EXPERIMENT NO. 7

Aim: Implement Principal Component Analysis.

Theory:

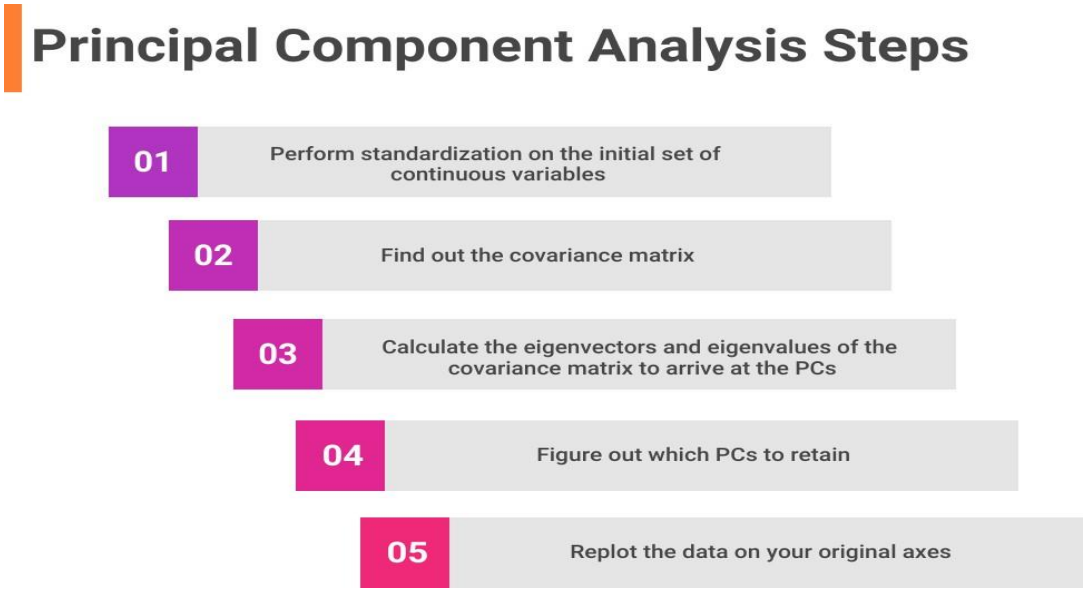
Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction and data visualization. It aims to transform high-dimensional data into a lower-dimensional space while preserving as much variance as possible. PCA identifies the directions (principal components) that capture the maximum variability in the data and projects the data onto these components.

Basic Concepts

- *Variance*: In PCA, the goal is to capture as much variance in the data as possible. Variance represents the spread or dispersion of data points around the mean. PCA seeks to find the directions in which the data varies the most.

- *Principal Components*: Principal components are the orthogonal axes in the feature space that capture the maximum variance in the data. The first principal component (PC1) accounts for the largest variance, followed by the second principal component (PC2), and so on. Each principal component is a linear combination of the original features.

- *Dimensionality Reduction*: By selecting a subset of the principal components that explain most of the variance in the data, PCA reduces the dimensionality of the dataset. This reduces the computational complexity of subsequent analyses and visualization while retaining the essential information in the data.



Steps of PCA

PCA involves the following steps:

- *Standardization*: Standardize the features by subtracting the mean and dividing by the standard deviation. This ensures that all features have the same scale, preventing features with larger magnitudes from dominating the analysis.

- *Compute Covariance Matrix*: Calculate the covariance matrix of the standardized data. The covariance matrix measures the pairwise relationships between features and provides information about how the features vary together.

- *Eigenvalue Decomposition*: Perform eigenvalue decomposition on the covariance matrix to find its eigenvectors and eigenvalues. The eigenvectors represent the directions (principal components), and the eigenvalues represent the amount of variance explained by each principal component.

- *Select Principal Components*: Sort the eigenvalues in descending order and choose the top k eigenvectors (principal components) corresponding

to the highest eigenvalues. These principal components capture most of the variance in the data.

- *Project Data*: Project the original data onto the selected principal components to obtain the lower-dimensional representation of the data.

Applications of PCA

- *Dimensionality Reduction*: PCA is commonly used to reduce the dimensionality of high-dimensional datasets in machine learning tasks such as classification, regression, and clustering.

- *Data Visualization*: PCA helps visualize high-dimensional data in lower-dimensional space, enabling insights into the underlying structure and patterns in the data.

- *Feature Engineering*: PCA can be used for feature extraction and feature engineering to create new features that capture the most important information in the data.

- *Noise Reduction*: PCA can help remove noise and redundant information from the data, improving the performance of subsequent analyses.

Principal Component Analysis (PCA) is a fundamental technique in data analysis and machine learning for reducing the dimensionality of high-dimensional datasets while retaining essential information. By identifying the directions of maximum variance in the data, PCA enables efficient data visualization, feature extraction, and noise reduction. Understanding the theory behind PCA is essential for effectively applying it to real-world datasets and gaining insights into complex data structures.

Program Code:

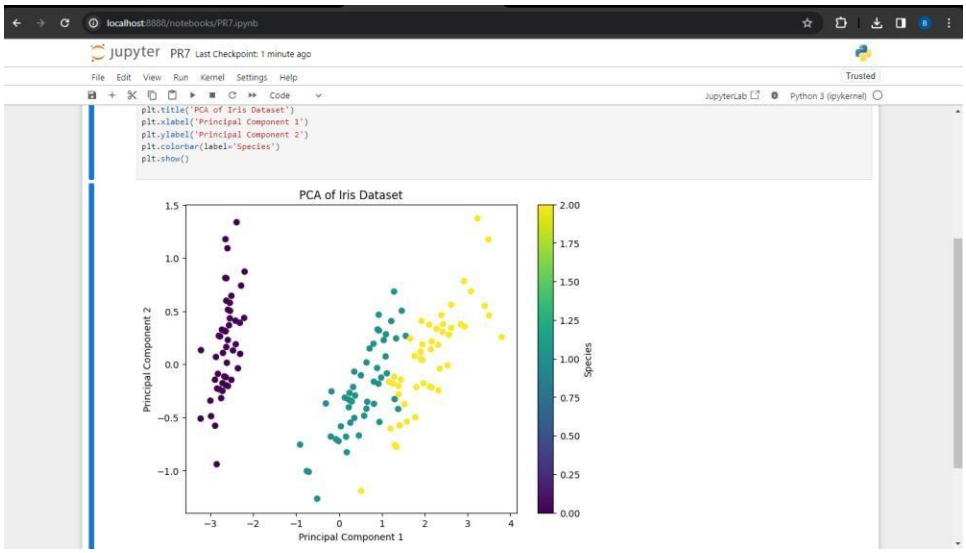
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris

# Load the Iris dataset (example dataset from scikit-learn)
iris = load_iris()
X = iris.data
y = iris.target

# Perform PCA
pca = PCA(n_components=2)  # Reduce the data to 2 dimensions
X_pca = pca.fit_transform(X)

# Visualize the transformed data
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis')
plt.title('PCA of Iris Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Species')
plt.show()
```

Output:



Conclusion:

By performing this practical we learnt to implement Principal Component Analysis in python.