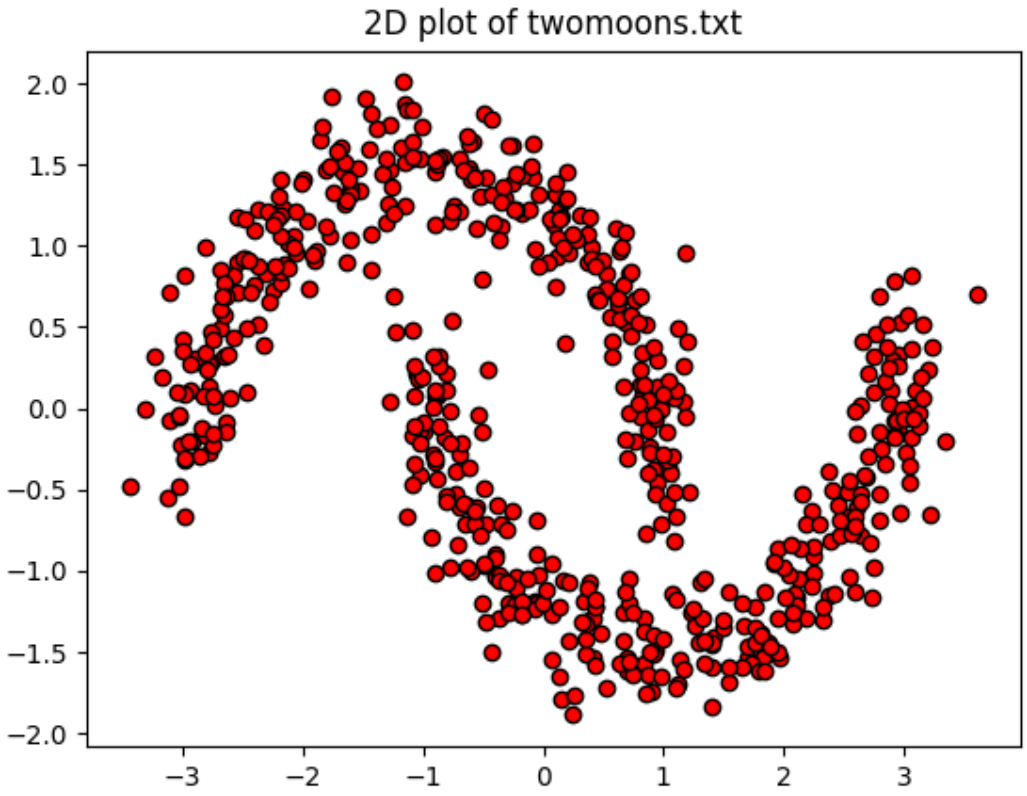
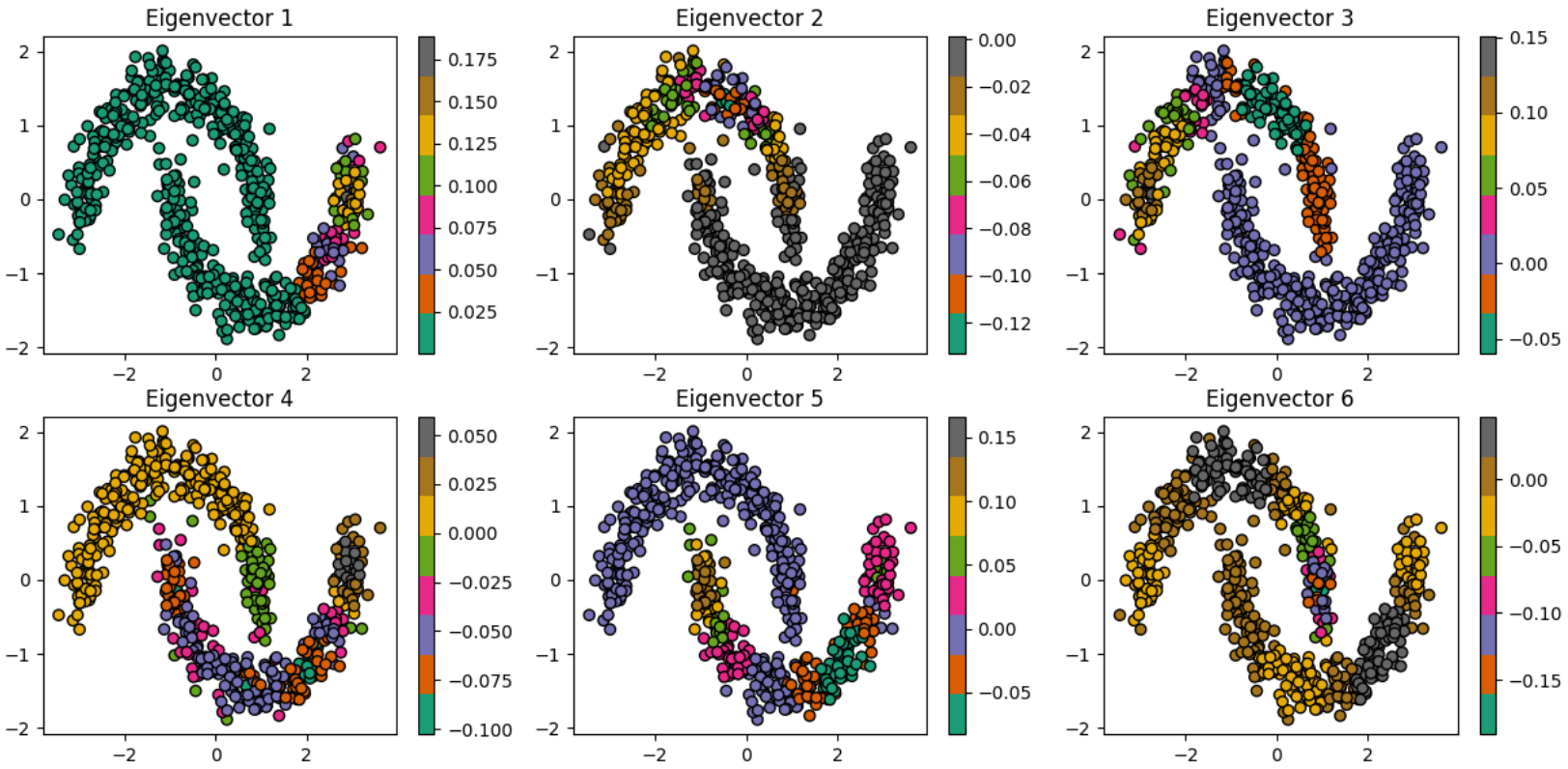

2D plot of twomoons.txt
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−3
−2
−1
0
1
2
3


Eigenvector 1
Eigenvector 2
Eigenvector 3
Eigenvector 4
Eigenvector 5
Eigenvector 6

Eigenvectors 2, 3 and 5 show a clearer separation between the two moons (by showing which moon has a lot more variance and which one has not), while eigenvectors 1 and 4 do not have either moon displaying one single class, and eigenvector 6 has virtually no use at all.

Code snippet:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neighbors import kneighbors_graph

# load data
twomoons = "C:/Users/alepa/Desktop/Data Science/[1] 2nd Semester/Mathematical Models
and Numerical Methods for Big Data/Code/twomoons.txt"
data_text = np.loadtxt(twomoons, delimiter=',')

x_coords = data_text[:, 0]
y_coords = data_text[:, 1]

data = pd.DataFrame(data_text, columns=['x', 'y'])

plt.scatter(x_coords, y_coords, color='red', edgecolor='k')
plt.title('2D plot of twomoons.txt')
plt.show()

# a thumb rule for k-NN is to set k to the square root of the number of data points
k = int(np.sqrt(len(data_text))) # 141

# constructing the k-nearest neighbor graph
knn_graph = kneighbors_graph(data, n_neighbors=k, mode='connectivity',
include_self=False)
A = knn_graph.toarray()

# get Laplacian
D = np.diag(A.sum(axis=1))
L = D - A

# eigendecomposition to get the first 6 eigenvectors with numpy
eigenvalues, eigenvectors = np.linalg.eigh(L)
eigenvectors = eigenvectors[:, :6]

# plotting the eigenvectors
plt.figure(figsize=(15, 7))
for i in range(6):
    plt.subplot(2, 3, i + 1)
    plt.scatter(x_coords, y_coords, c=eigenvectors[:, i], cmap='Dark2', edgecolor='k')
    plt.title(f'Eigenvector {i + 1}')
    plt.colorbar()

plt.show()
```