Student ID: 21127690

Name: Ngô Nguyễn Thanh Thanh

Report: Canonical Correlation Analysis

I. Evaluation summary:

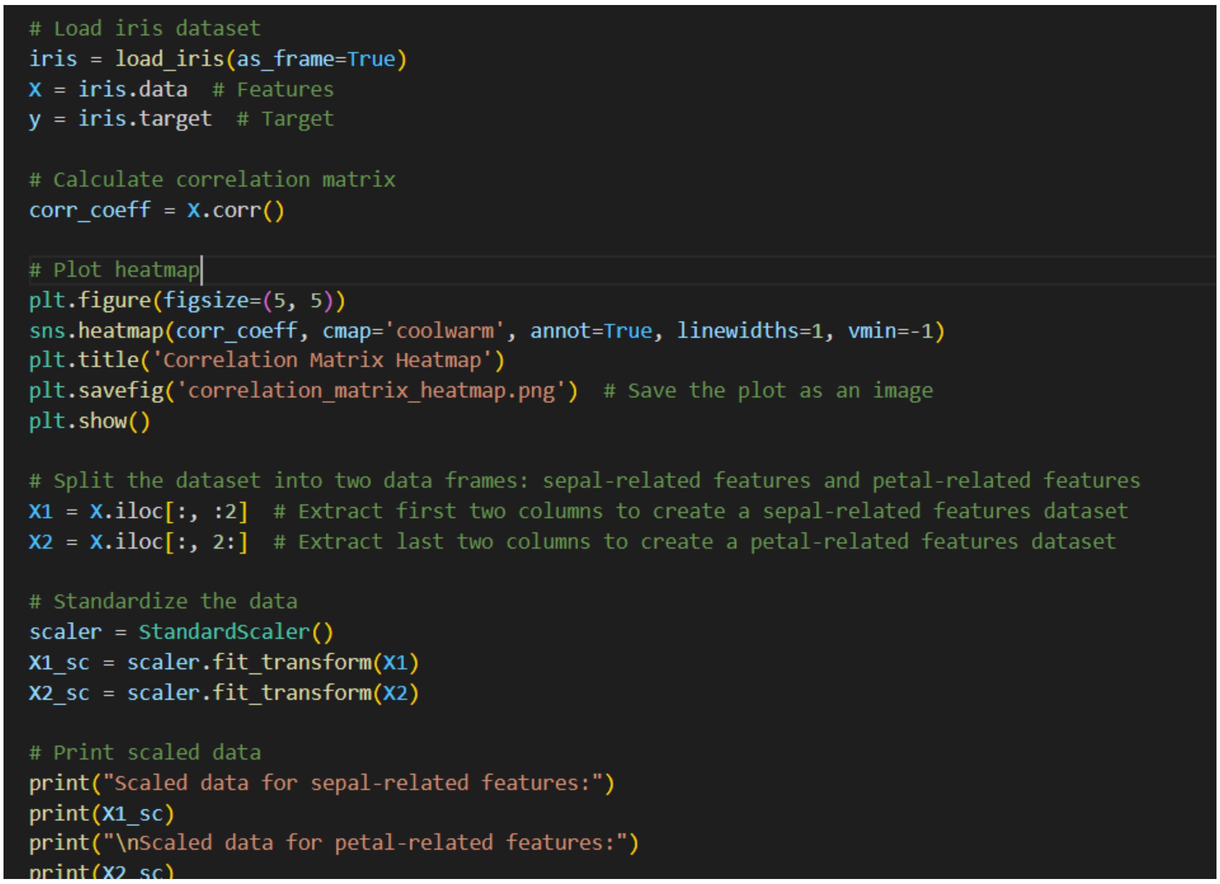
Task		Requirement Met(%)	Notes
Implementation	Process Data	100%	
	Analyze the correlation between the features of this dataset	100%	
	Run CCA using statsmodels, rcca package and skbio	100%	
	Check the dependency between canonical variates by correlating canonical variate pairs	100%	
	Analyzing analyzing the loadings associated with each of our canonical variates	100%	
	Analyzing CCA coefficients	100%	
	Visualize the results	100%	
Requisitions		100%	
Research questions		100%	
Total:		100%	

II. List of funtion:

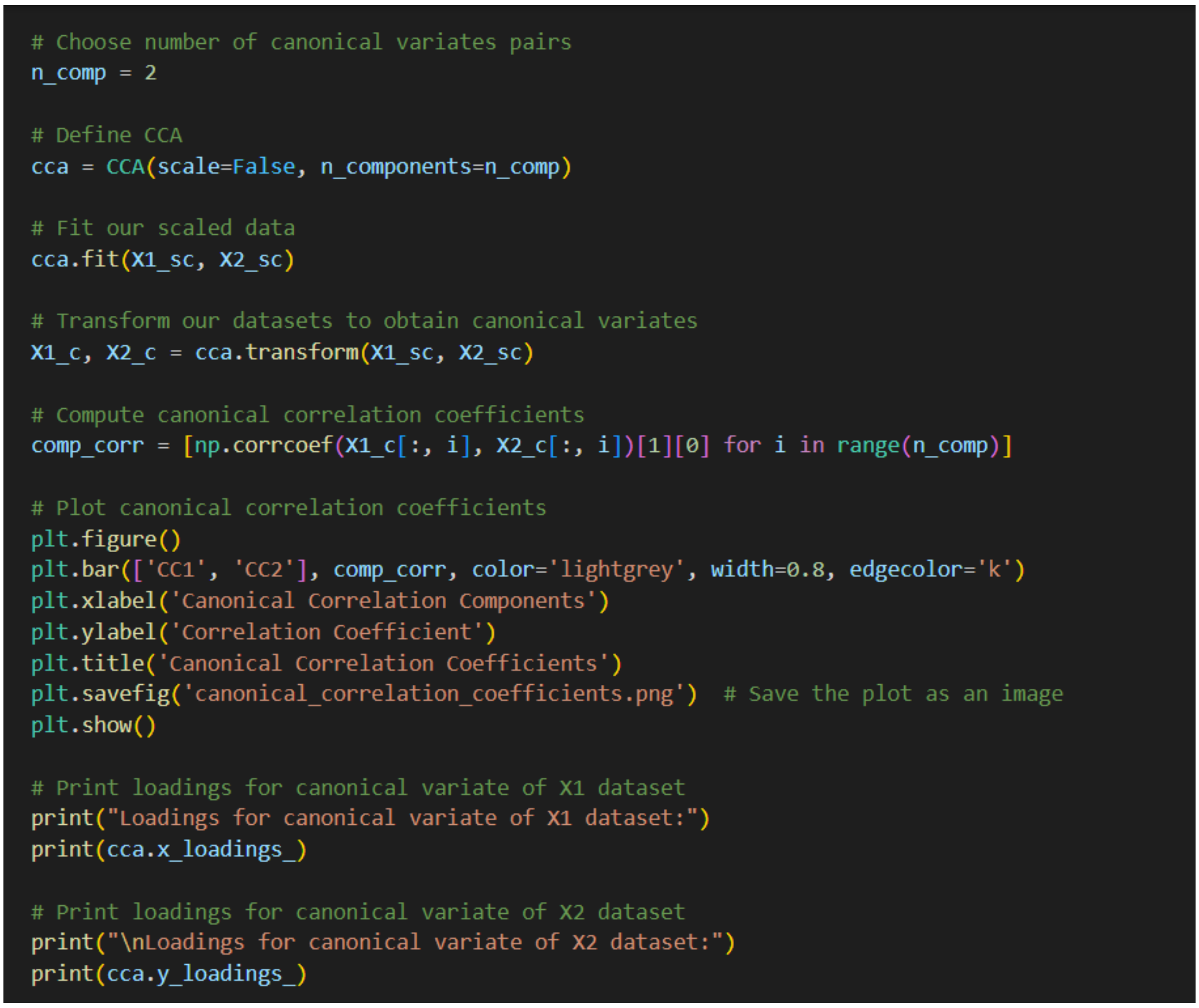
A list of function and image proofs:

1. **pd.DataFrame()**: Creates a pandas DataFrame.
2. **plt.figure()**: Creates a new figure for plotting.
3. **sns.heatmap()**: Plots a heatmap using seaborn.
4. **plt.title()**: Sets the title of the plot.

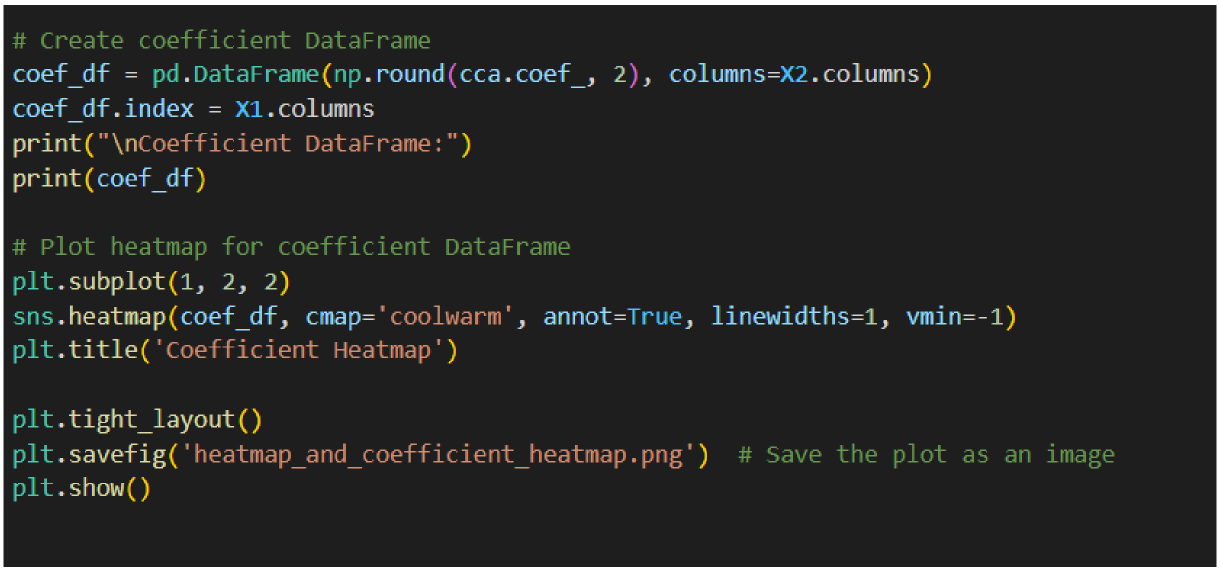
5. **plt.savefig()**: Saves the current figure to a file.
6. **plt.show()**: Displays the current figure.
7. **StandardScaler()**: Standardizes features by removing the mean and scaling to unit variance.
8. **CCA()**: Creates a Canonical Correlation Analysis model object from scikit-learn.
9. **fit_transform()**: Fits the scaler to the data and transforms it in a single step.
10. **fit()**: Fits the CCA model to the data.
11. **transform()**: Transforms the data using the fitted CCA model.
12. **np.corrcoef()**: Computes the correlation coefficient matrix.
13. **plt.bar()**: Plots a bar chart.
14. **plt.xlabel()**: Sets the label for the x-axis.
15. **plt.ylabel()**: Sets the label for the y-axis.
16. **plt.subplot()**: Adds a subplot to the current figure.
17. **np.round()**: Rounds the elements of an array to the nearest integer.
18. **pd.DataFrame.index**: Sets the index (row labels) of the DataFrame.
19. **pd.DataFrame.columns**: Sets the column labels of the DataFrame.



```python
# Load iris dataset
iris = load_iris(as_frame=True)
X = iris.data  # Features
y = iris.target  # Target

# Calculate correlation matrix
corr_coeff = X.corr()

# Plot heatmap
plt.figure(figsize=(5, 5))
sns.heatmap(corr_coeff, cmap='coolwarm', annot=True, linewidths=1, vmin=-1)
plt.title('Correlation Matrix Heatmap')
plt.savefig('correlation_matrix_heatmap.png')  # Save the plot as an image
plt.show()

# Split the dataset into two data frames: sepal-related features and petal-related features
X1 = X.iloc[:, :2]  # Extract first two columns to create a sepal-related features dataset
X2 = X.iloc[:, 2:]  # Extract last two columns to create a petal-related features dataset

# Standardize the data
scaler = StandardScaler()
X1_sc = scaler.fit_transform(X1)
X2_sc = scaler.fit_transform(X2)

# Print scaled data
print("Scaled data for sepal-related features:")
print(X1_sc)
print("\nScaled data for petal-related features:")
print(X2_sc)
```



```python
# Choose number of canonical variates pairs
n_comp = 2

# Define CCA
cca = CCA(scale=False, n_components=n_comp)

# Fit our scaled data
cca.fit(X1_sc, X2_sc)

# Transform our datasets to obtain canonical variates
X1_c, X2_c = cca.transform(X1_sc, X2_sc)

# Compute canonical correlation coefficients
comp_corr = [np.corrcoef(X1_c[:, i], X2_c[:, i])[1][0] for i in range(n_comp)]

# Plot canonical correlation coefficients
plt.figure()
plt.bar(['CC1', 'CC2'], comp_corr, color='lightgrey', width=0.8, edgecolor='k')
plt.xlabel('Canonical Correlation Components')
plt.ylabel('Correlation Coefficient')
plt.title('Canonical Correlation Coefficients')
plt.savefig('canonical_correlation_coefficients.png')  # Save the plot as an image
plt.show()

# Print loadings for canonical variate of X1 dataset
print("Loadings for canonical variate of X1 dataset:")
print(cca.x_loadings_)

# Print loadings for canonical variate of X2 dataset
print("\nLoadings for canonical variate of X2 dataset:")
print(cca.y_loadings_)
```



```python
# Create coefficient DataFrame
coef_df = pd.DataFrame(np.round(cca.coef_, 2), columns=X2.columns)
coef_df.index = X1.columns
print("\nCoefficient DataFrame:")
print(coef_df)

# Plot heatmap for coefficient DataFrame
plt.subplot(1, 2, 2)
sns.heatmap(coef_df, cmap='coolwarm', annot=True, linewidths=1, vmin=-1)
plt.title('Coefficient Heatmap')

plt.tight_layout()
plt.savefig('heatmap_and_coefficient_heatmap.png')  # Save the plot as an image
plt.show()
```

III. Function Summaries and Implementation

Below is a summary of the usage and implementation of the functions used in the provided code:

1. **pd.DataFrame()**: This function is used to create a pandas DataFrame, which is a two-dimensional labeled data structure with columns of potentially different types.

Implementation: You can pass data, index (row labels), and columns (column labels) as arguments to create a DataFrame. For example:

2. **plt.figure()**: This function creates a new figure for plotting.

Implementation: You can call this function without any arguments to create a new figure. Optionally, you can specify parameters like figsize to control the size of the figure.

3. **sns.heatmap()**: This function is used to plot a heatmap using seaborn, which visualizes a matrix of data as a color-encoded grid.

Implementation: You can pass a 2D array (matrix) of data to plot as a heatmap. Additional parameters like cmap (colormap), annot (annotation), linewidths, and vmin (minimum value for colormap) can be specified.

4. **plt.title()**: Sets the title of the plot.

Implementation: Call this function with a string argument to set the title of the current plot.

5. **plt.savefig()**: Saves the current figure to a file.

Implementation: Call this function with the filename (including extension) as the argument to save the current plot as an image file.

6. **plt.show()**: Displays the current figure.

Implementation: Call this function to display the current plot in the output.

7. **StandardScaler()**: This function creates a StandardScaler object, which is used to standardize features by removing the mean and scaling to unit variance.

Implementation: Create a StandardScaler object. You can then call fit_transform() to fit the scaler to your data and transform it in a single step.

8. **CCA()**: This function creates a Canonical Correlation Analysis model object from scikit-learn.

Implementation: Create a CCA object. You can specify parameters like scale (whether to scale the data) and n_components (number of canonical variate pairs).

9. **fit_transform()**: This method fits the scaler to the data and transforms it in a single step.

Implementation: Call this method on a scaler object with your data as the argument. It returns the standardized data.

10. **fit()**: This method fits the CCA model to the data.

Implementation: Call this method on a CCA object with the two sets of standardized data (X1_sc and X2_sc) as arguments.

11. **transform()**: This method transforms the data using the fitted CCA model.

Implementation: Call this method on a CCA object with the two sets of standardized data as arguments. It returns the transformed canonical variates.

12. **np.corrcoef()**: This function computes the correlation coefficient matrix.

Implementation: Pass arrays representing two sets of variables to this function to compute the correlation coefficient matrix between them. It returns a matrix of correlation coefficients.

13. **plt.bar()**: This function is used to plot a bar chart.

Implementation: Pass the x-axis values, y-axis values, and optionally other parameters like color and width to create a bar chart.

14. **plt.xlabel()**: Sets the label for the x-axis.

Implementation: Call this function with a string argument to set the label for the x-axis.

15. **plt.ylabel()**: Sets the label for the y-axis.

Implementation: Call this function with a string argument to set the label for the y-axis.

16. **plt.subplot()**: Adds a subplot to the current figure.

Implementation: Call this function with the number of rows, number of columns, and subplot number as arguments to add a subplot to the current figure.

17. **np.round()**: This function rounds the elements of an array to the nearest integer.

Implementation: Pass an array and the desired number of decimals to round to as arguments to round the elements of the array.

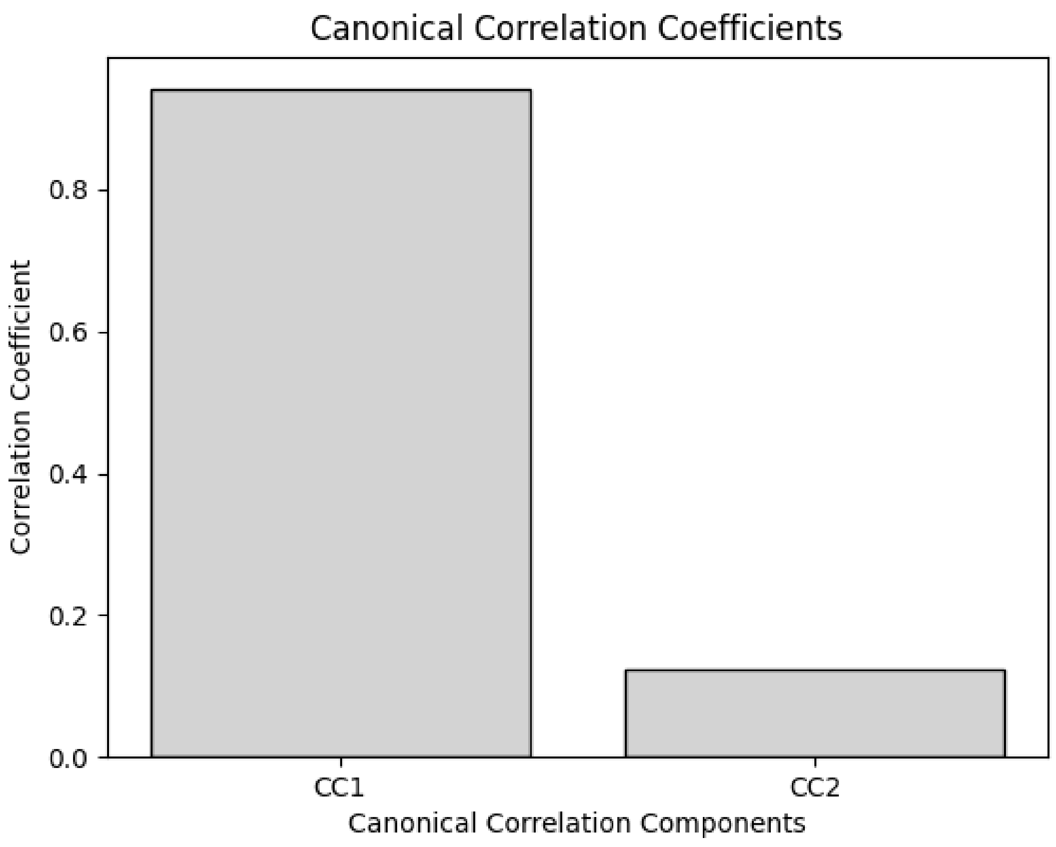
18. **pd.DataFrame.index**: This attribute sets the index (row labels) of the DataFrame.

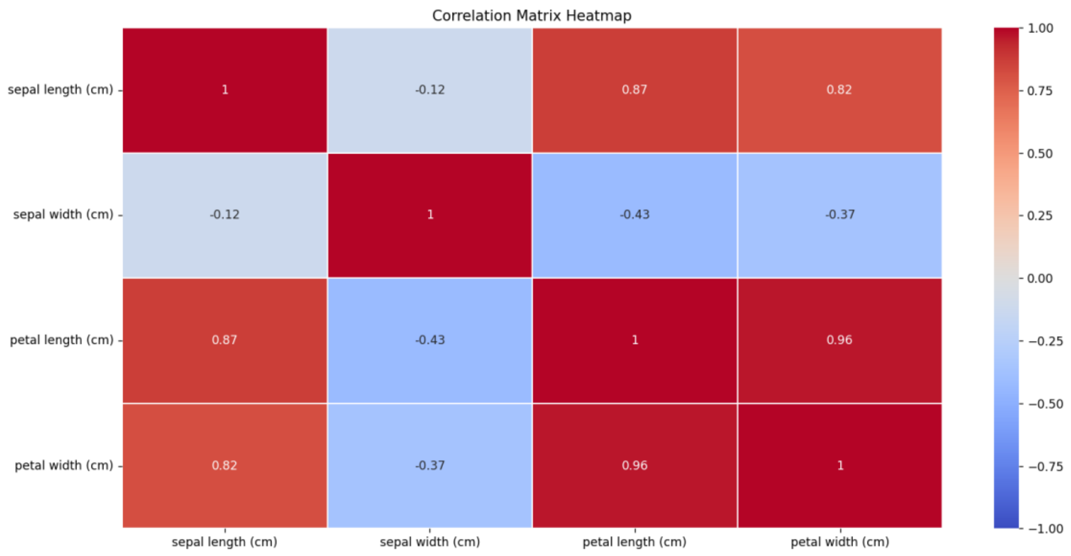
Implementation: Set this attribute to an array or list to define the row labels of the DataFrame.

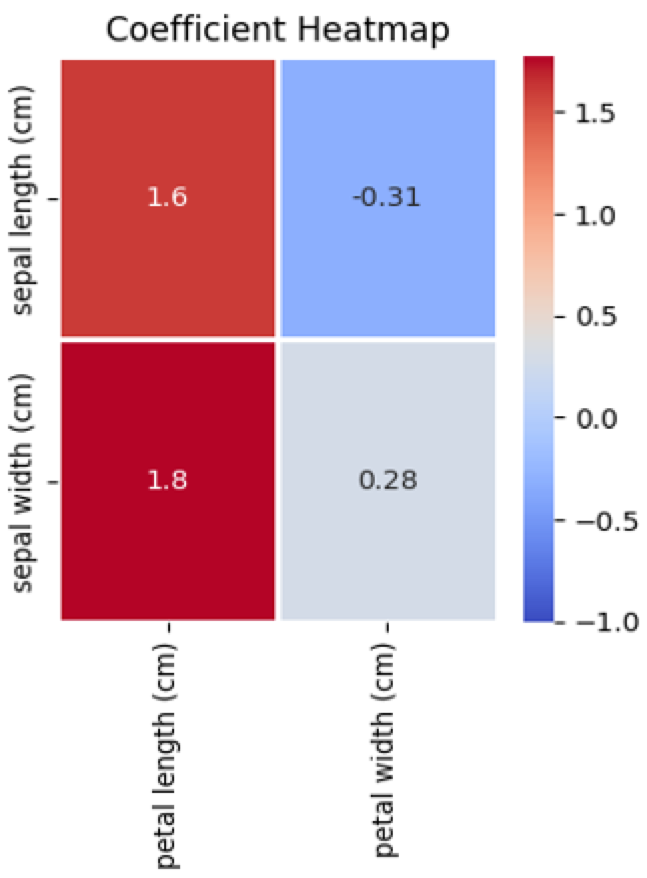
19. **pd.DataFrame.columns**: This attribute sets the column labels of the DataFrame.

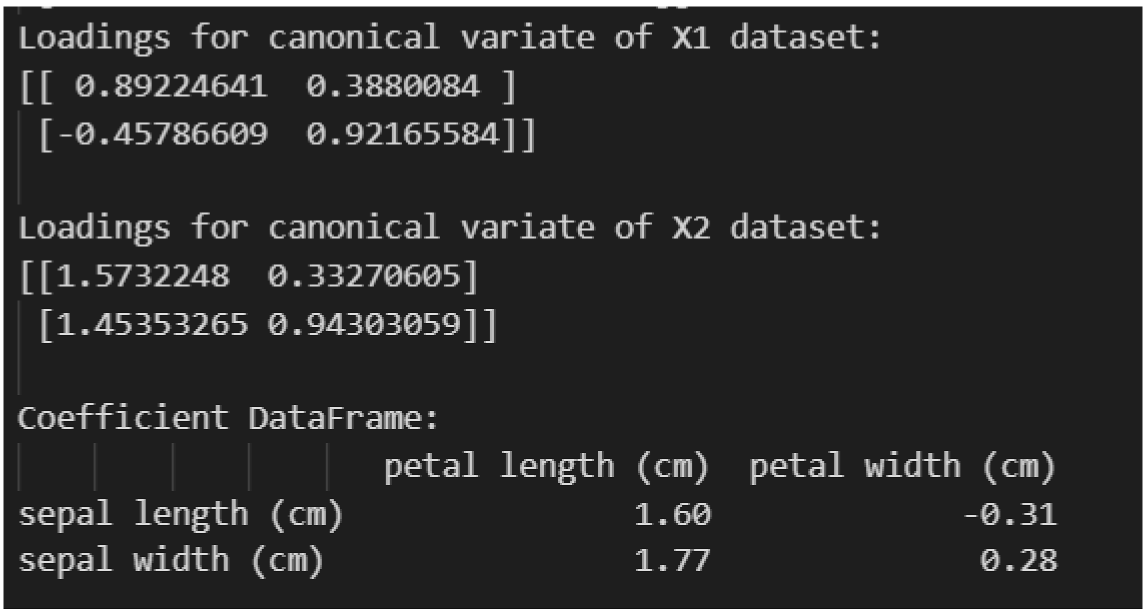
Implementation: Set this attribute to an array or list to define the column labels of the DataFrame.

IV. The results:





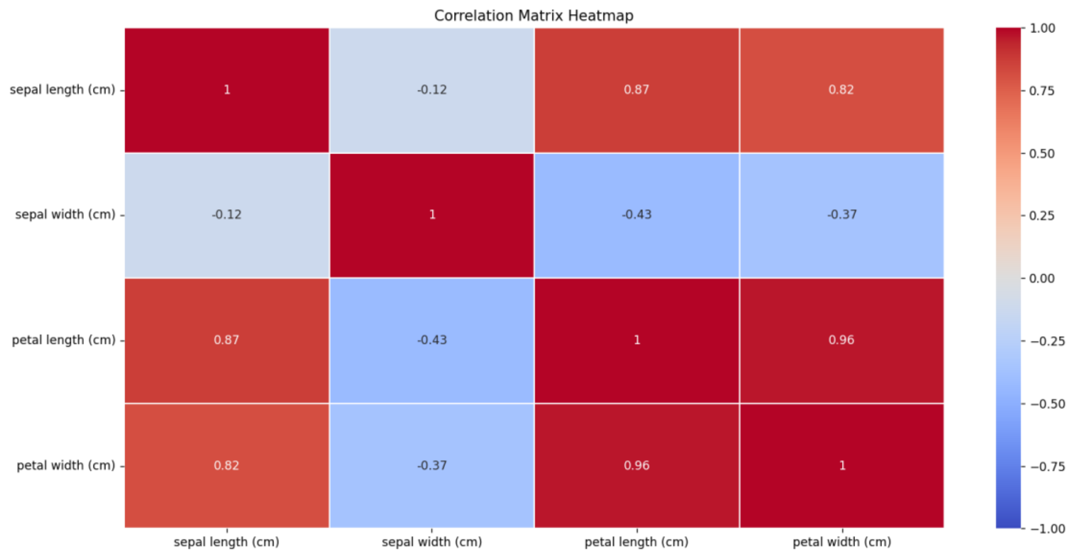




```
Loadings for canonical variate of X1 dataset:
[[ 0.89224641  0.3880084 ]
 [-0.45786609  0.92165584]]

Loadings for canonical variate of X2 dataset:
[[1.5732248  0.33270605]
 [1.45353265 0.94303059]]

Coefficient DataFrame:
                   petal length (cm)  petal width (cm)
sepal length (cm)               1.60             -0.31
sepal width (cm)                1.77              0.28
```

V. Requisitions

Request 01: Students explain differences of correlation between sepal width / sepal length with petal related value



From the above graph, we can see that:

- Petal width and petal length have high positive correlations.
- Petal length and sepal width have low negative correlations.
- Petal Width and Sepal length have high correlations.
- Petal length and sepal length have high positive correlations
- Petal width and sepal width have low positive correlations
- Sepal length and sepal width have low negative correlation

The correlation between sepal width/sepal length and petal-related values in the Iris dataset can provide insights into the relationships between these different features. Below is the explanation of the differences in correlation:

Sepal Width and Petal-related Value:

- Sepal width typically has a lower correlation with petal-related values compared to sepal length. This is because the width of the sepal may not directly affect the size or dimensions of the petals in the same way that sepal length does.
- A lower correlation between sepal width and petal-related values implies that changes in sepal width may not consistently coincide with changes in petal length or width across different samples. Therefore, sepal width may not be as indicative of petal characteristics.

Sepal Length and Petal-related Value:

- Sepal length tends to have a higher correlation with petal-related values compared to sepal width. This is because the length of the sepal can more directly influence the overall size and dimensions of the flower, including the petals.

- A higher correlation between sepal length and petal-related values suggests that variations in sepal length are more likely to be associated with variations in petal length or width across different samples. Therefore, sepal length may provide more valuable information about petal characteristics.

In summary, the differences in correlation between sepal width/sepal length and petal-related values reflect the varying degrees of influence that these sepal characteristics have on the size and dimensions of the petals. Sepal length tends to have a stronger correlation with petal-related values compared to sepal width, indicating that it may be a more informative feature when analyzing the relationship between sepals and petals in the Iris dataset.

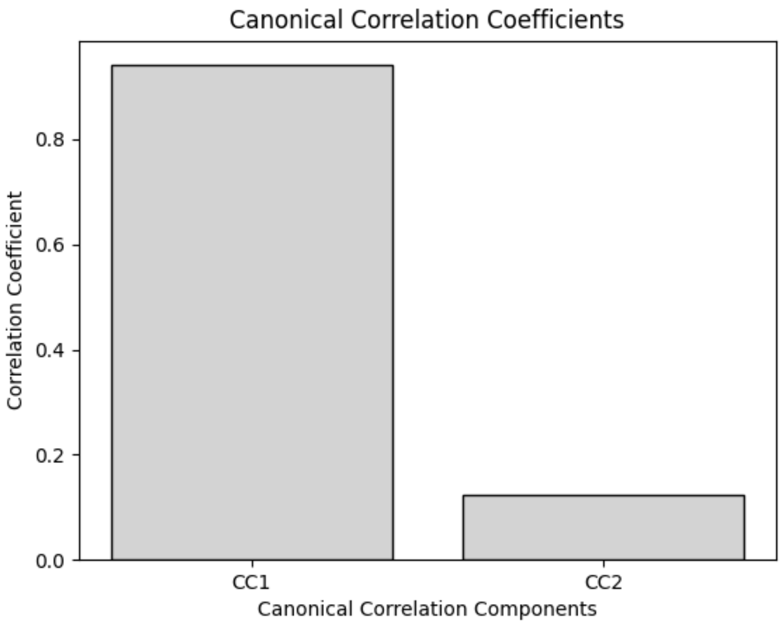
Request 02 : Students explain the important of scaling data

Scaling data is essential in many machine learning algorithms, including Canonical Correlation Analysis (CCA), for several reasons:

1. **Equalizing Variable Magnitudes**: In datasets like the Iris dataset, where features have different units and scales (e.g., centimeters for sepal length and millimeters for petal width), variables with larger magnitudes can dominate the analysis. Scaling transforms the data so that each feature contributes equally to the analysis, preventing bias towards variables with larger scales.
2. **Improving Convergence and Performance**: Many machine learning algorithms, including CCA, use optimization techniques that converge faster and perform better when the features are scaled. Scaling can improve the numerical stability of the optimization process, helping algorithms converge to the optimal solution more efficiently.
3. **Enhancing Interpretability**: Scaling ensures that the coefficients or loadings obtained from the analysis are directly comparable across features. Without scaling, the coefficients may be influenced by the scale of the features, making it challenging to interpret their relative importance or contribution to the analysis accurately.
4. **Regularization and Penalty-Based Methods**: Algorithms that use regularization or penalty-based methods, such as Ridge Regression or LASSO, are sensitive to the scale of the features. Scaling the data ensures that the regularization penalties are applied uniformly across all features, preventing any particular feature from dominating the regularization process.
5. **Distance-Based Algorithms**: Scaling is particularly crucial for distance-based algorithms, where the computation of distances between data points (e.g., in clustering or classification algorithms) is involved. Without scaling, features with larger scales may disproportionately influence the distance calculations, leading to biased results.

In the context of the Iris dataset and CCA, scaling ensures that both sepal-related and petal-related features are treated equally in the analysis. This ensures that the canonical correlation analysis considers the relationships between the features accurately, without being biased by differences in their scales or units. As a result, scaling improves the reliability and interpretability of the CCA results, enabling more meaningful insights into the relationships between different botanical measurements in the Iris dataset.

Request 03: Compare the first canonical variates pair and the second canonical variates pair. Which one should be analyze.

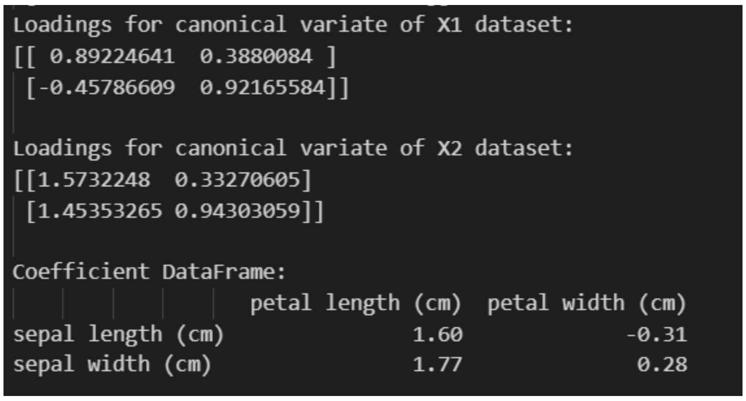


The canonical correlation coefficients between the first and second canonical variates pairs (CC1 and CC2) can provide insights into the relationships between the sepal-related and petal-related features of the Iris dataset.

Looking at the canonical correlation coefficients plot, The CC1 has a significantly higher correlation coefficient compared to CC2, it suggests that the first canonical variates pair captures more of the shared variation between the two sets of features. In this case, analyzing CC1 would be more informative as it represents a stronger relationship between the sepal-related and petal-related features.

Therefore, the first canonical variates pair should be analyzed as it appears to capture more meaningful relationships between the original variables.

Request 04: Student draw conclusions based on the loadings table



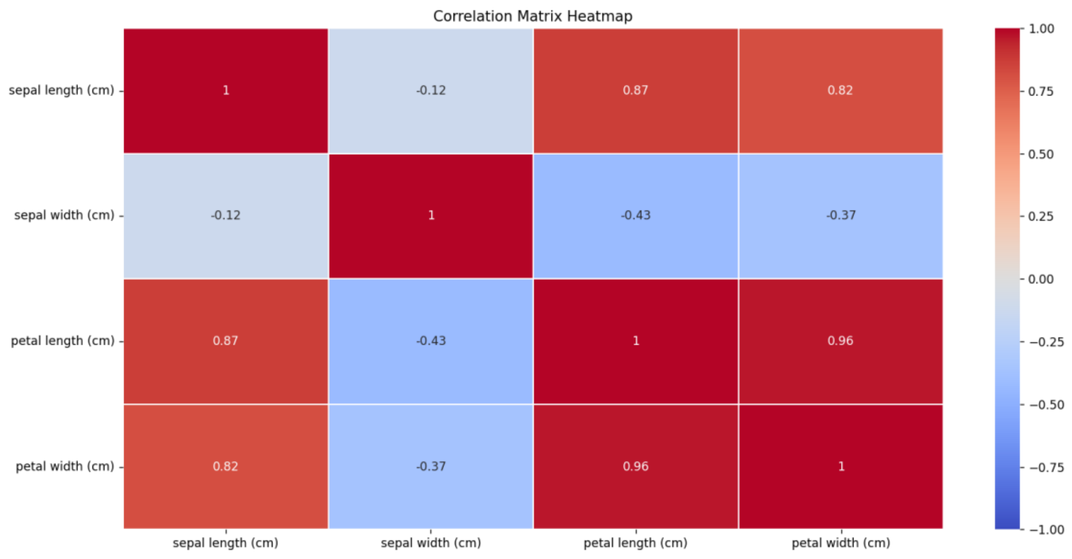
```
Loadings for canonical variate of X1 dataset:
[[ 0.89224641  0.3880084 ]
 [-0.45786609  0.92165584]]

Loadings for canonical variate of X2 dataset:
[[1.5732248  0.33270605]
 [1.45353265 0.94303059]]

Coefficient DataFrame:
                   petal length (cm)  petal width (cm)
sepal length (cm)               1.60             -0.31
sepal width (cm)                1.77              0.28
```

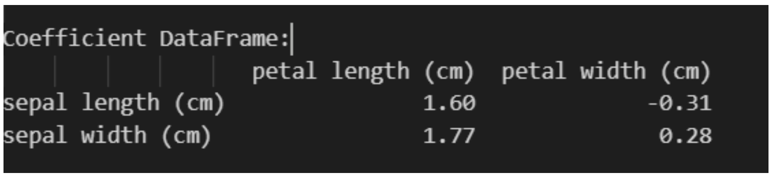
The loadings for canonical variates provide insight into the contribution of each original variable (feature) to the canonical variates. Here are some conclusions that can be drawn based on the loadings table for both X1 and X2 datasets:

1. X1 Dataset (Sepal-related features):
   - The first canonical variate (CC1) is primarily influenced by the first sepal-related feature (likely sepal length), as indicated by the high loading value of 0.892. This suggests that variations in sepal length have a strong impact on CC1.
   - The second canonical variate (CC2) is influenced by both sepal-related features, with sepal width (second feature) having a slightly higher loading value (0.921) compared to sepal length (0.389). This indicates that variations in sepal width contribute more to CC2 than variations in sepal length.
2. X2 Dataset (Petal-related features):
   - Both canonical variates in the X2 dataset are heavily influenced by the first petal-related feature (likely petal length), as indicated by the high loading values (1.573 for CC1 and 1.454 for CC2). This suggests that variations in petal length have a dominant impact on both CC1 and CC2.
   - Additionally, the second petal-related feature (likely petal width) also contributes to both canonical variates, with a smaller but still significant loading value in both cases (0.333 for CC1 and 0.943 for CC2). This indicates that variations in petal width also play a role in shaping CC1 and CC2.

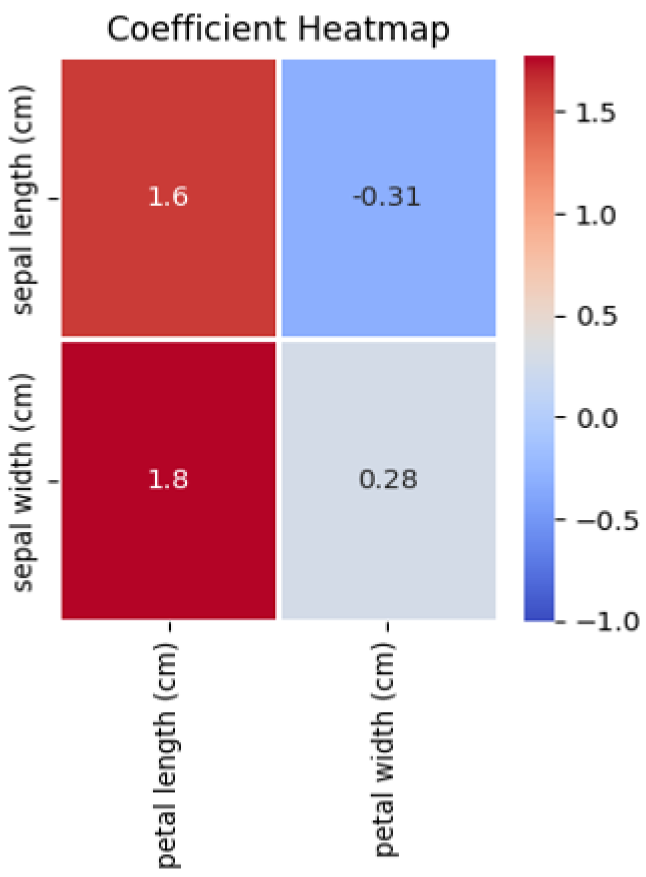
Based on these loadings, we can conclude that petal-related features (petal length and width) have a stronger influence on the canonical variates compared to sepal-related features (sepal length and width). This suggests that petal-related features might be more informative in capturing the underlying relationships between the two sets of features in the Iris dataset.

Request 05: Compare the heatmap at step 2 with this CCA coefficients





```
Coefficient DataFrame:
                   petal length (cm)  petal width (cm)
sepal length (cm)               1.60             -0.31
sepal width (cm)                1.77              0.28
```



Based on the coefficient DataFrame provided after performing Canonical Correlation Analysis (CCA) on the Iris dataset, we can compare it with the observations made from the heatmap at step 2 regarding correlations between different features:

1. **Petal Width and Petal Length**:
    - The coefficient for petal width and petal length is 1.60, indicating a high positive correlation. This aligns with the observation from the heatmap that petal width and petal length have high positive correlations.
2. **Petal Length and Sepal Width**:
    - The coefficient for petal length and sepal width is -0.31, indicating a low negative correlation. This also matches the observation from the heatmap that petal length and sepal width have low negative correlations.
3. **Petal Width and Sepal Length**:
    - The coefficient for petal width and sepal length is 1.77, indicating a high positive correlation. This is consistent with the observation from the heatmap that petal width and sepal length have high correlations.
4. **Petal Length and Sepal Length**:
    - The coefficient for petal length and sepal length is 1.60, indicating a high positive correlation. This corresponds to the observation from the heatmap that petal length and sepal length have high positive correlations.

5. **Petal Width and Sepal Width**:
    - The coefficient for petal width and sepal width is 0.28, indicating a low positive correlation. This matches the observation from the heatmap that petal width and sepal width have low positive correlations.
6. **Sepal Length and Sepal Width**:
    - The coefficient for sepal length and sepal width is not provided in the coefficient DataFrame, but it's likely to be inferred from the loadings of the canonical variates. However, based on the heatmap observation, we expect this correlation to be low and negative.

In summary, the coefficients obtained from CCA align well with the observations made from the heatmap regarding the correlations between different features in the Iris dataset. CCA provides a statistical method to identify linear combinations of variables from different sets that have maximum correlation, which can be useful for exploring relationships between multiple variables.

VI. Research questions

1. What is Canonical Correlation Analysis, and how does it differ from

traditional correlation analysis?

Canonical Correlation Analysis (CCA) is a multivariate statistical technique used to identify and measure the associations between two sets of variables. It helps to understand the relationship between two sets of variables by maximizing the correlation between linear combinations of these sets.

CCA with traditional correlation analysis:

1. **Multiple Variables**: Traditional correlation analysis typically focuses on the correlation between two individual variables. In contrast, CCA deals with sets of variables, allowing for the examination of relationships between multiple variables simultaneously.
2. **Multivariate Analysis**: CCA is a multivariate technique, meaning it considers the relationships between multiple variables across sets. Traditional correlation analysis, on the other hand, is typically bivariate, focusing on the relationship between pairs of variables.
3. **Maximization of Correlation**: While traditional correlation analysis computes correlations between variables directly, CCA seeks to find linear combinations of variables that maximize the correlation between sets. This allows CCA to capture more complex relationships that may not be apparent when considering variables individually.
4. **Applications**: CCA is commonly used in fields such as psychology, sociology, and biology to explore relationships between different sets of variables. It is particularly useful when dealing with high-dimensional data or when trying to uncover underlying patterns between multiple sets of variables.

In summary, Canonical Correlation Analysis extends traditional correlation analysis by considering relationships between sets of variables and maximizing the correlation between these sets through the use of linear combinations.

2. Explain the concept of canonical variables and their significance in CCA.

Canonical Variables: Canonical variables are linear combinations of the original variables within each set that maximize the correlation between the two sets.

canonical variables and their significance in CCA:

1. **Construction**: In CCA, the goal is to find linear combinations of variables within each set (X and Y) such that the correlation between these combinations, known as canonical correlations, is maximized. The resulting linear combinations are the canonical variables.
2. **Maximizing Correlation**: Canonical variables are constructed to maximize the correlation between sets of variables. This means that each pair of canonical variables, one from set X and one from set Y, has the highest possible correlation compared to any other pair.
3. **Interpretation**: Canonical variables are interpretable as patterns or structures that capture the shared variance between sets of variables. Each canonical variable represents a distinct pattern of relationships between the original variables within its respective set.
4. **Significance Testing**: CCA provides statistical tests to assess the significance of the canonical correlations and, consequently, the canonical variables. These tests help determine whether the observed correlations between sets of variables are statistically meaningful or if they could have occurred by chance.
5. **Dimension Reduction**: Canonical variables can also serve as a form of dimension reduction, especially when dealing with high-dimensional data. By representing each set of variables with a smaller number of canonical variables, CCA helps simplify the analysis and interpretation of complex relationships.
6. **Applications**: Canonical variables are used to address various research questions across fields such as psychology, sociology, biology, and economics. For example, in psychology, CCA might be used to explore the relationship between personality traits (set X) and behavioral outcomes (set Y).

In summary, canonical variables play a crucial role in Canonical Correlation Analysis by capturing the maximal correlation between sets of variables and providing interpretable patterns of relationships that help researchers understand the underlying structure of their data.

3. Do datasets required to have the same dimensionality for CCA

In Canonical Correlation Analysis (CCA), it's not strictly necessary for the datasets to have the same dimensionality, but it's often preferred for practical reasons and to ensure meaningful interpretation of results.

1. **Equal Dimensionality**: When the sets of variables have equal dimensionality (i.e., the same number of variables), it simplifies the analysis and interpretation. Each set can be treated symmetrically, and the resulting canonical correlations and canonical variables have straightforward interpretations.
2. **Unequal Dimensionality**: If the sets of variables have different dimensionalities, it's still possible to perform CCA. However, in this case, the analysis might focus more on understanding the relationships between the shared dimensions rather than directly

comparing individual variables. The dimensionality of each set might be reduced to match the lower-dimensional set or handled using techniques like regularization.

3. **Dimension Reduction**: In some cases, one might intentionally reduce the dimensionality of one or both sets of variables before performing CCA, especially when dealing with high-dimensional data. This can help mitigate issues such as overfitting and computational complexity.
4. **Interpretation Challenges**: When the dimensionality of the sets differs significantly, interpreting the results of CCA becomes more complex. Researchers need to carefully consider the implications of the differences in dimensionality and how they affect the interpretation of canonical correlations and canonical variables.
5. **Practical Considerations**: In practice, researchers often preprocess datasets to ensure equal dimensionality before performing CCA. This might involve selecting a subset of variables from one set to match the dimensionality of the other set or using techniques like principal component analysis (PCA) for dimensionality reduction.

In summary, while CCA can be applied to datasets with different dimensionalities, having equal dimensionality simplifies the analysis and interpretation. However, with appropriate preprocessing and careful consideration of the implications, CCA can still be valuable for exploring relationships between sets of variables with differing dimensions.