Assignment

Objective:

The objective of this assignment is to implement PCA on a given dataset and analyse the results.

Instructions:

Download the wine dataset from the UCI Machine Learning Repository

Load the dataset into a Pandas dataframe.

Split the dataset into features and target variables.

Perform data preprocessing (e.g., scaling, normalisation, missing value imputation) as necessary.

Implement PCA on the preprocessed dataset using the scikit-learn library.

Determine the optimal number of principal components to retain based on the explained variance ratio.

Visualise the results of PCA using a scatter plot.

Perform clustering on the PCA-transformed data using K-Means clustering algorithm.

Interpret the results of PCA and clustering analysis.

(https://archive.ics.uci.edu/ml/datasets/Wine).

Below is a step-by-step guide on how to implement PCA on the Wine dataset, analyze the results, and perform clustering using the K-Means algorithm. For this, we'll use Python with the pandas, scikit-learn, and matplotlib libraries. Make sure you have these libraries installed before running the code.

python

Copy code

```
import        as
import       as
import                    as
from                        import
from                        import
from                  import
from                  import
from                 import


      "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
                "Class"  "Alcohol"  "Malic Acid"  "Ash"  "Alcalinity of Ash"
"Magnesium"
 "Total Phenols"  "Flavanoids"  "Nonflavanoid Phenols"  "Proanthocyanins"
 "Color Intensity"  "Hue"  "OD280/OD315 of Diluted Wines"  "Proline"
```

```
                                    None

print

                    "Class"       1
               "Class"


                               'mean'









          range 1  len                              1
                                    'o'
            'Number of Principal Components'
            'Cumulative Explained Variance'
           'Explained Variance Ratio'









                     0            1             'viridis'            'k'    50
            'Principal Component 1'
            'Principal Component 2'
           'PCA Results'
```

```
            range 2  10

for   in
                                        42


                                       'o'
          'Number of Clusters (k)'
          'Silhouette Score'
         'K-Means Clustering: Silhouette Score vs. Number of Clusters'


                                  max                    2

                                                42
```

This code covers the key steps of loading the dataset, preprocessing the data, implementing PCA, determining the optimal number of components, visualizing the results, and performing clustering using the K-Means algorithm. Adjustments may be needed based on the specific goals and characteristics of the Wine dataset.

Deliverables:

Jupyter notebook containing the code for the PCA implementation.
the code for PCA implementation in a Jupyter notebook format. However, keep in mind that the execution of the code requires an active internet connection to download the Wine dataset from the UCI Machine Learning Repository. Please ensure that you have the required libraries installed by running `pip install pandas numpy scikit-learn matplotlib`.

python

Copy code

```
import        as
import       as
import                    as
from                        import
from                        import
from                  import
from                  import
from                 import
```

```
      "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
                "Class"  "Alcohol"  "Malic Acid"  "Ash"  "Alcalinity of Ash"
"Magnesium"
 "Total Phenols"  "Flavanoids"  "Nonflavanoid Phenols"  "Proanthocyanins"
 "Color Intensity"  "Hue"  "OD280/OD315 of Diluted Wines"  "Proline"
```

```
                                  None
```

```
                   "Class"       1
              "Class"
```

```
                               'mean'
```

```
range 1  len                                1
                              'o'
  'Number of Principal Components'
  'Cumulative Explained Variance'
 'Explained Variance Ratio'









           0            1             'viridis'            'k'    50
  'Principal Component 1'
  'Principal Component 2'
 'PCA Results'



  range 2  10


for   in
                                    42




                                     'o'
  'Number of Clusters (k)'
  'Silhouette Score'
 'K-Means Clustering: Silhouette Score vs. Number of Clusters'
```

```
max                    2

                       42
```

Please create a new Jupyter notebook, copy and paste the code into cells, and run them sequentially. Ensure you have an active internet connection to download the dataset.

Report: PCA and Clustering Analysis on Wine Dataset

Executive Summary:

This report presents the results of Principal Component Analysis (PCA) and clustering analysis performed on the Wine dataset obtained from the UCI Machine Learning Repository. The

objective was to gain insights into the underlying patterns within the dataset and identify potential clusters among the observations.

Dataset Overview:

The Wine dataset consists of 178 observations with 13 features representing different chemical properties of wine, and a target variable indicating the class of wine. The dataset was preprocessed, including imputation of missing values and scaling of features.

Principal Component Analysis (PCA):

PCA Implementation:

PCA was applied to reduce the dimensionality of the dataset while preserving its variance. The optimal number of principal components was determined based on the explained variance ratio.

Explained Variance Ratio:

The plot of cumulative explained variance ratio revealed an elbow point, suggesting that the first few principal components capture a significant portion of the dataset's variance. The number of principal components chosen can be based on this elbow point.

Scatter Plot of PCA Results:

A scatter plot of the first two principal components highlighted the distribution of observations in reduced-dimensional space. Each point on the plot represents a wine sample colored by its class, providing visual insights into potential patterns.

Clustering Analysis:

K-Means Clustering:

K-Means clustering was applied to the PCA-transformed data to identify natural groupings within the dataset. The optimal number of clusters was determined using silhouette scores for various values of k.

Silhouette Score vs. Number of Clusters:

The silhouette score plot indicated an optimal number of clusters, where the highest silhouette score was observed. This value guides the selection of the number of clusters for K-Means clustering.

Interpretation of Clusters:

The results of clustering provide a segmentation of the dataset into distinct groups. Further analysis can be conducted to interpret the characteristics of each cluster and understand how the chemical properties contribute to the separation.

Conclusion:

The combined use of PCA and K-Means clustering on the Wine dataset has revealed valuable insights into the inherent patterns within the data. This analysis can be leveraged for various applications, such as targeted marketing or quality assessment of different wine classes.

Recommendations:

- Investigate the chemical properties that contribute the most to the separation of clusters.
- Explore additional unsupervised learning techniques to further validate cluster assignments.
- Consider the practical implications of identified clusters, especially in the context of winemaking or marketing strategies.

This report serves as a foundation for deeper exploration and application of the identified patterns in the Wine dataset.

*Scatter plot showing the results of PCA.

code snippet to generate a scatter plot showing the results of PCA on the Wine dataset:

python

Copy code

```
import        as
import       as
import                  as
from                      import
from                      import
from               import


      "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
                "Class"  "Alcohol"  "Malic Acid"  "Ash"  "Alcalinity of Ash"
"Magnesium"
 "Total Phenols"  "Flavanoids"  "Nonflavanoid Phenols"  "Proanthocyanins"
 "Color Intensity"  "Hue"  "OD280/OD315 of Diluted Wines"  "Proline"

                                  None


                    "Class"       1
               "Class"


                                 'mean'









                   10  8
                    0           1             'viridis'            'k'    50
            'Principal Component 1'
            'Principal Component 2'
           'PCA Results - Scatter Plot'
                   'Wine Class'
```

This code generates a scatter plot with the first two principal components on the x-axis and y-axis, respectively. Each point is colored based on the wine class. Adjust the figure size, color map, and other parameters as needed to suit your preferences.

*A table showing the performance metrics for the clustering algorithm.
In the context of K-Means clustering, common performance metrics include the Silhouette Score and possibly the Inertia or Sum of Squared Errors (SSE). Here's an example code snippet to compute and display these metrics for different values of k:

python

Copy code

```
from                  import
from                  import

            range 2   11



for    in
                                          42

 'Number of Clusters (k)'
 'Silhouette Score'
 'Inertia (SSE)'
```

```
print
```

This code performs K-Means clustering for different values of k, computes the Silhouette Score and Inertia for each clustering, and stores the results in a Pandas DataFrame. You can customize this table further based on your specific requirements or include additional metrics depending on your analysis goals.

Additional Information:

*You can use the python programming language.
code snippet in Python using the `pandas` library to create a table showing the performance metrics for the clustering algorithm (K-Means):

python

Copy code

```
import        as
from                  import
from                  import
import                    as


           range 2   11



for    in
                                    42
```

```
 'Number of Clusters (k)'
 'Silhouette Score'
 'Inertia (SSE)'


print "Performance Metrics for K-Means Clustering:"
print


                  10  5


            1  2  1
                                          'o'
           'Number of Clusters (k)'
           'Silhouette Score'
          'Silhouette Score vs. Number of Clusters'


            1  2  2
                                       'o'
           'Number of Clusters (k)'
           'Inertia (SSE)'
          'Inertia vs. Number of Clusters'
```

This code not only creates a table with the performance metrics but also plots the Silhouette Score and Inertia to help visually identify the optimal number of clusters. You can customize the code based on your specific requirements and preferences.

*You can use any other machine learning libraries or tools as necessary.

I'll use the `yellowbrick` library to create a visual representation of the silhouette score and inertia for different values of k during K-Means clustering. First, install the `yellowbrick` library if you haven't already:

bash

Copy code

Now, you can use the following code:

python

Copy code

```
import        as
from                  import
from                  import
from                      import
import                    as


           range 2   11


for    in
                                          42
```

```
                                                                    42
    'yellowbrick'


                                                                42      2   11
```

This code uses `yellowbrick` to create a silhouette plot and an elbow plot for K-Means clustering. The silhouette plot helps visualize the silhouette scores for different cluster numbers, and the elbow plot helps identify the optimal number of clusters based on the inertia (sum of squared errors). Adjust the code based on your specific needs and preferences.

*You can use any visualisation libraries or tools as necessary.
I'll use the `matplotlib` and `seaborn` libraries for visualization. If you haven't installed these libraries, you can do so using:

bash

Copy code

Here's the modified code:

python

Copy code

```
import        as
from                import
from                import
import                  as
import         as
```

```
            range 2  11


for   in
                                          42


                 12  5

             1  2  1
                                        'o'
             'Number of Clusters (k)'
             'Silhouette Score'
            'Silhouette Score vs. Number of Clusters'


             1  2  2
                                      'o'
             'Number of Clusters (k)'
             'Inertia (SSE)'
            'Inertia vs. Number of Clusters'
```

This code will generate a figure with two subplots: one for the Silhouette Score and another for the Inertia. Adjust the code according to your preferences or use other visualization libraries as needed.