NAME: Shashwat Shah

SAPID:60004220126

C22

B DIV

CASE 1: FROM SCRATCH ON GIVEN DATASET

```
import numpy as np

class PCA:

    def __init__(self, num_components):
        self.num_components = num_components
        self.components     = None
        self.mean           = None
        self.variance_share = None


    def fit(self, X):

        # 1. Centered Data
        print("Centered Data:")
        self.mean = np.mean(X, axis=0)
        centered_data = X - self.mean
        print(centered_data)

        # data centering
        X = X.astype(np.float64)
        X -= self.mean

        # calculate eigenvalues & vectors
        cov_matrix= np.cov(X.T)

        # 2. Covariance Matrix
        print("\nCovariance Matrix:")
        print(cov_matrix)

        values, vectors = np.linalg.eig(cov_matrix)

        # 3. Eigen Values
        print("\nEigen Values:")
        print(values)

        # 4. Eigen Vectors
        print("\nEigen Vectors:")
        print(vectors)
```

```python
        # sort eigenvalues & vectors
        sort_idx = np.argsort(values)[::-1]
        values   = values[sort_idx]
        vectors  = vectors[:, sort_idx]

        # store principal components & variance
        self.components = vectors[:self.num_components]
        self.variance_share = np.sum(values[:self.num_components]) / np.sum(values)

        # 5. New Values (Principal Components)
        print("\nNew Values (Principal Components):")
        print(self.components)


    def transform(self, X):

        # data centering
        X -= self.mean

        # decomposition
        return np.dot(X, self.components.T)

X = np.array([[4, 6],
              [8, 2],
              [13, 3],
              [7, 15]])

# Instantiate PCA with 2 components and fit to data
pca = PCA(num_components=2)
pca.fit(X)
```

OUTPUT:

```
Centered Data:
[[-4.  -0.5]
 [ 0.  -4.5]
 [ 5.  -3.5]
 [-1.   8.5]]

Covariance Matrix:
[[14. -8.]
 [-8. 35.]]

Eigen Values:
[11.29962122 37.70037878]

Eigen Vectors:
[[-0.94747869  0.31981892]
 [-0.31981892 -0.94747869]]

New Values (Principal Components):
[[ 0.31981892 -0.94747869]
 [-0.94747869 -0.31981892]]
```

CASE 2 : DATASET OF YOUR CHOICE USING LIBRARIES

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Load the dataset
df = pd.read_csv("Automobile_data.csv")

# Drop rows with missing values if any
df = df.dropna()

# Select numerical features for PCA
numerical_features = df.select_dtypes(include=['float64', 'int64'])

# Preprocess the data by scaling numerical features
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numerical_features)

# 1. Centered Data
print("Centered Data:")
centered_data = scaled_data - scaled_data.mean(axis=0)
print(centered_data)

# Calculate covariance matrix
cov_matrix = np.cov(centered_data.T)

# 2. Covariance Matrix
print("\nCovariance Matrix:")
print(cov_matrix)

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

# 3. Eigen Values
print("\nEigen Values:")
print(eigenvalues)
```

```
# 3. Eigen Values
print("\nEigen Values:")
print(eigenvalues)

# 4. Eigen Vectors
print("\nEigen Vectors:")
print(eigenvectors)

# Apply PCA
pca = PCA()
pca.fit(scaled_data)

# 5. New Values (Principal Components)
print("\nNew Values (Principal Components):")
print(pca.components_)
```

OUTPUT:

```
Centered Data:
[[ 1.74347043 -1.6907718  -0.42652147 ... -0.28834891 -0.64655303
  -0.54605874]
 [ 1.74347043 -1.6907718  -0.42652147 ... -0.28834891 -0.64655303
  -0.54605874]
 [ 0.133509   -0.70859588 -0.23151305 ... -0.28834891 -0.95301169
  -0.69162706]
 ...
 [-1.47645244  1.72187336  1.19854871 ... -0.33882413 -1.10624102
  -1.12833203]
 [-1.47645244  1.72187336  1.19854871 ...  3.24491627  0.11959362
  -0.54605874]
 [-1.47645244  1.72187336  1.19854871 ... -0.16216087 -0.95301169
  -0.83719538]]

Covariance Matrix:
[[ 1.00490196 -0.5345613  -0.35936452 -0.23406082 -0.54369035 -0.22880672
  -0.10630829 -0.17939016 -0.03599823  0.03477564]
 [-0.5345613   1.00490196  0.87887467  0.79904141  0.59232415  0.78019214
   0.57211951  0.25101029 -0.47271956 -0.54674899]
 [-0.35936452  0.87887467  1.00490196  0.8452414   0.49343646  0.88203105
   0.68670968  0.15919024 -0.67419743 -0.70811583]
 [-0.23406082  0.79904141  0.8452414   1.00490196  0.280579    0.87128262
   0.73903847  0.18201651 -0.64585485 -0.68053761]
 [-0.54369035  0.59232415  0.49343646  0.280579    1.00490196  0.29702061
   0.0674779   0.26249469 -0.04887806 -0.10788389]
 [-0.22880672  0.78019214  0.88203105  0.87128262  0.29702061  1.00490196
   0.85476365  0.15210371 -0.7611266  -0.80137393]
 [-0.10630829  0.57211951  0.68670968  0.73903847  0.0674779   0.85476365
   1.00490196  0.02911338 -0.65686212 -0.68079084]
 [-0.17939016  0.25101029  0.15919024  0.18201651  0.26249469  0.15210371
   0.02911338  1.00490196  0.3262931   0.2665014 ]
 [-0.03599823 -0.47271956 -0.67419743 -0.64585485 -0.04887806 -0.7611266
  -0.65686212  0.3262931   1.00490196  0.9760985 ]
 [ 0.03477564 -0.54674899 -0.70811583 -0.68053761 -0.10788389 -0.80137393
  -0.68079084  0.2665014   0.9760985   1.00490196]]
```

```
Eigen Values:
[5.66327865 2.06393304 0.97359585 0.51023591 0.28809477 0.25971235
 0.12508537 0.09029827 0.02196028 0.05282511]

Eigen Vectors:
[[-0.1406803   0.46938248  0.43416281 -0.65753407  0.26084403 -0.20958733
   0.12454723 -0.10570576  0.01452664  0.01332464]
 [ 0.36466184 -0.26203402 -0.0569007  -0.00350329 -0.03796216 -0.41729053
   0.44102192 -0.6343559  -0.13509842 -0.06616736]
 [ 0.3965888  -0.09774234 -0.00607855 -0.13489422 -0.02988667 -0.19022404
   0.47207053  0.71099134  0.14661516 -0.16732211]
 [ 0.38057618 -0.01365742  0.20388087  0.05735174 -0.03937346 -0.55393849
  -0.69811788  0.08754105  0.03470707 -0.07405338]
 [ 0.17202759 -0.47948861 -0.35515743 -0.58511996  0.38055304  0.23974043
  -0.26151004 -0.02579491 -0.00931975 -0.02364729]
 [ 0.40393127  0.0424753   0.16504009  0.0425094   0.06392698  0.17204012
   0.03911469  0.03008734 -0.05454601  0.87572551]
 [ 0.34194007  0.15915237  0.24957784  0.37866195  0.65791046  0.31137929
   0.02860932 -0.08200702 -0.00296087 -0.33870031]
 [ 0.03509208 -0.45916395  0.70716925 -0.11882868 -0.35717351  0.3417246
  -0.01595892 -0.04417378 -0.02988909 -0.16211238]
```

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance_ratio = np.cumsum(explained_variance_ratio)
plt.bar(range(1, len(explained_variance_ratio) + 1), explained_variance_ratio, alpha=0.5, align='center', label='Individual explained variance')
plt.step(range(1, len(cumulative_variance_ratio) + 1), cumulative_variance_ratio, where='mid', label='Cumulative explained variance')
plt.xlabel('Principal components')
plt.ylabel('Explained variance ratio')
plt.title('Explained Variance Ratio')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

OUTPUT: