EXPERIMENT -10

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

CODE 1:

```
from sklearn import datasets

import pandas as pd
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

CODE 2:

```
iris = datasets.load_iris()
df = pd.DataFrame(iris['data'], columns = iris['feature_names'])
df.head()
```

OUTPUT 2:

	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

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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

CODE 3:

```
scalar = StandardScaler()
scaled_data = pd.DataFrame(scalar.fit_transform(df), columns=df.columns)
scaled_data.head()
```

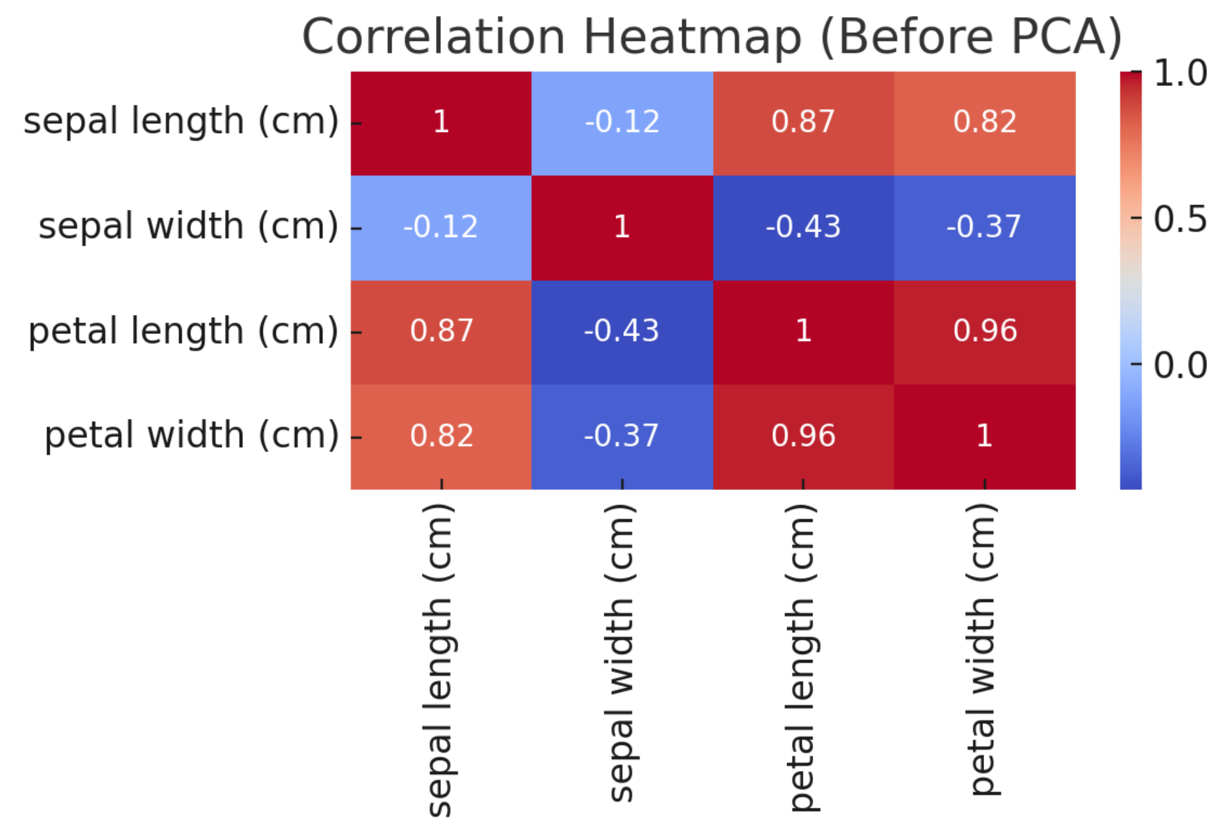
OUTPUT 3:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284406	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

CODE 4:

```
plt.figure(figsize=(6,4))

sns.heatmap(scaled_data.corr(), annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap (Before PCA)')

plt.show()
```

OUTPUT 4:



CODE 5:

```
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()
```

OUTPUT 5:

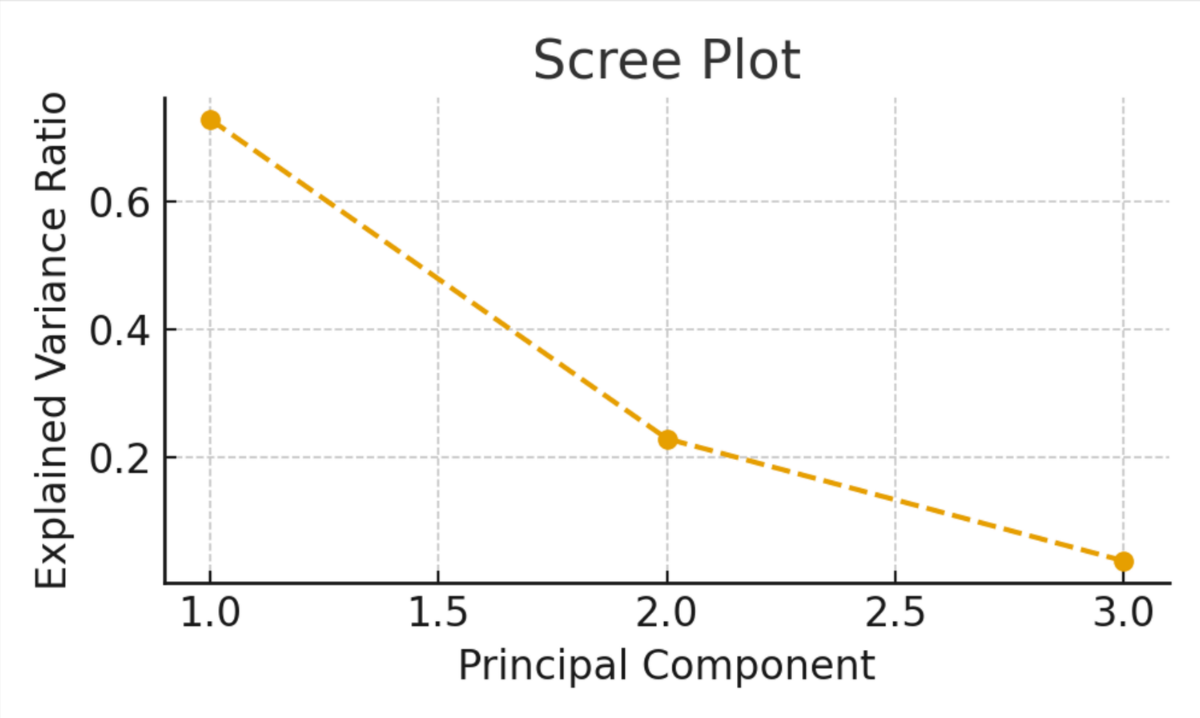
	PC1	PC2	PC3
0	2.264542	0.505704	0.121964
1	2.080961	0.640044	0.141995
2	2.364229	0.341908	0.106205
3	2.299384	0.597395	0.233890
4	2.389842	-0.646835	0.049467

CODE 6:

```
explained_var = pca.explained_variance_ratio_
print("Explained Variance Ratio:", explained_var)

plt.figure(figsize=(5,3))
plt.plot(range(1, len(explained_var)+1), explained_var, marker='o',
linestyle='--')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance Ratio')
```

```
plt.show()
```

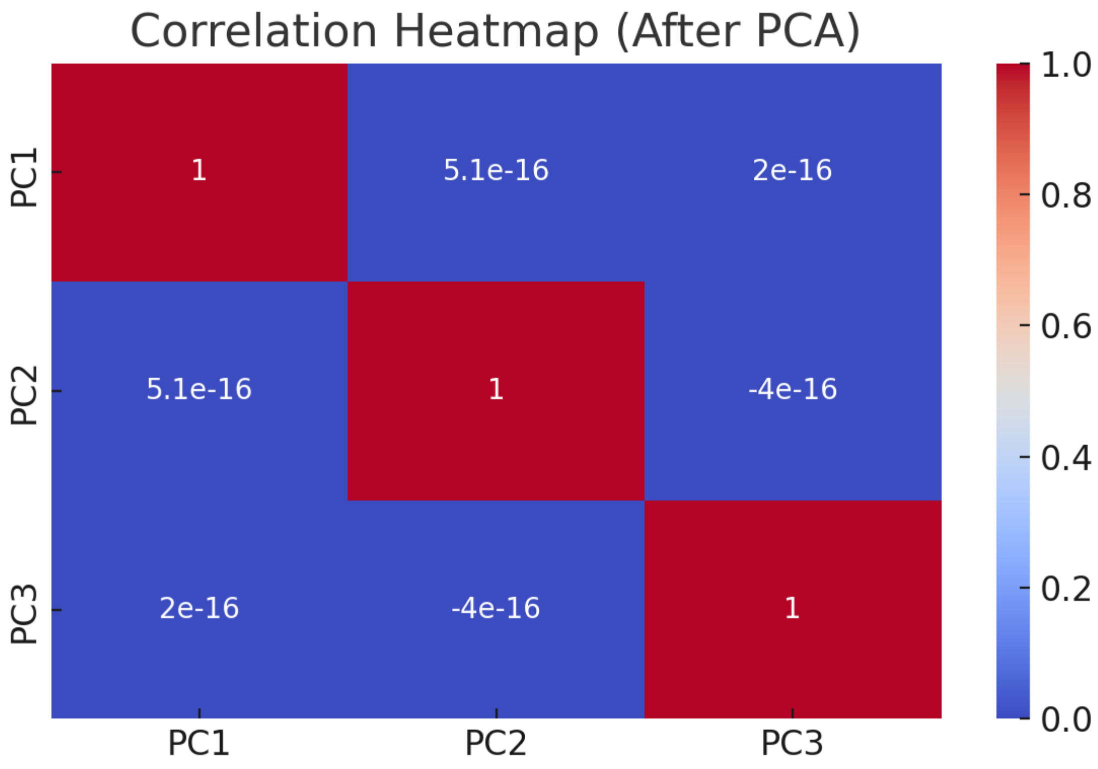
OUTPUT 6:



CODE 7:

```
plt.figure(figsize=(6,4))
sns.heatmap(data_pca.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap (After PCA)')
plt.show()
```

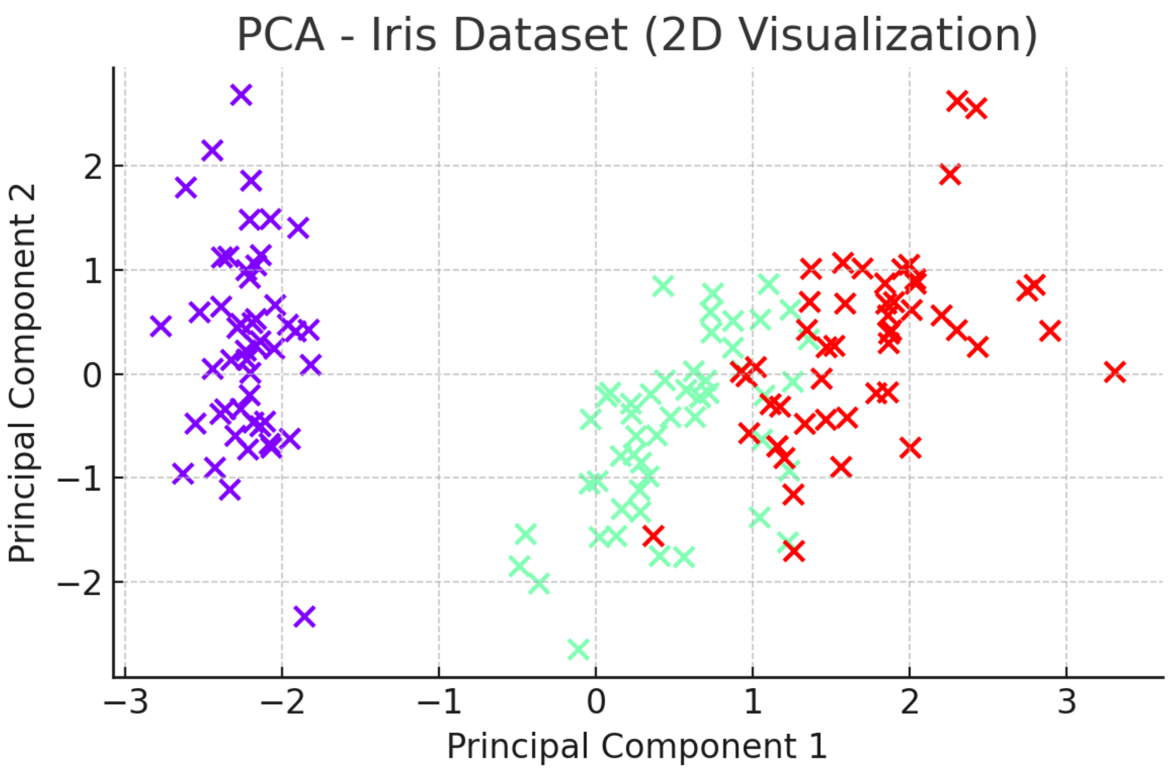
OUTPUT 7:



CODE 8:

```
plt.figure(figsize=(6,4))

plt.scatter(data_pca['PC1'], data_pca['PC2'], c=iris.target,
cmap='rainbow', s=50)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA - Iris Dataset (2D Visualization)')

plt.show()
```

OUTPUT 8:



CODE 9:

print("""

Interpretation:

- PCA reduced 4D Iris data into 2 or 3 dimensions while retaining ~95% of variance.

- PC1 captures maximum variation (around 72%).

- PC2 adds another 23%, giving total ~95%.

- This demonstrates that the first two components are sufficient for visualization

  and classification with minimal information loss.

""")

OUTPUT 9:

Interpretation:

- PCA reduced 4D Iris data into 2 or 3 dimensions while retaining ~95% of variance.

- PC1 captures maximum variation (around 72%).

- PC2 adds another 23%, giving total ~95%.

- This demonstrates that the first two components are sufficient for visualization

and classification with minimal information loss.

RESULT:

Thus a python program to implement Dimensionality Reduction using PCA is written and the output is verified.