Data Mining Assighment 1

Nitin Sharma (22CS01027)

Task: To find the principal components for the sample data of 10 entities with 5 attributes.

```
import pandas as pd
```

```
import numpy as np
```

Used the .cve file for data access. pandas convert it intoo to a data frame.

```
df=pd.read_csv('/content/dm_2.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Horsepower           10 non-null     int64
 1   Engine_Size_L        10 non-null     float64
 2   Fuel_Efficiency_MPG  10 non-null     int64
 3   Weight_kg            10 non-null     int64
 4   Price_1000s          10 non-null     int64
dtypes: float64(1), int64(4)
memory usage: 532.0 bytes
```

10 x 5 dataset

```
print(df)
```

```
   Horsepower  Engine_Size_L  Fuel_Efficiency_MPG  Weight_kg  Price_1000s
0         132            1.8                   30       1250           20
1         158            2.0                   32       1300           22
2         160            2.0                   28       1400           25
3         255            3.0                   25       1550           42
4         261            2.9                   24       1600           45
5         283            0.0                  130       1610           48
6         188            2.5                   29       1450           27
7         147            2.0                   33       1275           21
8         160            1.5                   27       1500           26
9         255            2.0                   26       1650           46
```

X will consists only values rather than name of attributes and row number, just like a 2D array.

```
X=df.values
```

```
print("Original Data: \n", X[:])
```

```
Original Data:
 [[1.320e+02 1.800e+00 3.000e+01 1.250e+03 2.000e+01]
 [1.580e+02 2.000e+00 3.200e+01 1.300e+03 2.200e+01]
 [1.600e+02 2.000e+00 2.800e+01 1.400e+03 2.500e+01]
 [2.550e+02 3.000e+00 2.500e+01 1.550e+03 4.200e+01]
 [2.610e+02 2.900e+00 2.400e+01 1.600e+03 4.500e+01]
 [2.830e+02 0.000e+00 1.300e+02 1.610e+03 4.800e+01]
 [1.880e+02 2.500e+00 2.900e+01 1.450e+03 2.700e+01]
 [1.470e+02 2.000e+00 3.300e+01 1.275e+03 2.100e+01]
 [1.600e+02 1.500e+00 2.700e+01 1.500e+03 2.600e+01]
 [2.550e+02 2.000e+00 2.600e+01 1.650e+03 4.600e+01]]
```

Calulating the mean of each attribute.

```
mean_vec = np.mean(X, axis=0)
print("Mean Vector: \n", mean_vec)
```

```
Mean Vector:
 [ 199.9     1.97   38.4  1458.5    32.2 ]
```

Center the data by subtracting the mean of each attribute.

```
X_centered = X - mean_vec
print("Centered Data: \n", X_centered[:])
```

```
Centered Data:
 [[-6.790e+01 -1.700e-01 -8.400e+00 -2.085e+02 -1.220e+01]
 [-4.190e+01  3.000e-02 -6.400e+00 -1.585e+02 -1.020e+01]
 [-3.990e+01  3.000e-02 -1.040e+01 -5.850e+01 -7.200e+00]
 [ 5.510e+01  1.030e+00 -1.340e+01  9.150e+01  9.800e+00]
 [ 6.110e+01  9.300e-01 -1.440e+01  1.415e+02  1.280e+01]
 [ 8.310e+01 -1.970e+00  9.160e+01  1.515e+02  1.580e+01]
 [-1.190e+01  5.300e-01 -9.400e+00 -8.500e+00 -5.200e+00]
 [-5.290e+01  3.000e-02 -5.400e+00 -1.835e+02 -1.120e+01]
 [-3.990e+01 -4.700e-01 -1.140e+01  4.150e+01 -6.200e+00]
 [ 5.510e+01  3.000e-02 -1.240e+01  1.915e+02  1.380e+01]]
```

used numpy function to find the covarience matrix.

```
cov_matrix = np.cov(X_centered, rowvar=False)
print("Covariance Matrix: \n", cov_matrix)
```

```
Covariance Matrix:
 [[ 3.24454444e+03 -3.17000000e+00  8.24044444e+02  7.56816667e+03
   6.48911111e+02]
 [-3.17000000e+00  7.04555556e-01 -2.29866667e+01 -7.49444444e+00
  -8.15555556e-01]
 [ 8.24044444e+02 -2.29866667e+01  1.04426667e+03  1.35677778e+03
   1.53688889e+02]
 [ 7.56816667e+03 -7.49444444e+00  1.35677778e+03  2.17225000e+04
   1.57422222e+03]
 [ 6.48911111e+02 -8.15555556e-01  1.53688889e+02  1.57422222e+03
   1.32844444e+02]]
```

linalg.eigh function from numpy to find eigenvalus and eigenvectors

```
eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
print("Eigenvalues: \n", eigenvalues)
print("Eigenvectors: \n", eigenvectors)
```

```
Eigenvalues:
 [1.85786074e-02 2.12380149e+00 3.71492386e+02 1.12280658e+03
 2.46484188e+04]
Eigenvectors:
 [[ 3.61093972e-02  1.62459693e-01  8.17807134e-01  4.36609023e-01
  -3.35956859e-01]
 [-9.91588100e-01  1.25024355e-01  2.85000639e-02 -1.75938430e-02
   3.91998856e-04]
 [-2.95284413e-02 -8.68114164e-05 -4.89689597e-01  8.68890620e-01
  -6.60397842e-02]
 [-2.33028854e-03  1.43715781e-02 -2.68745536e-01 -2.22752880e-01
  -9.36987203e-01]
 [-1.20713679e-01 -9.78656839e-01  1.35496025e-01  6.68824520e-02
  -6.94734741e-02]]
```

```
sorted_indices = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[sorted_indices]
eigenvectors = eigenvectors[:, sorted_indices]
```

```
print(sorted_indices)
```

```
[4 3 2 1 0]
```

```
top_eigenvectors = eigenvectors[:, :2]
print("Top 2 Pricipal Components: \n", top_eigenvectors)
```

```
Top 2 Pricipal Components:
 [[-3.35956859e-01  4.36609023e-01]
 [ 3.91998856e-04 -1.75938430e-02]
 [-6.60397842e-02  8.68890620e-01]
 [-9.36987203e-01 -2.22752880e-01]
 [-6.94734741e-02  6.68824520e-02]]
```

Doing matrix multiplication X_centered(10x5) with top_eigenvectors(5x2) to get X_pca(10x2)

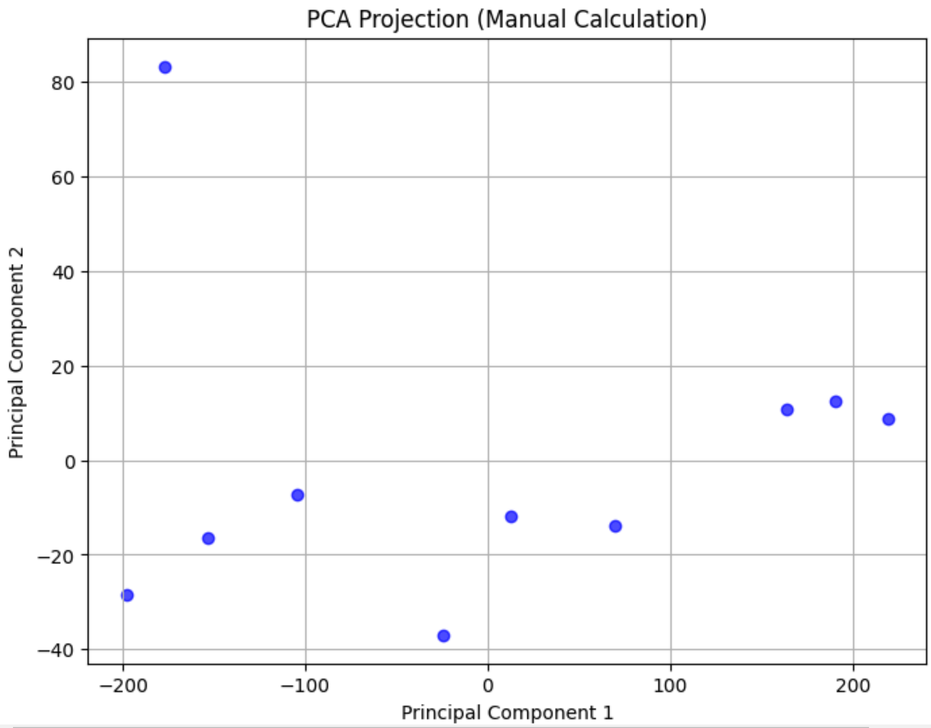
```
X_pca = X_centered @ top_eigenvectors
df_pca = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
print("Transformed Data: \n",df_pca)
```

```
Transformed Data:
          PC1        PC2
0  219.575546   8.686567
1  163.720360  10.768785
2   69.405465 -13.908200
3 -104.041055  -7.330539
4 -153.048576 -16.515013
5 -177.019274  83.216932
6   12.944522 -11.826933
7  190.843999  12.336916
8  -24.296885 -36.976700
9 -198.084101 -28.451813
```

Additionally ploting the data points taking principal components as axis.

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(8, 6))
plt.scatter(df_pca['PC1'], df_pca['PC2'], color='blue', alpha=0.7)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Projection (Manual Calculation)')
plt.grid()
plt.show()
```




```
Start coding or generate with AI.
```