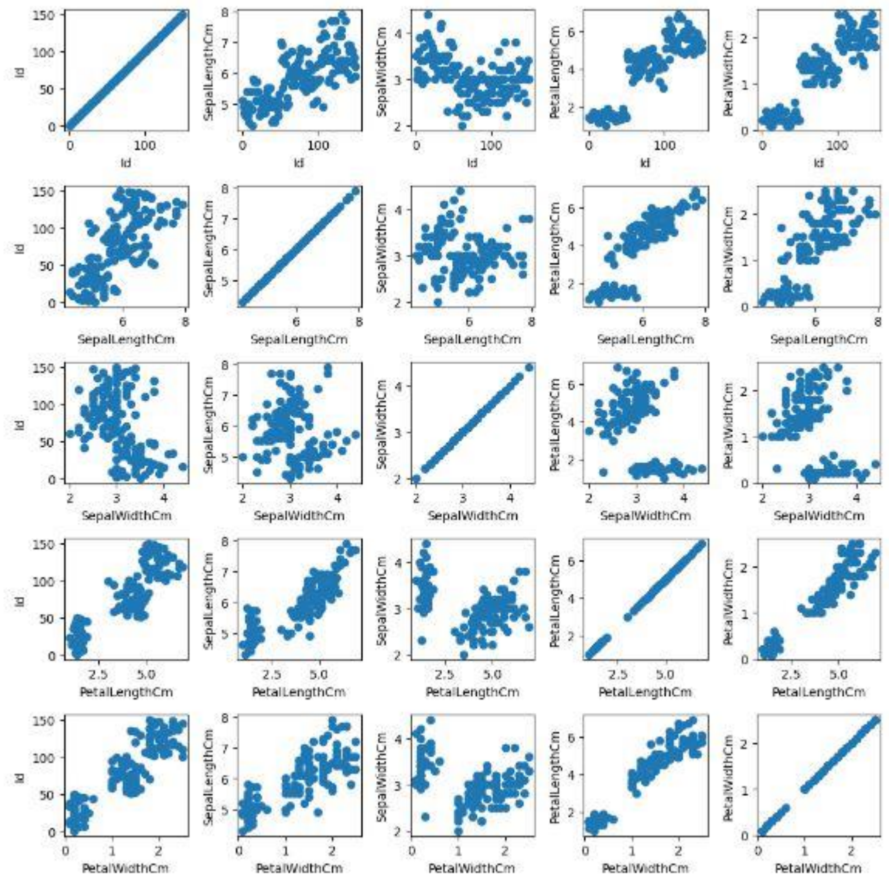


```python
import numpy as np

def Covariance_matrix(dt):
    # Filter for numeric columns only
    data_numeric = dt.select_dtypes(include=[np.number])

    result=[]

    # Iterate over each pair of columns in the dataset
    for i in range(data_numeric.shape[1]):
        for j in range(data_numeric.shape[1]):

            # Compute the covariance between the ith and jth columns using covariance_implmentation
            result.append(covariance_implmentation(data_numeric.iloc[:,[i,j]]))
    print(np.asarray(result).reshape(data_numeric.shape[1],data_numeric.shape[1]))
Covariance_matrix(dt)
```

```
[[ 1.88750000e+03  2.57828859e+01 -7.49228188e+00  6.76677852e+01
   2.98322148e+01]
 [ 2.57828859e+01  6.85693512e-01 -3.92684564e-02  1.27368233e+00
   5.16903803e-01]
 [-7.49228188e+00 -3.92684564e-02  1.88004027e-01 -3.21712752e-01
  -1.17981208e-01]
 [ 6.76677852e+01  1.27368233e+00 -3.21712752e-01  3.11317942e+00
   1.29638747e+00]
 [ 2.98322148e+01  5.16903803e-01 -1.17981208e-01  1.29638747e+00
   5.82414318e-01]]
```



```python
import matplotlib.pyplot as plt
%matplotlib inline

# generate a grid of scatter plots showing pairwise comparisons
# between different columns of a dataset
def scatterplot_impl(dt):

    # Plot position index
    p=1

    # Set figure size for the plot grid
    plt.figure(figsize=(10, 10))

    for i in range(dt.shape[1]):
        for j in range(dt.shape[1]):
            plt.subplot(dt.shape[1],dt.shape[1],p)    # Create a subplot in the grid
            plt.scatter(dt.iloc[:,i], dt.iloc[:,j])   # Create a scatter plot for column pair (i, j)
            plt.xlabel(dt.columns[i])    # Label the x-axis with the name of the ith column
            plt.ylabel(dt.columns[j])    # Label the y-axis with the name of the jth column

            # Move to the next plot position
            p+=1

    plt.tight_layout()   # Adjust layout so plots do not overlap
    plt.show()           # Display the plot grid
scatterplot_impl(dt.select_dtypes(include=[np.number]))
```



References :

- https://www.kaggle.com/code/tashisam07/covariance-and-correlation
- https://www.kaggle.com/code/ajay101tiwari/covariance-and-correlation-python-implementation
- https://colab.research.google.com/drive/1vLl9UjTLVbiJmz29tQ6uYiQ
- https://www.cuemath.com/algebra/covariance-matrix/

Conclusion :

In this assignment, we examined the correlation matrix of the Iris dataset, which illuminates the relationships among various features. By visualizing these correlations with a heatmap, we gain clearer insights that can inform subsequent analyses or modeling strategies. This essential groundwork lays the foundation for more sophisticated statistical techniques and machine learning applications.