PCA Implementation

Data

Here is the set-up code to load in the data and import dependencies:

```
import numpy as np
from sklearn.preprocessing import StandardScaler

data = np.array([[0.2, -0.3], [-1.1, 2], [1, -2.2], [0.5, -1], [-0.6, 1]])
num_components = 1
print("data:\n", data)
```

Output:

```
[[ 0.2 -0.3]
 [-1.1  2. ]
 [ 1.  -2.2]
 [ 0.5 -1. ]
 [-0.6  1. ]]
```

Scaled data

The data is scaled using a standard scaler:

```
scaler = StandardScaler()
scaler.fit(data)
scaled_data = scaler.transform(data)
print("\nscaled:\n", scaled_data)
```

Output:

```
[[ 0.26444294 -0.13558154]
 [-1.45443618  1.42360613]
 [ 1.32221471 -1.42360613]
 [ 0.66110736 -0.61011691]
 [-0.79332883  0.74569845]]
```

Covariance Matrix

The covariance matrix of the scaled data is calculated as follows:

```
covariance_matrix = np.cov(scaled_data.T)
print("\ncovariance matrix:\n", covariance_matrix)
```

Output:

```
[[ 1.25        -1.24591192]
 [-1.24591192  1.25      ]]
```

Eigenvalues and Eigenvectors

The eigenvalues and eigenvectors of the covariance matrix are calculated as follows:

```
eigenvalues, eigenvectors = np.linalg.eig(covariance_matrix)
print("\neigenvalues:\n", eigenvalues)
print("\neigenvectors:\n", eigenvectors)
```

Output:

```
eigenvalues: [2.49591192 0.00408808]
eigenvectors: [[ 0.70710678  0.70710678]
 [-0.70710678  0.70710678]]
```

Proportion of variance

The proportion of variance is calculated as follows:

```
print("\nProportion of variance:\n", sum(eigenvalues[:num_components]) /
sum(eigenvalues))
```

Output:

```
0.99836476739648
```

Since the proportion of variance is so close to `1`, the information lost is likely negligible.

Projection Matrix

The projection matrix is formed from the eigenvectors:

```
projection_matrix = (eigenvectors.T[:][:num_components]).T
print("\nprojection matrix:\n", projection_matrix)
```

Output:

```
[[ 0.70710678]
 [-0.70710678]]
```

PCA Data

The data after applying PCA is:

```
data_pca = scaled_data.dot(projection_matrix)
print("\ndata_pca:\n", data_pca)
```

Output:

```
[[ 0.28286002]
 [-2.03508324]
 [ 1.94158854]
 [ 0.8988913 ]
 [-1.08825662]]
```