```python
##################################
# TITLE: ex5 machine learning BIU
# WRITER: Raphael Haehnel
# DATE: 03/01/2023
##################################

import scipy.io
from sklearn.model_selection import train_test_split
import numpy as np
from matplotlib import pyplot as plt
from sklearn.metrics import accuracy_score
from tqdm import tqdm


def helper_retrieve(data):
    a1 = np.split(data, 15)
    a2 = np.array(a1)
    a31 = a2[:, 0:8, :]
    a32 = a2[:, 8:11, :]

    train = np.concatenate(a31)
    test = np.concatenate(a32)

    return train, test


def retrieve_data():
    mat = scipy.io.loadmat('facesData.mat')
    faces = mat.get('faces')
    labels = mat.get('labeles')

    X_train, X_test = helper_retrieve(faces)
    y_train, y_test = helper_retrieve(labels)

    return X_train, X_test, y_train, y_test


def extract_eigenvectors(X: np.ndarray, mu: np.ndarray):
    A = np.matmul((X - mu).T, X - mu)
    w, v = np.linalg.eig(A)
    return w.real.astype('float64'), v.real.astype('float64')


def sort_eigenvectors(w, v):
    indexes = np.argsort(np.abs(w))

    v = v[:, indexes]
    w = w[indexes]

    v = np.flip(v, axis=1)
    w = np.flip(w, axis=0)

    return w, v


def show_PCA(mu, v):
    fig = plt.figure(figsize=(10, 2))
    num = 8
    # Display the mean
    fig.add_subplot(1, num, 1)
    plt.imshow(mu.reshape((32, 32)).T, cmap='gray')
    plt.title(f"mu")
    plt.axis('off')

    # Displaying the first 5 eigenvectors
    for i in range(num-1):
        fig.add_subplot(1, num, i+2)
        plt.imshow(v[:, i].reshape((32, 32)).T, cmap='gray')
        plt.title(f"v{i}")
        plt.axis('off')

    fig.canvas.manager.set_window_title('show_PCA')
    plt.show()


def compute_projection(v, x_T, mu, K):

    v_k = v[:, :K]
```

```
    projection = np.matmul(x_T - mu, v_k)
    reconstruction = mu + np.matmul(projection, v_k.T)

    return projection, reconstruction


def reconstruct_image(v, mu, x_T):

    # Build the displaying window
    fig = plt.figure(figsize=(7, 7))
    rows = 5
    cols = 5

    for i in range(24):

        # We choose the i first eigenvectors
        projection, reconstruction = compute_projection(v, x_T, mu, i)

        fig.add_subplot(rows, cols, i+1)

        # Display the projected eigenvector
        plt.imshow(reconstruction.reshape((32, 32)).T, cmap='gray')
        plt.axis('off')
        plt.title(f"K={i}")

    projection, reconstruction = compute_projection(v, x_T, mu, v.shape[1])

    fig.add_subplot(rows, cols, 25)

    # Display the projected eigenvector
    plt.imshow(reconstruction.reshape((32, 32)).T, cmap='gray')
    plt.axis('off')
    plt.title(f"K=1024")

    fig.canvas.manager.set_window_title('reconstruct_image')
    plt.show()


def mesure_projections(projections_train, projections_test, y_train):

    label_list = []

    for i in range(len(projections_test)):

        # Measure the euclidian distance between one train projection and
        # all the test projections
        dist = np.linalg.norm(projections_train - projections_test[i], axis=1)

        # Extract the index from the train set for which the distance is minimal
        index_train = np.argmin(dist)

        # Find the corresponding label of this train sample
        label = y_train[index_train]

        label_list.append(label)

    return np.array(label_list)


def show_accuracy_graph(X_train, mu, X_test, y_train, y_test):
    accuracy_list = []
    x = list(range(1, 50, 1))

    for i in tqdm(x):
        K = i

        # Get the projections of all the train set
        projections_train, _ = compute_projection(v, X_train, mu, K)

        # Get the projections of all the test set
        projections_test, _ = compute_projection(v, X_test, mu, K)

        # Mesure the distance from each projection of the train set
        # to all the projections of the train set
        y_pred = mesure_projections(
            projections_train, projections_test, y_train)

        # Compute the accuracy classification score.
        accuracy = accuracy_score(y_test, y_pred)
```

```
        accuracy_list.append(accuracy)

    plt.plot(x, accuracy_list)
    plt.xlabel("K")
    plt.ylabel("Accuracy")
    plt.title("Accuracy of the PCA projections")
    plt.show()


if __name__ == "__main__":

    # Creating training and testing sets
    X_train, X_test, y_train, y_test = retrieve_data()

    # Find the mean image
    mu = X_train.mean(0)

    # Find the eigenvectors
    w, v = extract_eigenvectors(X_train, mu)

    # Sort the eigenvectors
    w, v = sort_eigenvectors(w, v)

    # Transpose the first image of the dataset
    x_T = np.reshape(X_train[0], (1, 1024))

    # Show the principal components
    show_PCA(mu, v)

    # Show the reconstruction of the images
    reconstruct_image(v, mu, x_T)

    # Show the accuracy graph
    show_accuracy_graph(X_train, mu, X_test, y_train, y_test)

    K = 16

    # Get the projections of all the train set
    projections_train, _ = compute_projection(v, X_train, mu, K)

    # Get the projections of all the test set
    projections_test, _ = compute_projection(v, X_test, mu, K)

    # Mesure the distance from each projection of the train set
    # to all the projections of the train set
    y_pred = mesure_projections(
        projections_train, projections_test, y_train)

    # Compute the accuracy classification score.
    accuracy = accuracy_score(y_test, y_pred)

    print(f"Accuracy: {accuracy*100}%")

```