PRACTICAL 6

Name:	Harsh Shah	**Semester:**	VII	**Division:**	6
Roll No.:	21BCP359	**Date:**	03-09-24	**Batch:**	G11
Aim:	Understanding Principal Component Analysis in Datasets.				

1. **Using sklearn library import the digits datasets (i.e. through load_digits()). It is a 8 x 8 pixel images dataset and they are 64-dimensional. In order to retrieve some of the intuition behind the relationship between these points, make use of PCA to reduce the dimension to a manageable number of dimensions (i.e., 2). After reducing the dimension plot them using scatter plots with cmap.**

2. **How many number of components will be ideal is an important part of using PCA. Using this data plot the cumulative explained variance ratio as a function of the number of components.**

3. **Try to reconstruct the data using the largest subset of principal components. The idea behind this is that any components with variance much larger than the effect of the noise should be relatively unaffected by the noise. Add some random noise to the dataset and replot it.**

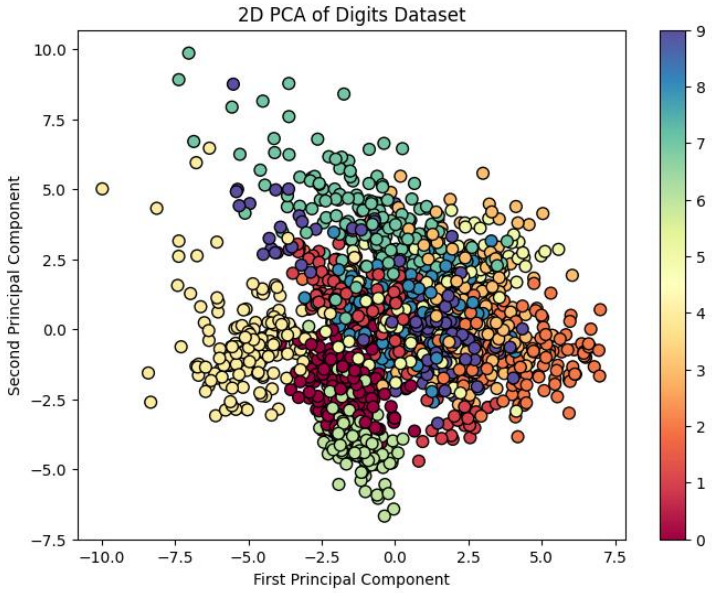
Code

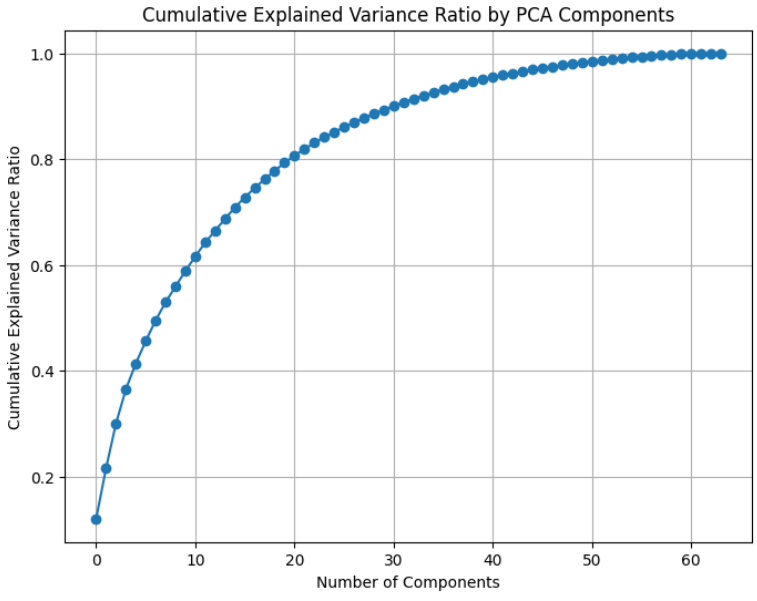
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Load the digits dataset
digits = load_digits()
X = digits.data  # Feature matrix
y = digits.target  # Labels

# Standardize the data (PCA works better on standardized data)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA to reduce the dimensionality to 2 dimensions
pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X_scaled)
```

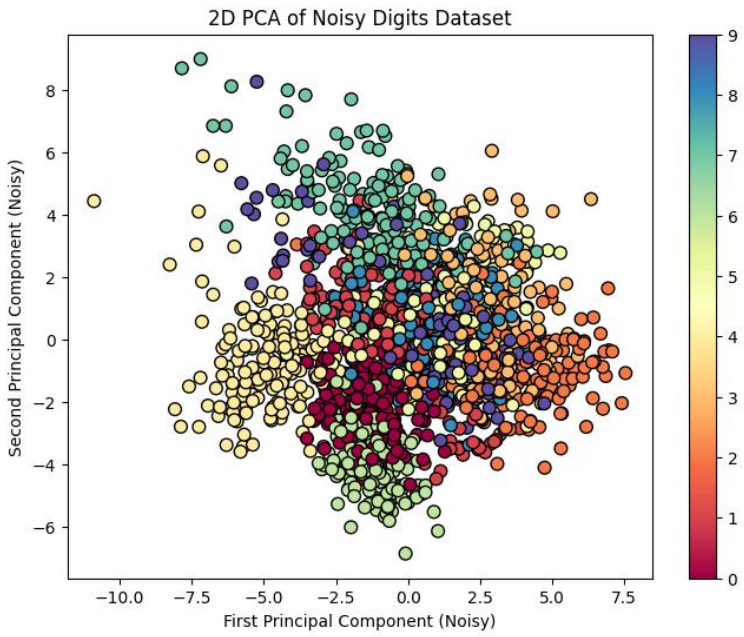
```
# Scatter plot of the 2D PCA-reduced data
plt.figure(figsize=(8, 6))
scatter = plt.scatter(X_pca_2d[:, 0], X_pca_2d[:, 1], c=y, cmap='Spectral', edgecolor='k', s=60)
plt.colorbar(scatter)
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.title('2D PCA of Digits Dataset')
plt.show()

# Plot the cumulative explained variance as a function of the number of components
pca_full = PCA().fit(X_scaled)
plt.figure(figsize=(8, 6))
plt.plot(np.cumsum(pca_full.explained_variance_ratio_), marker='o')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.title('Cumulative Explained Variance Ratio by PCA Components')
plt.grid(True)
plt.show()

# Reconstruct the data using the largest subset of principal components
n_components = 30  # Select the top 30 components
pca_reconstruct = PCA(n_components=n_components)
X_pca_reduced = pca_reconstruct.fit_transform(X_scaled)
X_reconstructed = pca_reconstruct.inverse_transform(X_pca_reduced)

# Add noise to the dataset
noise_factor = 0.5
X_noisy = X_scaled + noise_factor * np.random.normal(size=X_scaled.shape)

# Apply PCA again to the noisy data
X_pca_noisy = pca_2d.fit_transform(X_noisy)
```

```
# Scatter plot of noisy PCA-reduced data
plt.figure(figsize=(8, 6))
scatter_noisy = plt.scatter(X_pca_noisy[:, 0], X_pca_noisy[:, 1], c=y, cmap='Spectral', edgecolor='k', s=60)
plt.colorbar(scatter_noisy)
plt.xlabel('First Principal Component (Noisy)')
plt.ylabel('Second Principal Component (Noisy)')
plt.title('2D PCA of Noisy Digits Dataset')
plt.show()
```

Output






2D PCA of Noisy Digits Dataset
Second Principal Component (Noisy)
First Principal Component (Noisy)
8
6
4
2
0
−2
−4
−6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
9
8
7
6
5
4
3
2
1
0