Terna Engineering College

Department of Artificial Intelligence and Data Science

Program : Sem VI

Course: Machine Learning Lab

Experiment No.07

PART A

(PART A: TO BE REFERRED BY STUDENTS)

A.1 Aim: To implement Principal Component Analysis using Python.

A.2 Theory:

Principal Component Analysis:

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. It is one of the popular tools that is used for exploratory data analysis and predictive modelling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA algorithm is based on mathematical concepts like eigenvalues, eigenvectors, variance and covariance. It works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality.

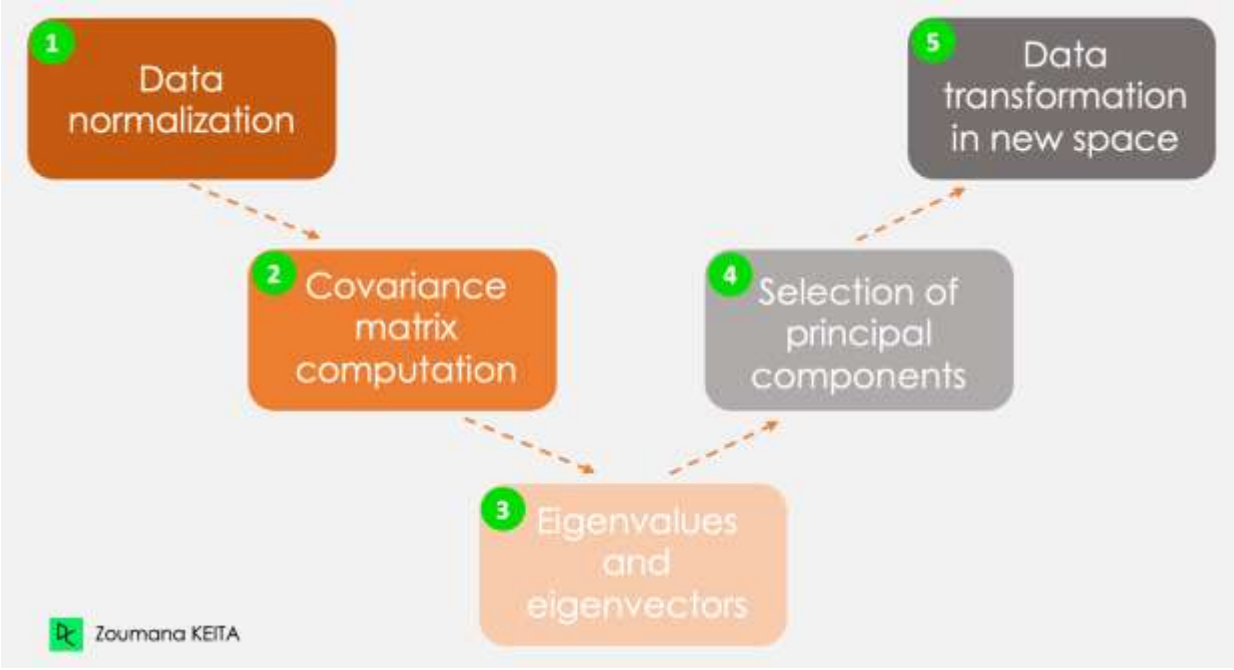
When to use PCA?

- to reduce the number of dimensions in factor analysis
- to categorize the dependent and independent variables indata
- to eliminate the noise components in dimension analysis

Uses of PCA:

- PCA techniques aid data cleaning and data preprocessing techniques.
- You can monitor multi-dimensional data (can visualize in 2D or 3D dimensions) over any platform using the Principal Component Method of factor analysis.
- PCA helps you compress the information and transmit the same using effective PCA analysis techniques. All these information processing techniques are without any loss in quality.
- This statistic is the science of analyzing different dimensions and can also be applied in several platforms like face recognition, image identification, pattern identification, and a lot more.
- PCA in machine learning technique helps in simplifying complex business algorithms
- Since Principal Component Analysis minimizes the more significant variance of dimensions, you can easily denoise the information and completely omit the noise and external factors.

Working of PCA:



1. Standardize the data: PCA requires standardized data, so the first step is to standardize the data to ensure that all variables have a mean of 0 and a standard deviation of 1.

2. Calculate the covariance matrix: The next step is to calculate the covariance matrix of the standardized data. This matrix shows how each variable is related to every other variable in the dataset.

3. Calculate the eigenvectors and eigenvalues: The eigenvectors and eigenvalues of the covariance matrix are then calculated. The eigenvectors represent the directions in which the data varies the most, while the eigenvalues represent the amount of variation along each eigenvector.

4. Choose the principal components: The principal components are the eigenvectors with the highest eigenvalues. These components represent the directions in which the data varies the most and are used to transform the original data into a lower-dimensional space.

5. Transform the data: The final step is to transform the original data into the lower-dimensional space defined by the principal components.

e.g.

Step 1. Data

We consider a dataset having n features or variables denoted by $X_1, X_2, \ldots, X_n$. Let there be N examples. Let the values of the i-th feature X_i be $X_{i1}, X_{i2}, \ldots, X_{iN}$ (see Table 4.1).

Features	Example 1	Example 2	$\cdots$	Example N
X_1	X_{11}	X_{12}	$\cdots$	X_{1N}
X_2	X_{21}	X_{22}	$\cdots$	X_{2N}
$\vdots$				
X_i	X_{i1}	X_{i2}	$\cdots$	X_{iN}
$\vdots$				
X_n	X_{n1}	X_{n2}	$\cdots$	X_{nN}

Table 4.1: Data for PCA algorithm

Step 2. Compute the means of the variables

We compute the mean $\bar{X}_i$ of the variable X_i:

$$\bar{X}_i = \frac{1}{N}(X_{i1} + X_{i2} + \cdots + X_{iN}).$$

Step 3. Calculate the covariance matrix

Consider the variables X_i and X_j (i and j need not be different). The covariance of the ordered pair (X_i, X_j) is defined as[1]

$$\text{Cov}\,(X_i, X_j) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{ik} - \bar{X}_i)(X_{jk} - \bar{X}_j). \tag{4.1}$$

We calculate the following $n \times n$ matrix S called the covariance matrix of the data. The element in the i-th row j-th column is the covariance $\text{Cov}\,(X_i, X_j)$:

$$S = \begin{bmatrix} \text{Cov}\,(X_1, X_1) & \text{Cov}\,(X_1, X_2) & \cdots & \text{Cov}\,(X_1, X_n) \\ \text{Cov}\,(X_2, X_1) & \text{Cov}\,(X_2, X_2) & \cdots & \text{Cov}\,(X_2, X_n) \\ \vdots & & & \\ \text{Cov}\,(X_n, X_1) & \text{Cov}\,(X_n, X_2) & \cdots & \text{Cov}\,(X_n, X_n) \end{bmatrix}$$

Step 4. Calculate the eigenvalues and eigenvectors of the covariance matrix

Let S be the covariance matrix and let I be the identity matrix having the same dimension as the dimension of S.

i) Set up the equation:

$$\det(S - \lambda I) = 0. \tag{4.2}$$

This is a polynomial equation of degree n in λ. It has n real roots (some of the roots may be repeated) and these roots are the eigenvalues of S. We find the n roots $\lambda_1, \lambda_2, \ldots, \lambda_n$ of Eq. (4.2).

ii) If $\lambda = \lambda'$ is an eigenvalue, then the corresponding eigenvector is a vector

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

such that

$$(S - \lambda' I)U = 0.$$

(This is a system of n homogeneous linear equations in u_1, u_2, ..., u_n and it always has a nontrivial solution.) We next find a set of n orthogonal eigenvectors $U_1, U_2, \ldots, U_n$ such that U_i is an eigenvector corresponding to λ_i.[2]

iii) We now normalise the eigenvectors. Given any vector X we normalise it by dividing X by its length. The length (or, the norm) of the vector

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

is defined as

$$\|X\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Given any eigenvector U, the corresponding normalised eigenvector is computed as

$$\frac{1}{\|U\|} U.$$

We compute the n normalised eigenvectors $e_1, e_2, \ldots, e_n$ by

$$e_i = \frac{1}{\|U_i\|} U_i, \quad i = 1, 2, \ldots, n.$$

Step 5. Derive new data set

Order the eigenvalues from highest to lowest. The unit eigenvector corresponding to the largest eigenvalue is the first principal component. The unit eigenvector corresponding to the next highest eigenvalue is the second principal component, and so on.

i) Let the eigenvalues in descending order be $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and let the corresponding unit eigenvectors be $e_1, e_2, \ldots, e_n$.

ii) Choose a positive integer p such that $1 \leq p \leq n$.

iii) Choose the eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and form the following $p \times n$ matrix (we write the eigenvectors as row vectors):

$$F = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_p^T \end{bmatrix},$$

where T in the superscript denotes the transpose.

iv) We form the following $n \times N$ matrix:

$$X = \begin{bmatrix} X_{11} - \bar{X}_1 & X_{12} - \bar{X}_1 & \cdots & X_{1N} - \bar{X}_1 \\ X_{21} - \bar{X}_2 & X_{22} - \bar{X}_2 & \cdots & X_{2N} - \bar{X}_2 \\ \vdots & & & \\ X_{n1} - \bar{X}_n & X_{n2} - \bar{X}_n & \cdots & X_{nN} - \bar{X}_n \end{bmatrix}$$

v) Next compute the matrix:

$$X_{\text{new}} = FX.$$

Note that this is a $p \times N$ matrix. This gives us a dataset of N samples having p features.

Step 6. New dataset

The matrix X_{new} is the new dataset. Each row of this matrix represents the values of a feature. Since there are only p rows, the new dataset has only features.

Step 7. Conclusion

This is how the principal component analysis helps us in dimensional reduction of the dataset. Note that it is not possible to get back the original n-dimensional dataset from the new dataset.

PARTB

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The softcopy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical incase there is no Blackboard access available)

Roll. No. A12	Name: Sufiyan Khan
Class: TE – AI & DS	Batch: A1
Date of Experiment:	Date of Submission: 24-03-24
Grade:	

B.1Input and Output:

```
Click here to ask Blackbox to help you code faster
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
```

✓ 0.0s

```
Click here to ask Blackbox to help you code faster
sugar = load_wine(as_frame = True)
df = sugar.frame
print('Original Dataframe Shape',df.shape)
```

✓ 0.0s

```
Original Dataframe Shape (178, 14)
```

```
Click here to ask Blackbox to help you code faster
X = df[sugar['feature_names']]
print('Input Dataframe Shape: ',X.shape)
```

✓ 0.0s

```
Input Dataframe Shape:  (178, 13)
```

```
Click here to ask Blackbox to help you code faster
X_mean = X.mean()
X_std = X.std()
Z = (X-X_mean)/X_std
c = Z.cov()
```

✓ 0.0s

```
Click here to ask Blackbox to help you code faster
eigenvalues, eigenvectors = np.linalg.eig(c)
print('Eigen Values: ',eigenvalues)
```

✓ 0.0s

```
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]
```

```
Click here to ask Blackbox to help you code faster
idx = eigenvalues.argsort()[::-1]
eigenvalues = eigenvalues[idx]

eigenvectors = eigenvectors[:,idx]
```

✓ 0.0s

```
Click here to ask Blackbox to help you code faster
explained_var = np.cumsum(eigenvalues)/np.sum(eigenvalues)
print(explained_var)
n_components = np.argmax(explained_var>=0.50)+1
print(n_components)
```

✓ 0.0s

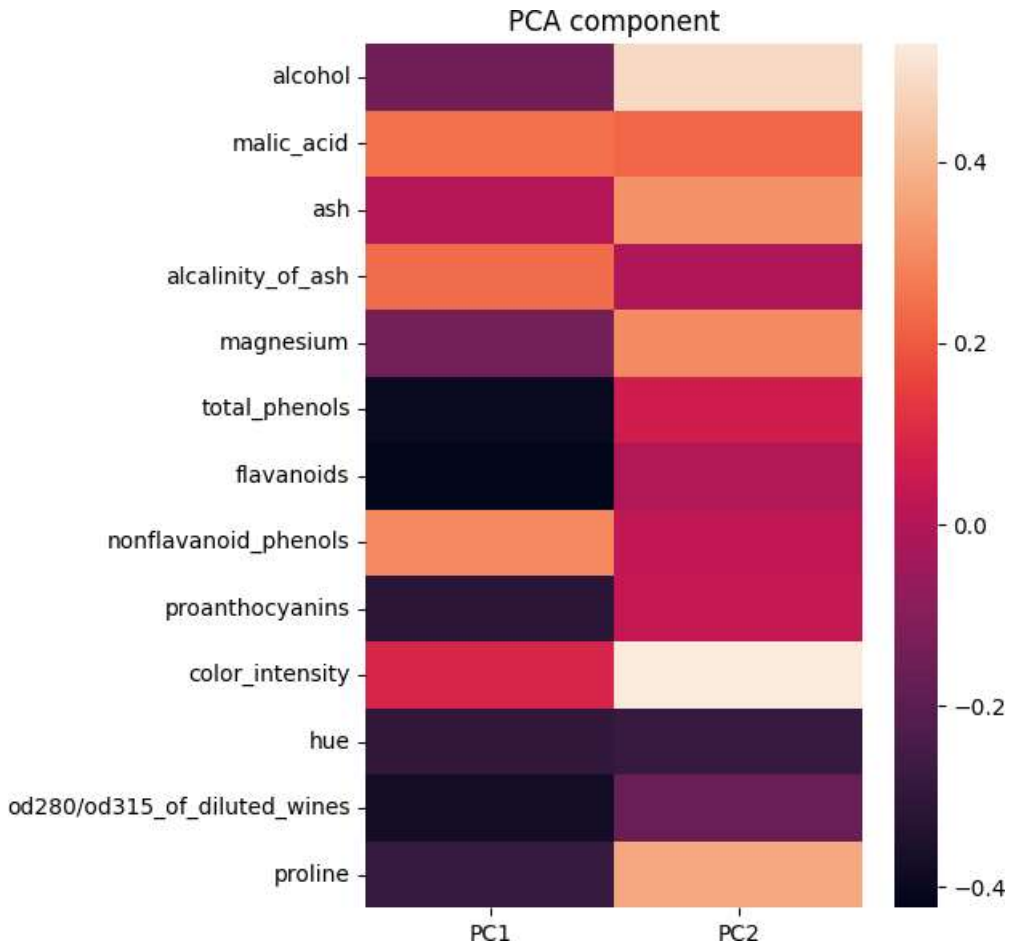
```
[0.36198848 0.55406338 0.66529969 0.73598999 0.80162293 0.85098116
 0.89336795 0.92017544 0.94239698 0.96169717 0.97906553 0.99204785
 1.        ]
2
```

```
Click here to ask Blackbox to help you code faster
u = eigenvectors[:,:n_components]
pca_component = pd.DataFrame(u,
                             index = sugar['feature_names'],
                             columns = ['PC1','PC2'])
```

✓ 0.0s

```
Click here to ask Blackbox to help you code faster
plt.figure(figsize=(5,7))
sns.heatmap(pca_component)
plt.title('PCA component')
plt.show()
```

✓ 0.1s



```
Click here to ask Blackbox to help you code faster
Z_pca = Z @ pca_component
Z_pca.rename({'PC1':'PCA1','PC2':'PCA2'}, axis=1, inplace = True)
print(Z_pca)
```

✓ 0.0s

```
        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]
```

```
Click here to ask Blackbox to help you code faster
pca = PCA(n_components=2)
pca.fit(Z)
x_pca = pca.transform(Z)
```

✓ 0.0s

```
Click here to ask Blackbox to help you code faster
df_pca1 = pd.DataFrame(x_pca,
                       columns=['PC{}'.format(i+1) for i in range(n_components)])
print(df_pca1)
```

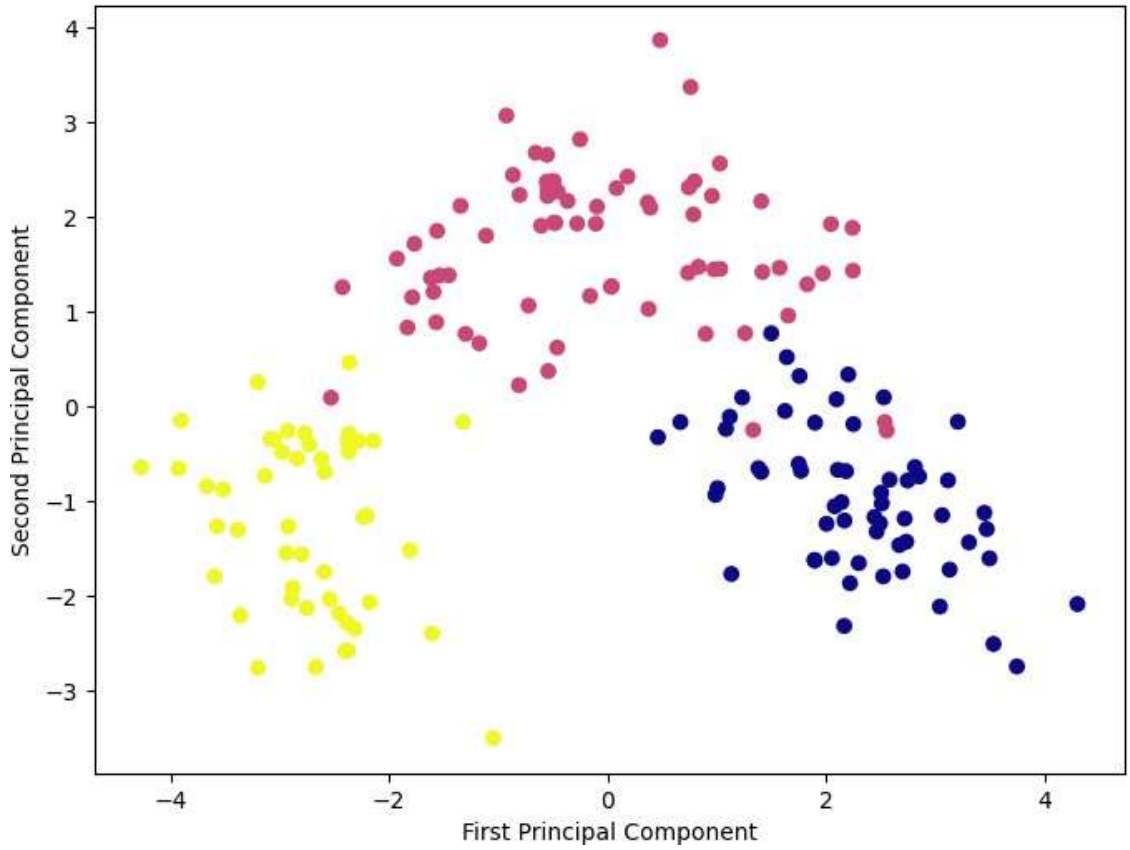
✓ 0.0s

```
          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]
```

```
Click here to ask Blackbox to help you code faster
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],
            c=sugar['target'],cmap='plasma')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.show()
```

✓ 0.0s



B.2 Conclusion:

Thus we have successfully implemented Principal Component Analysis in Python and understood how we have performed dimensionality reduction and enhanced model performance by extracting the principal components.