Roll No.: A-25	**Name:SHRADDHA SAWANT**
Class: T.E.(AI & DS)	**Batch: A-2**
Date Of Performance: 15/03/25	**Date Of Submission:07/04/25**
Grade:	

Experiment No. 07

Aim: Implementation of Principal Component Analysis (PCA).

Outcome:

After completing this practical, Principal Component Analysis (PCA) was successfully implemented, showcasing its utility in dimensionality reduction and data visualization. Hands-on experience was gained in standardizing data, computing principal components, and projecting data onto lower-dimensional subspaces. The impact of PCA on improving model efficiency and reducing overfitting was observed through its integration with machine learning algorithms. This practical enhanced understanding of PCA's role in preprocessing high-dimensional datasets and improved the ability to extract meaningful patterns while preserving essential information.

Theory:

Introduction to PCA:

Principal Component Analysis (PCA) is a statistical technique used for **dimensionality reduction**, which transforms a large set of variables into a smaller one that still contains most of the original dataset's information. It is an **unsupervised learning** method that simplifies the complexity of high-dimensional data while retaining trends and patterns.

Need for Dimensionality Reduction:

In machine learning, datasets often have hundreds or thousands of features (columns). This leads to several issues:

- Increased training time
- Difficulty in visualization
- Risk of overfitting
- Multicollinearity (correlated features causing instability)

PCA mitigates these problems by converting the data into a new coordinate system where the axes (called **principal components**) are ordered in terms of the variance they explain.

Mathematical Background:

The core idea of PCA revolves around linear algebra concepts such as **covariance matrices**, **eigenvalues**, and **eigenvectors**.

1. **Standardization:**
   The first step is to normalize the data to zero mean and unit variance since PCA is affected by scale.
2. **Covariance Matrix Computation:**
   This matrix represents how each feature correlates with others. If the dataset has n features, the covariance matrix is an n × n square matrix.
3. **Eigenvalue and Eigenvector Calculation:**
   Eigenvectors represent directions of maximum variance (principal components), and eigenvalues represent their magnitude (how much variance is explained).
4. **Feature Vector Formation:**
   By choosing the top k eigenvectors (with the highest eigenvalues), we form a feature vector to transform the data into a k-dimensional space.
5. **Projection onto New Axes:**
   Finally, the original data is projected onto the new axes formed by the principal components.

Visualization:

PCA also helps in **visualizing high-dimensional data in 2D or 3D** by projecting it onto the first few principal components, which often retain most of the original variance. This is especially useful in exploratory data analysis.

When to Use PCA?

- When dealing with datasets with a large number of variables

- When multicollinearity is present
- When you want to improve the speed and performance of ML models
- When trying to visualize complex datasets

CODE:

```
from sklearn.decomposition import PCA

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load_wine


sugar = load_wine(as_frame=True)

df = sugar.frame

print("original DataFrame Shape is", df.shape)


sugar = load_wine(as_frame=True)

df = sugar.frame

print("original DataFrame Shape is", df.shape)


X_mean = X.mean()

X_std = X.std()

Z = (X-X_mean)/X_std

c = Z.cov()

eigenvalues, eigenvectors = np.linalg.eig(c)

print("eigen values: ",eigenvalues)
```

```
idx = eigenvalues.argsort()[::-1]

eigenvalues = eigenvalues[idx]


eigenvectors = eigenvectors[:,idx]


explained_var = np.cumsum(eigenvalues)/np.sum(eigenvalues)

print(explained_var)

n_components = np.argmax(explained_var>=0.50)+1

print("\n",n_components)

u = eigenvectors[:,:n_components]

pca_compontent = pd.DataFrame(u, index=sugar['feature_names'], columns=['PC1','PC2'])

plt.figure(figsize=(10,10))

sns.heatmap(pca_compontent)

plt.title('PCA Component')

plt.show()


Z_pca = Z @ pca_compontent

Z_pca.rename({'PC1':'PCA1','PC2':'PCA2'},axis = 1, inplace = True)

print(Z_pca)

pca = PCA(n_components=2)

pca.fit(Z)

x_pca = pca.transform(Z)

df_pca1 = pd.DataFrame(x_pca, columns=['PC{}'.format(i+1) for i in range(n_components)])

print(df_pca1)


plt.figure(figsize=(10,10))

plt.scatter(x_pca[:,0],x_pca[:,1],
```

c=sugar['target'],cmap='plasma')

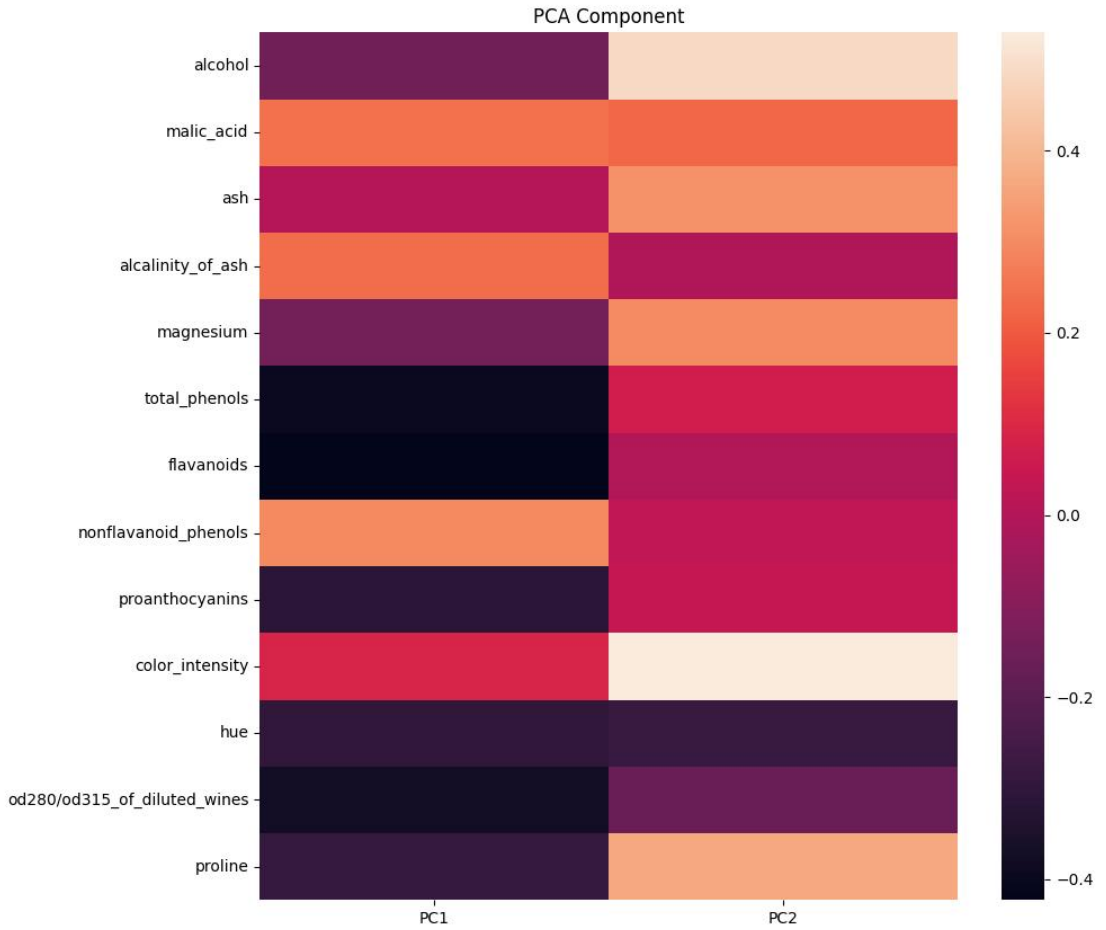
plt.xlabel('First Principal Component')

plt.ylabel('Second Principal Component')

OUTPUT:

```
original DataFrame Shape is (178, 14)
input dataframe shape:  (178, 13)
eigen values:  [4.70585025 2.49697373 1.44607197 0.91897392 0.85322818 0.64165703
 0.55102831 0.10337794 0.34849736 0.16877023 0.28887994 0.22578864
 0.25090248]
[0.36198848 0.55406338 0.66529969 0.73598999 0.80162293 0.85098116
 0.89336795 0.92017544 0.94239698 0.96169717 0.97906553 0.99204785
 1.        ]

 2
```

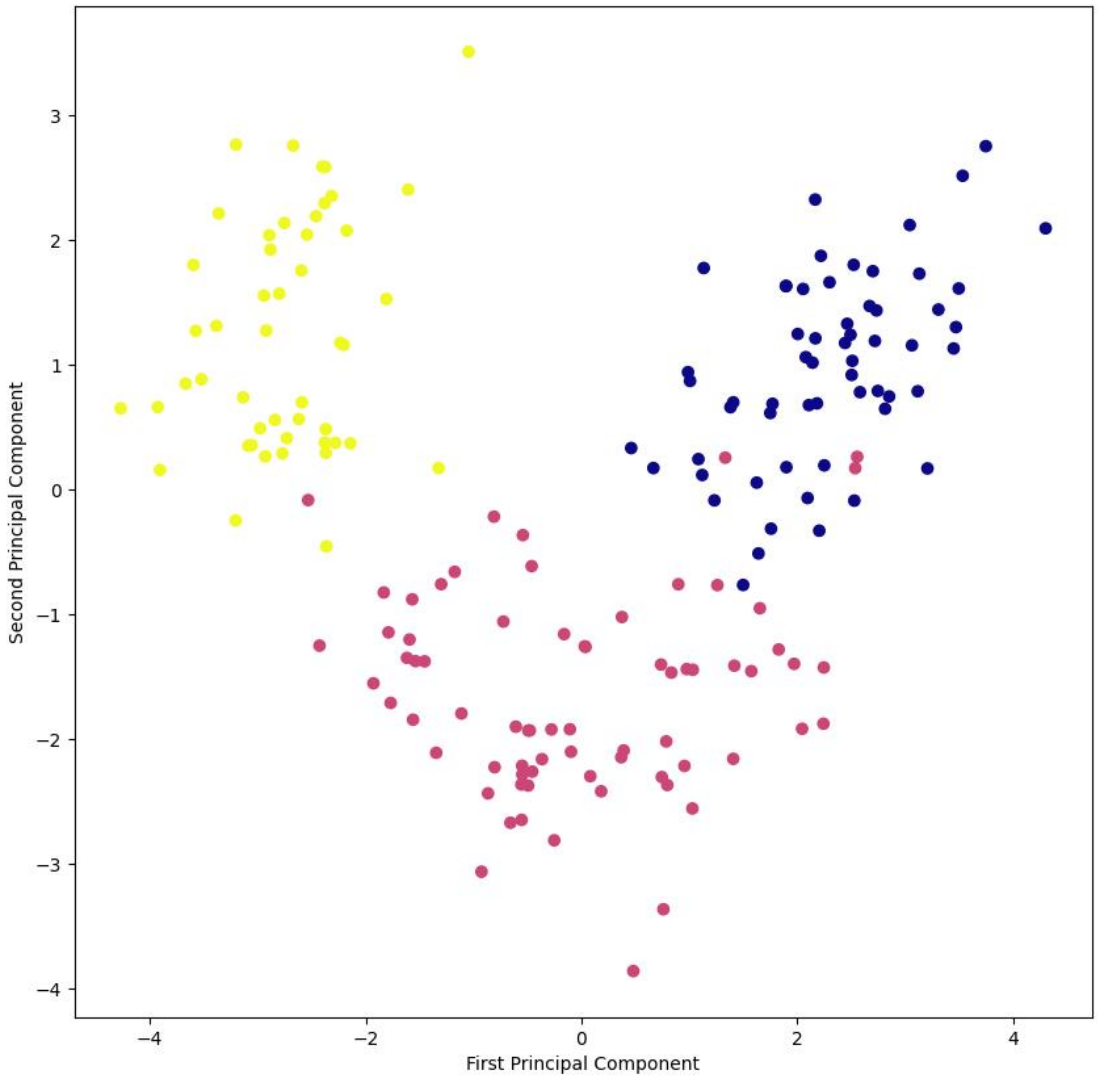


```
          PCA1      PCA2
0   -3.307421  1.439402
1   -2.203250 -0.332455
2   -2.509661  1.028251
3   -3.746497  2.748618
4   -1.006070  0.867384
..        ...       ...
173  3.361043  2.210055
174  2.594637  1.752286
175  2.670307  2.753133
176  2.380303  2.290884
177  3.199732  2.761131

[178 rows x 2 columns]
```

```
          PC1       PC2
0    3.307421  1.439402
1    2.203250 -0.332455
2    2.509661  1.028251
3    3.746497  2.748618
4    1.006070  0.867384
..        ...       ...
173 -3.361043  2.210055
174 -2.594637  1.752286
175 -2.670307  2.753133
176 -2.380303  2.290884
177 -3.199732  2.761131

[178 rows x 2 columns]
```



Conclusion:

Principal Component Analysis (PCA) was successfully implemented as a dimensionality reduction technique. The practical provided hands-on experience in standardizing data, calculating covariance matrices, extracting eigenvalues and eigenvectors, and selecting principal components to retain maximum variance.