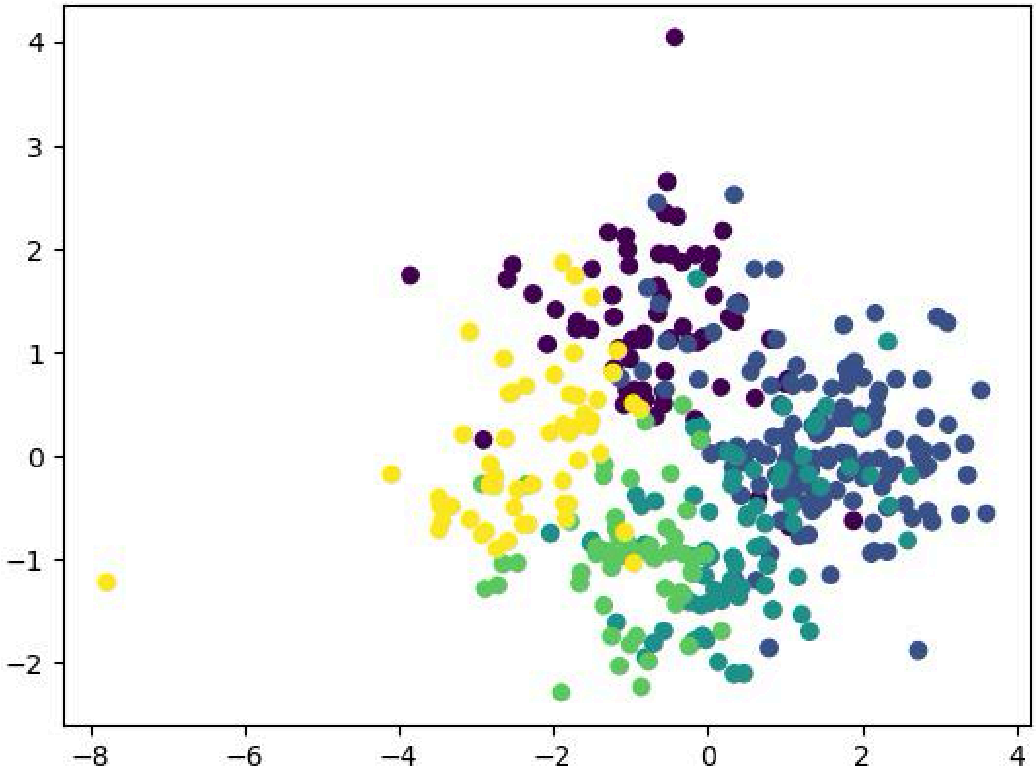
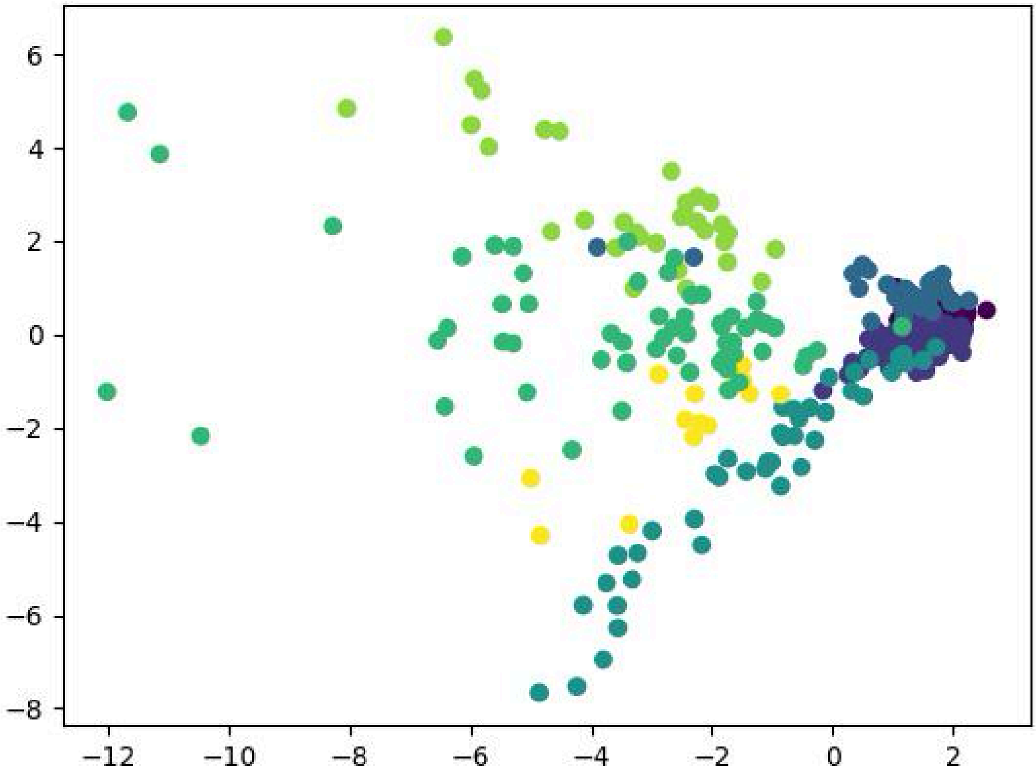
Assignment 1

1) Two scatter plots obtained by running PCA on Cho and Iyer datasets.

- Cho



- Iyer



2) The codes of PCA and plot drawing.

```
def pca(dataMat, PC_num=2):
    '''
    Input:
        dataMat: obtained from the loadDataSet function, each row represents an
observation
                 and each column represents an attribute
        PC_num:  The number of desired dimensions after applyting PCA. In this
project keep it to 2.
    Output:
        lowDDataMat: the 2-d data after PCA transformation
    '''
    # print(dataMat.shape) # (150, 4) for iris dataset. 4 attributes, 150 observati
```

```
ons

    # use the formula mentioned in the lecture slides to implement PCA!

    # cal the mean for each attribute
    meanVals = mean(dataMat, axis=0)
    # print(meanVals.shape) ## (4,) for iris dataset

    # center the data
    x_prime = dataMat - meanVals # NumPy applies broadcasting: It matches the (4,) vector against each row of (150, 4).
    # print(x_prime.shape) ## (150, 4)

    # compute the covariance matrix
    x_prime_T = x_prime.T
    S = (x_prime_T @ x_prime) / (dataMat.shape[0] - 1) # @ is matrix multiplication operator
    # print(S.shape) ## (4, 4) for iris dataset

    # find the eigenvalues and eigenvectors
    eigvals, eigvecs = numpy.linalg.eig(S)
    # print(eigvals.shape)  # (4,) for iris dataset
    # print(eigvecs.shape)  # (4, 4) for iris dataset

    # sort the eigenvalues in descending order and select the top 2 eigenvectors
    pairs = []
    for i in range(len(eigvals)):
        # In NumPy’s eig output, each column is an eigenvector, not each row.
        pairs.append((eigvals[i], eigvecs[:, i]))

    # sort the tuples based on the first element, which is the eigenvalue. Reverse to get descending order
    # key is x[0], which means we sort by the first element of the tuple
    pairs.sort(key=lambda x: x[0], reverse=True)
    # select the top 2 eigenvectors
    # reshape the vector to a specific shape (rows, columns)
    # The -1 tells NumPy to "figure out" the correct number of rows automatically
    # based on the length of the array, and 1 fixes the number of columns.
    featureVec1 = pairs[0][1].reshape(-1, 1) # from (4,) to (4, 1). 1D → 2D
    featureVec2 = pairs[1][1].reshape(-1, 1) # from (4,) to (4, 1). 1D → 2D

    # construct the projection matrix
    W = numpy.hstack((featureVec1, featureVec2)) # W is (4, 2)

    # project the data onto the new subspace
    lowDDataMat = x_prime @ W # x_prime is (150, 2)

    return array(lowDDataMat)



def plot(lowDDataMat, labelMat, figname):
    '''
    Input:
        lowDDataMat: the 2-d data after PCA transformation obtained from pca function
        labelMat: the corresponding label of each observation obtained from loadData
    '''
#    In the plot function you need to plot all observations as scatter plots and color the data
#    points according to their labels. You also need to save the figure.
    plt.figure()
    plt.scatter(lowDDataMat[:, 0], lowDDataMat[:, 1], c=labelMat)
    plt.savefig(figname)
    plt.show()
    plt.close()
```

3) AI usage disclosure statement

I use Copilot in Visual Studio Code for code and comment completion on assignment 1.