**Ex. No.: 10**
**Date: 10/11/24**

# A PYTHON PROGRAM TO IMPLEMENT DIMENSIONALITY REDUCTION USING PCA

**Aim:**

To implement Dimensionality Reduction using PCA in a python program.

**Algorithm:**

Step 1: Import Libraries

Import necessary libraries, including pandas, numpy, matplotlib.pyplot, and sklearn.decomposition.PCA.

Step 2: Load the Dataset (iris dataset)

Load your dataset into a pandas DataFrame.

Step 3: Standardize the Data

Standardize the features of the dataset using StandardScaler from sklearn.preprocessing.

Step 4: Apply PCA

- Create an instance of PCA with the desired number of components.
- Fit PCA to the standardized data.
- Transform the data to its principal components using transform.

Step 5: Explained Variance Ratio

- Calculate the explained variance ratio for each principal component.
- Plot a scree plot to visualize the explained variance ratio.

Step 6: Choose the Number of Components

Based on the scree plot, choose the number of principal components that explain a significant amount of variance.

Step 7: Apply PCA with Chosen Components

Apply PCA again with the chosen number of components.

Step 8: Visualize the Reduced Data

- Transform the original data to the reduced dimension using the fitted PCA.

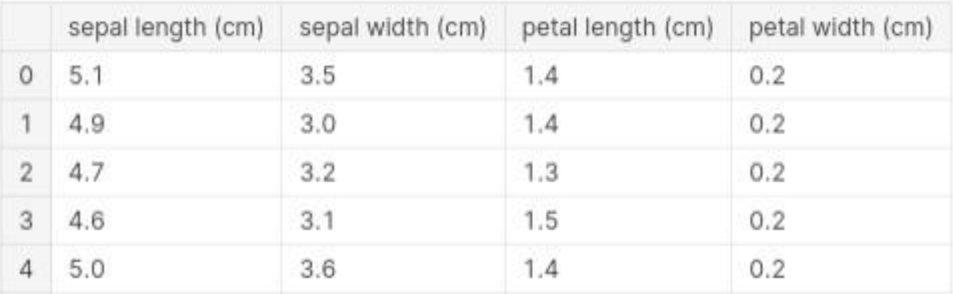
- Visualize the reduced data using a scatter plot.

Step 9: Interpretation

Interpret the results, considering the trade-offs between dimensionality reduction and information loss.

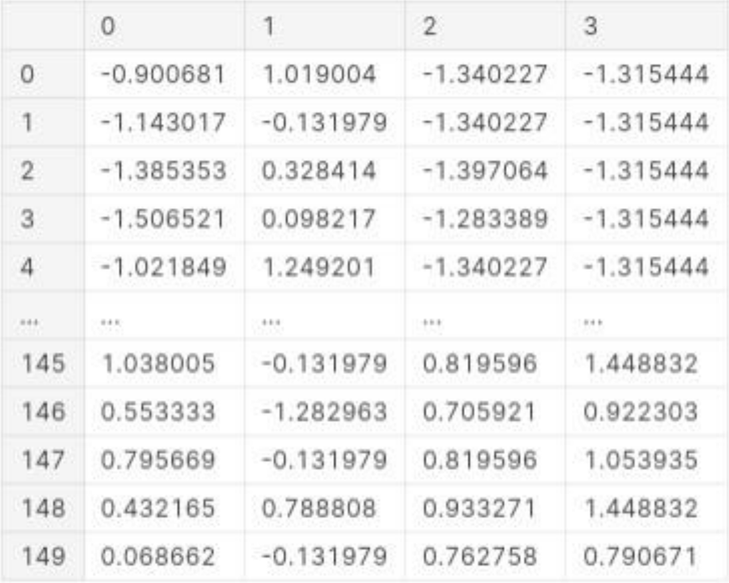
**PROGRAM:**

```
from sklearn import datasets import pandas as pd from
sklearn.preprocessing import StandardScaler from
sklearn.decomposition import PCA import seaborn as sns iris =
datasets.load_iris() df = pd.DataFrame(iris['data'], columns =
iris['feature_names']) df.head()
```



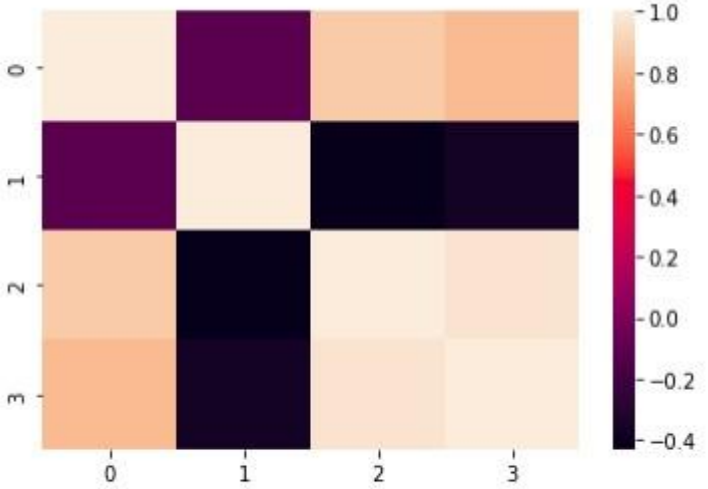
|   | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 |

```
scalar = StandardScaler()
scaled_data = pd.DataFrame(scalar.fit _transform(df)) #scaling the data scaled_data
sns.heatmap(scaled_data.corr())
```

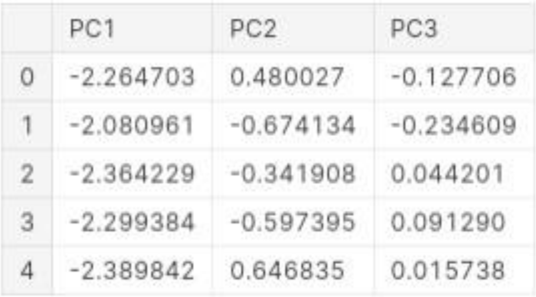


|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | -0.900681 | 1.019004 | -1.340227 | -1.315444 |
| 1 | -1.143017 | -0.131979 | -1.340227 | -1.315444 |
| 2 | -1.385353 | 0.328414 | -1.397064 | -1.315444 |
| 3 | -1.506521 | 0.098217 | -1.283389 | -1.315444 |
| 4 | -1.021849 | 1.249201 | -1.340227 | -1.315444 |
| ... | ... | ... | ... | ... |
| 145 | 1.038005 | -0.131979 | 0.819596 | 1.448832 |
| 146 | 0.553333 | -1.282963 | 0.705921 | 0.922303 |
| 147 | 0.795669 | -0.131979 | 0.819596 | 1.053935 |
| 148 | 0.432165 | 0.788808 | 0.933271 | 1.448832 |
| 149 | 0.068662 | -0.131979 | 0.762758 | 0.790671 |

```
<AxesSubplot:>
```

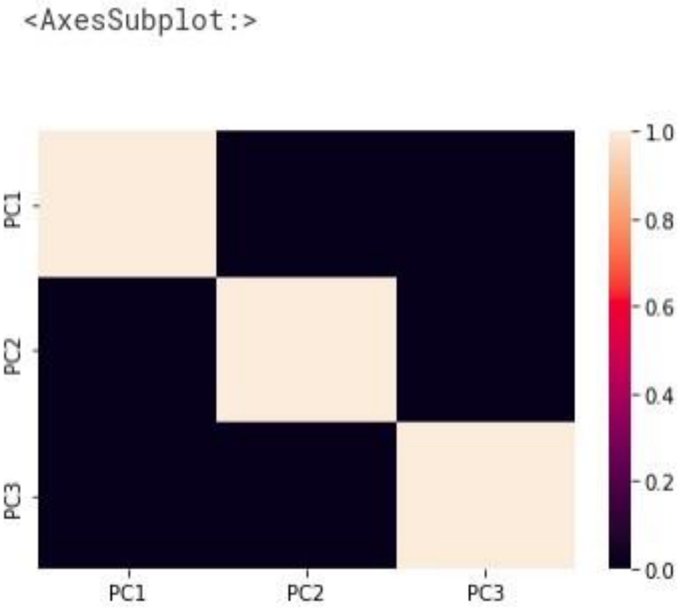


```
pca = PCA(n_components = 3) pca.fit(scaled_data) data_pca =
pca.transform(scaled_data) data_pca =
pd.DataFrame(data_pca,columns=['PC1','PC2','PC3'])
data_pca.head()
```



| | PC1 | PC2 | PC3 |
|---|---|---|---|
| 0 | -2.264703 | 0.480027 | -0.127706 |
| 1 | -2.080961 | -0.674134 | -0.234609 |
| 2 | -2.364229 | -0.341908 | 0.044201 |
| 3 | -2.299384 | -0.597395 | 0.091290 |
| 4 | -2.389842 | 0.646835 | 0.015738 |

```
sns.heatmap(data_pca.corr())
```



```
<AxesSubplot:>
```

**RESULT:-**

Thus Dimensionality Reduction has been implemented using PCA in a python program successfully and the results have been analyzed.