Experiment – 8

Dimensionality Reduction: Apply dimensionality reduction techniques like Principal Component Analysis (PCA) or t-distributed Stochastic Neighbor Embedding (t-SNE) to visualize and reduce the number of features.

Date Set: *The dataset contains measurements of four features of iris flowers: Sepal Length, Sepal Width, Petal Length, and Petal Width. These measurements are provided in centimeters (cm). Additionally, each data point is associated with a specific species of iris flower. Each row in the dataset represents a single observation or measurement of an iris flower. The dataset contains a total of 30 observations.*

Objective:

Apply dimensionality reduction techniques to visualize and reduce the number of features.

Tasks:

1. Implement Principal Component Analysis (PCA) or t-SNE.
2. Visualize the reduced-dimensional dataset.
3. Evaluate the impact of dimensionality reduction on model performance.

Dimensionality Reduction is a technique used to reduce the number of features (or dimensions) in a dataset while preserving its essential information. This is beneficial for several reasons:

1. **Curse of Dimensionality:** High-dimensional datasets are prone to the curse of dimensionality, where the data becomes sparse and the distance between points becomes less meaningful. Dimensionality reduction can help alleviate this problem by reducing the number of features.
2. **Visualization:** It's challenging to visualize high-dimensional data. Dimensionality reduction techniques transform the data into a lower-dimensional space that can be easily visualized, allowing for better understanding and interpretation.
3. **Computational Efficiency:** Many machine learning algorithms suffer from increased computational complexity with higher dimensions. Dimensionality reduction can lead to faster training and inference times by reducing the number of features.

Two commonly used dimensionality reduction techniques are:

1. **Principal Component Analysis (PCA):** PCA is a linear dimensionality reduction technique that identifies the axes (principal components) along which the data varies the most. It projects the data onto these axes, preserving as much variance as possible. The resulting principal components are orthogonal to each other, and they capture the most important information in the data. PCA is suitable for datasets with continuous numerical features.
2. **t-distributed Stochastic Neighbor Embedding (t-SNE):** t-SNE is a non-linear dimensionality reduction technique that focuses on preserving the local structure of the data. It maps high-dimensional data to a lower-dimensional space, such as 2D or 3D, while preserving the pairwise similarities between data points. t-SNE is particularly useful for visualizing high-dimensional data clusters or manifold structures.

The choice between PCA and t-SNE depends on the specific characteristics of the dataset and the goals of the analysis. PCA is faster and more scalable but may not capture complex non-linear relationships in the data as effectively as t-SNE. On the other hand, t-SNE can provide more insightful visualizations but is computationally expensive and may suffer from overfitting in high-dimensional spaces.

Python Program:

Importing the libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
```

Note:

1. **matplotlib.pyplot**: This library provides functions for creating visualizations, such as scatter plots, histograms, etc.
2. **seaborn**: Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
3. **PCA** (Principal Component Analysis) from **sklearn.decomposition**: PCA is a technique used for dimensionality reduction. It linearly transforms the data into a new coordinate system such that the greatest variance lies on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.
4. **TSNE** (t-distributed Stochastic Neighbor Embedding) from **sklearn.manifold**: t-SNE is a nonlinear dimensionality reduction technique that is particularly well-suited for visualizing high-dimensional data. It aims to preserve the local structure of the data by modeling the similarities between points in high-dimensional space and mapping them to a lower-dimensional space while minimizing the divergence between the distributions of pairwise similarities in the original and lower-dimensional spaces.

Importing the dataset

```
# Load the dataset
data = pd.read_csv(r'C:\Users\KUNAL
KUNDU\OneDrive\Desktop\GHRCEM\...\Exp-8\Experiment-8.csv')

# Display the first few rows of the dataset
print("Shape of the dataset:", data.shape)
data.head()
```

Note:

1. **data.shape**: This prints the shape of the dataset, which consists of two values - the number of rows and the number of columns. For example, if the output is **(30, 5)**, it means that the dataset has 30 rows and 5 columns.

2. **data.head()**: This function displays the first few rows of the dataset. By default, it shows the first 5 rows, but you can specify the number of rows to display by passing an argument (e.g., **data.head(10)** to display the first 10 rows).

Output:

```
Shape of the dataset: (30, 5)
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Explore the Dataset

```
# Summary statistics
print("Summary Statistics:")
print(data.describe())

# Visualize the distribution of features
plt.figure(figsize=(10, 6))
sns.pairplot(data, hue='Species')
plt.title("Pairplot of Iris Dataset Features")
plt.show()
```

Note:

This code block serves two main purposes:

1. **Summary Statistics**:
    - **print("Summary Statistics:")**: This line prints a header indicating that summary statistics are about to be displayed.
    - **print(data.describe())**: This line prints summary statistics of the dataset. **describe()** is a pandas function that generates descriptive statistics of the numerical columns in the DataFrame, such as count, mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum values.
2. **Visualize the Distribution of Features**:
    - **plt.figure(figsize=(10, 6))**: This line initializes a new figure for plotting with a specific size of 10 inches in width and 6 inches in height.
    - **sns.pairplot(data, hue='Species')**: This line creates a pairplot using Seaborn (**sns**). A pairplot is a grid of pairwise scatterplots that visualize the relationships

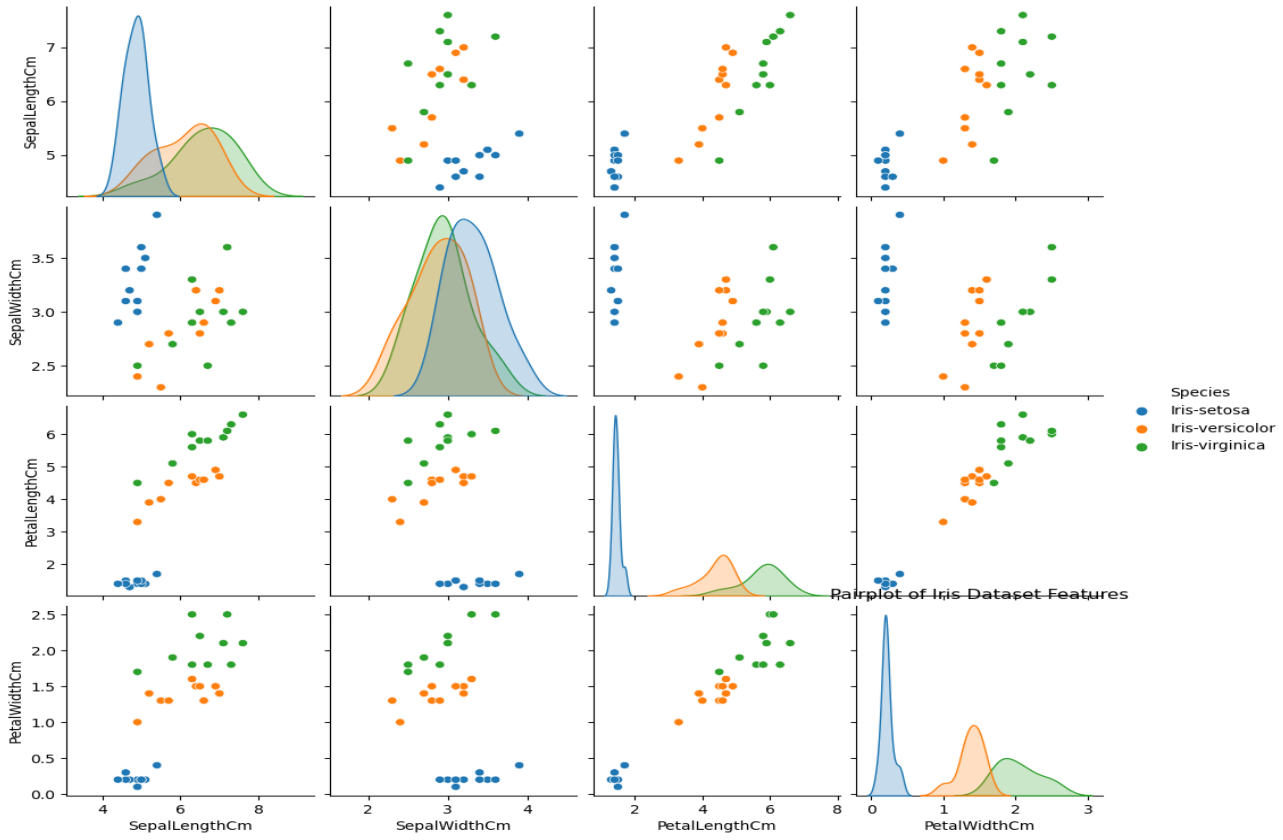
between different pairs of features. The **hue** parameter is set to 'Species', which colors the data points according to the different species of iris flowers.

- **plt.title("Pairplot of Iris Dataset Features")**: This line sets the title of the plot.
- **plt.show()**: This line displays the plot.

Overall, this code block provides both numerical summary statistics and visual insights into the distribution and relationships between features in the iris dataset, aiding in exploratory data analysis.

Output:

```
Summary Statistics:
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count      30.000000     30.000000      30.000000     30.000000
mean        5.843333      3.040000       3.863333      1.213333
std         0.964073      0.372873       1.881394      0.789034
min         4.400000      2.300000       1.300000      0.100000
25%         4.925000      2.825000       1.500000      0.225000
50%         5.750000      3.000000       4.500000      1.400000
75%         6.575000      3.275000       5.475000      1.800000
max         7.600000      3.900000       6.600000      2.500000
<Figure size 1000x600 with 0 Axes>
```



Apply Principal Component Analysis (PCA)

```
# Separate features and target variable
X = data.drop('Species', axis=1)
y = data['Species']
```

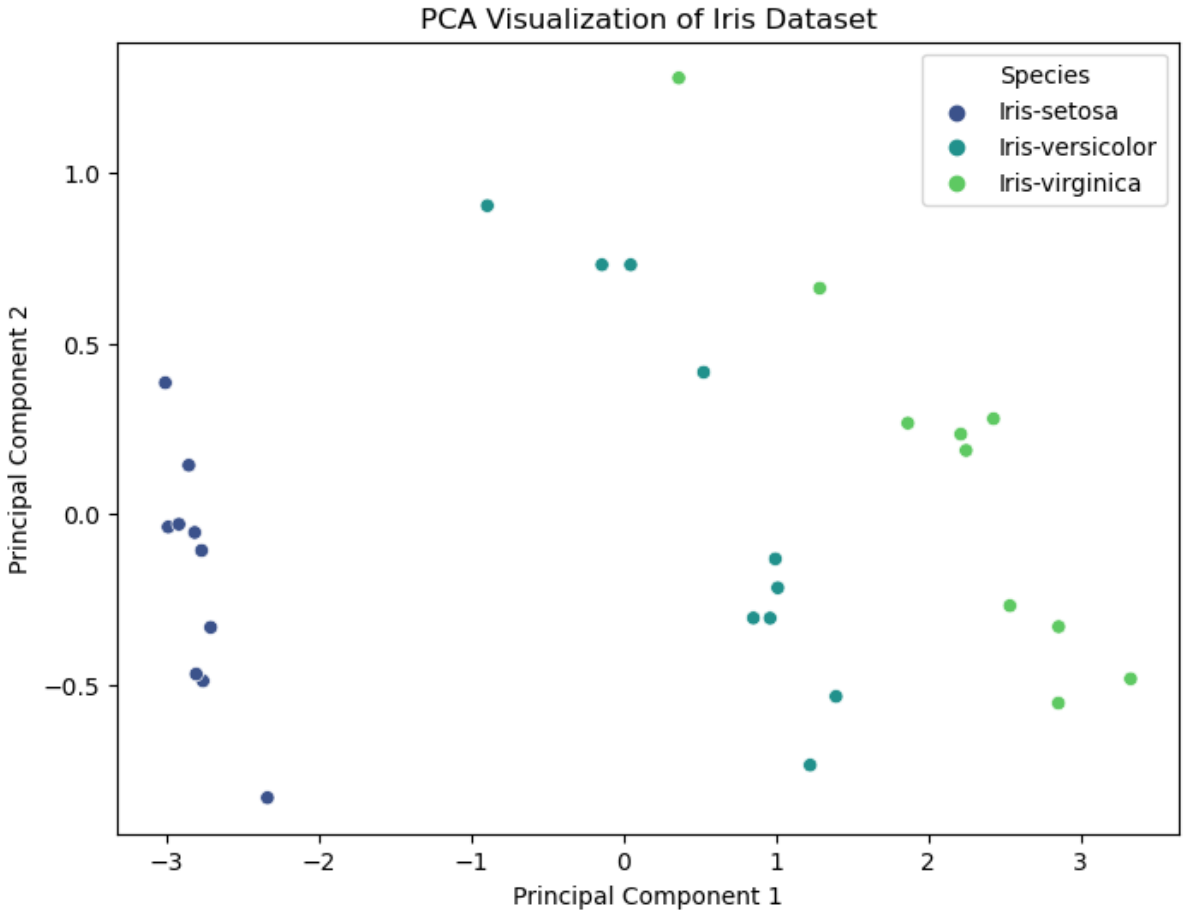
```
# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Visualize PCA results
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:,   0],   y=X_pca[:,   1],   hue=y,
palette='viridis')
plt.title("PCA Visualization of Iris Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Note:
This code block performs Principal Component Analysis (PCA) on the Iris dataset and visualizes the results:

1. **Separate features and target variable**:
    - **X = data.drop('Species', axis=1)**: This line creates a new DataFrame **X** by dropping the 'Species' column from the original dataset. **axis=1** indicates that we are dropping a column (as opposed to a row).
    - **y = data['Species']**: This line creates a Series **y** containing the target variable 'Species' from the original dataset.
2. **Apply PCA**:
    - **pca = PCA(n_components=2)**: This line initializes a PCA object with **n_components** set to 2, indicating that we want to reduce the dimensionality of the data to 2 principal components.
    - **X_pca = pca.fit_transform(X)**: This line applies PCA to the feature matrix **X** and transforms it into a new coordinate system defined by the principal components. **fit_transform()** fits the PCA model to the data and then transforms the data into the new coordinate system.
3. **Visualize PCA results**:
    - **plt.figure(figsize=(8, 6))**: This line initializes a new figure for plotting with a specific size of 8 inches in width and 6 inches in height.
    - **sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=y, palette='viridis')**: This line creates a scatter plot using Seaborn (**sns**). It plots the first and second principal components (**X_pca[:, 0]** and **X_pca[:, 1]**) on the x and y axes, respectively. The points are colored according to the target variable **y** (species) using the 'viridis' color palette.
    - **plt.title("PCA Visualization of Iris Dataset")**: This line sets the title of the plot.
    - **plt.xlabel("Principal Component 1")**: This line sets the label for the x-axis.
    - **plt.ylabel("Principal Component 2")**: This line sets the label for the y-axis.
    - **plt.show()**: This line displays the plot.

Basically, this code block applies PCA to reduce the dimensionality of the Iris dataset to 2 principal components and visualizes the transformed data in a scatter plot. Each point in the plot represents an observation (flower), and its position is determined by its values along the first and second principal components.

Output:



Apply t-distributed Stochastic Neighbor Embedding (t-SNE)

```
# Apply t-SNE with reduced perplexity
tsne = TSNE(n_components=2, perplexity=10, random_state=42)  # Set perplexity to a lower value, e.g., 10
X_tsne = tsne.fit_transform(X)

# Visualize t-SNE results
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_tsne[:, 0], y=X_tsne[:, 1], hue=y, palette='viridis')
plt.title("t-SNE Visualization of Iris Dataset")
plt.xlabel("t-SNE Component 1")
plt.ylabel("t-SNE Component 2")
plt.show()
```

Note:

Here's a detailed breakdown of this code segment:

1. **Apply t-SNE with reduced perplexity**:
    - **tsne = TSNE(n_components=2, perplexity=10, random_state=42)**: This line initializes a t-SNE object with **n_components** set to 2, indicating that we want to reduce the dimensionality of the data to 2 dimensions. The **perplexity**

parameter controls the number of effective neighbors used in the algorithm. Lowering the perplexity can help mitigate the error encountered previously.

- **X_tsne = tsne.fit_transform(X)**: This line applies t-SNE to the feature matrix **X** and transforms it into a new 2-dimensional space defined by the t-SNE components. **fit_transform()** fits the t-SNE model to the data and then transforms the data into the new 2D space.

2. **Visualize t-SNE results**:
   - **plt.figure(figsize=(8, 6))**: This line initializes a new figure for plotting with a specific size of 8 inches in width and 6 inches in height.
   - **sns.scatterplot(x=X_tsne[:, 0], y=X_tsne[:, 1], hue=y, palette='viridis')**: This line creates a scatter plot using Seaborn (**sns**). It plots the t-SNE components (**X_tsne[:, 0]** and **X_tsne[:, 1]**) on the x and y axes, respectively. The points are colored according to the target variable **y** (species) using the 'viridis' color palette.
   - **plt.title("t-SNE Visualization of Iris Dataset")**: This line sets the title of the plot.
   - **plt.xlabel("t-SNE Component 1")**: This line sets the label for the x-axis.
   - **plt.ylabel("t-SNE Component 2")**: This line sets the label for the y-axis.
   - **plt.show()**: This line displays the plot.

This code block applies t-SNE to reduce the dimensionality of the Iris dataset to 2 dimensions with a lower perplexity value, and then visualizes the transformed data in a scatter plot. Each point in the plot represents an observation (flower), and its position is determined by its values along the t-SNE components. The plot helps visualize the clusters formed by different species of iris flowers in the reduced 2D space.

Output: