**Machine Learning Experiment 11**

**Title: Implement Dimensionality reduction using Principle Component Analysis (PCA) method.**

**Objective:** The objective of this lab assignment is to implement Dimensionality Reduction using Principal Component Analysis (PCA) and gain hands-on experience in reducing the dimensionality of a dataset while preserving its essential information.

**Tasks:**

**1. Implement the PCA algorithm from scratch or using scikit-learn.**

**Code:**

# Importing necessary libraries

import pandas as pd

import numpy as np

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Load Titanic dataset from seaborn

import seaborn as sns

titanic = sns.load\_dataset('titanic')

# Drop columns with too many missing values or that are irrelevant for analysis

titanic = titanic.drop(columns=['embarked', 'who', 'deck', 'embark\_town', 'alive', 'class'])

# Fill missing values in 'age' and 'fare' with their means using an updated method to avoid warnings

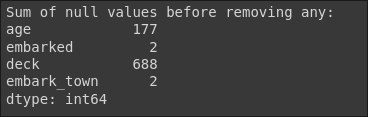
titanic['age'] = titanic['age'].fillna(titanic['age'].mean())

titanic['fare'] = titanic['fare'].fillna(titanic['fare'].mean())

# One-hot encode categorical variables

titanic = pd.get\_dummies(titanic, drop\_first=True)

**Output:**

**

**2. Perform data standardization (mean-centering and scaling) as a preprocessing step for PCA.**

**Code:**

# Separate features (X) and target (y)

X = titanic.drop(columns=['survived'])

y = titanic['survived']

# Standardize the data (mean-centering and scaling)

scaler = StandardScaler()

titanic\_scaled = scaler.fit\_transform(X)

# Check the shape of the scaled dataset

print(f"Shape of the dataset after standardization: {titanic\_scaled.shape}")

**Output:**

****

**3. Determine the number of principal components to retain.**

**Code:**

# Perform PCA and compute explained variance

pca = PCA()

pca.fit(titanic\_scaled)

# Cumulative explained variance

cumulative\_variance = np.cumsum(pca.explained\_variance\_ratio\_)

# Plot cumulative explained variance

plt.figure(figsize=(8, 5))

plt.plot(range(1, len(cumulative\_variance) + 1), cumulative\_variance, marker='o', linestyle='--', color='r')

plt.axhline(y=0.95, color='b', linestyle='--') # Line indicating 95% variance

plt.title('Cumulative Explained Variance by Principal Components')

plt.xlabel('Number of Principal Components')

plt.ylabel('Cumulative Explained Variance')

plt.grid(True)

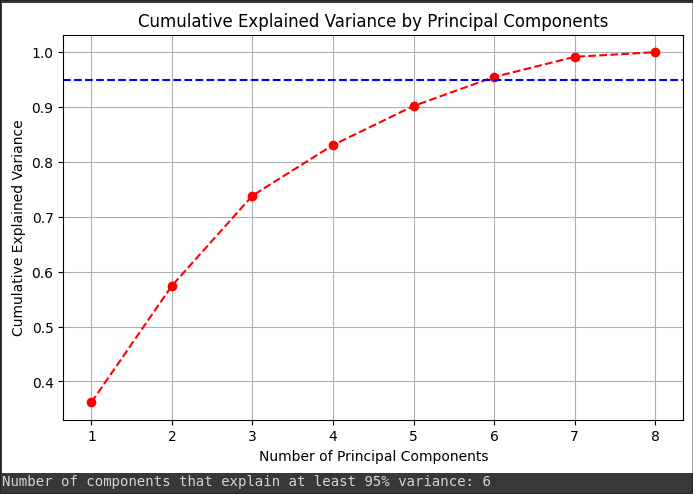
plt.show()

# Select the number of components that explain at least 95% variance

n\_components\_95 = np.argmax(cumulative\_variance >= 0.95) + 1

print(f"Number of components that explain at least 95% variance: {n\_components\_95}")

**Output:**

****

**4. Apply PCA to the preprocessed dataset and reduce its dimensionality.**

**Code:**

# Apply PCA with the selected number of components

pca\_95 = PCA(n\_components=n\_components\_95)

titanic\_reduced\_95 = pca\_95.fit\_transform(titanic\_scaled)

# Check the shape of the reduced dataset

print(f"Shape of the dataset after PCA (95% variance): {titanic\_reduced\_95.shape}")

**Output:**



**5. Visualize the dataset before and after PCA using scatterplots or other appropriate visualizations.**

**Code:**

# Visualization after PCA (for the first 2 components)

plt.figure(figsize=(8, 5))

plt.scatter(titanic\_reduced\_95[:, 0], titanic\_reduced\_95[:, 1], c=y, cmap='viridis')

plt.title('Data after PCA (First two components)')

plt.xlabel('Principal Component 1')

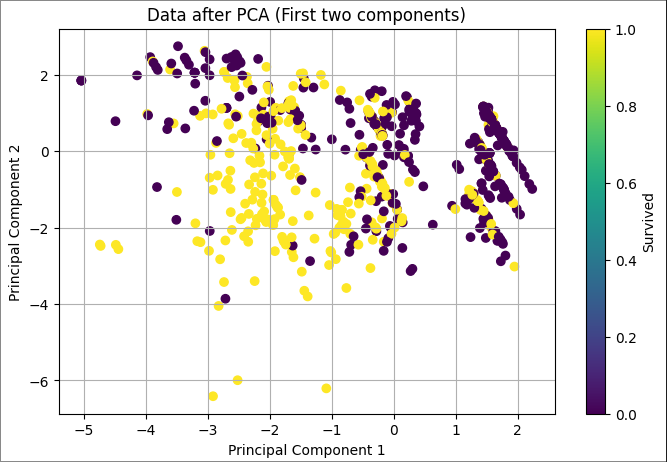
plt.ylabel('Principal Component 2')

plt.colorbar(label='Survived')

plt.grid(True)

plt.show()

**Output:**

****

**6. Evaluate the impact of dimensionality reduction on the dataset's performance in a machine learning task (e.g., classification or regression).**

**Code:**

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(titanic\_scaled, y, test\_size=0.2, random\_state=42)

# Logistic Regression without PCA (original data)

clf = LogisticRegression()

clf.fit(X\_train, y\_train)

y\_pred = clf.predict(X\_test)

accuracy\_before\_pca = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy before PCA: {accuracy\_before\_pca:.4f}")

# Logistic Regression after PCA (with 95% variance retained)

X\_train\_pca, X\_test\_pca, y\_train\_pca, y\_test\_pca = train\_test\_split(titanic\_reduced\_95, y, test\_size=0.2, random\_state=42)

clf\_pca = LogisticRegression()

clf\_pca.fit(X\_train\_pca, y\_train\_pca)

y\_pred\_pca = clf\_pca.predict(X\_test\_pca)

accuracy\_after\_pca = accuracy\_score(y\_test\_pca, y\_pred\_pca)

print(f"Accuracy after PCA (95% variance): {accuracy\_after\_pca:.4f}")

**Output:**

****

**7. Suggest possible use cases where PCA can be beneficial.**

PCA is widely used in cases where the following is required:

**Data compression:** Reducing the dimensionality while retaining most of the information.

**Noise reduction:** By discarding low-variance components, noise can be reduced.

**Visualization:** PCA helps visualize high-dimensional data in 2D or 3D.

**Speeding up machine learning models:** Reducing the number of features can speed up training and inference.

Use cases include:

**Image compression and recognition:** PCA is used in facial recognition and image processing.

**Genomics:** PCA helps reduce dimensionality in gene expression datasets.

**Finance:** PCA is used to reduce complexity in stock market datasets.