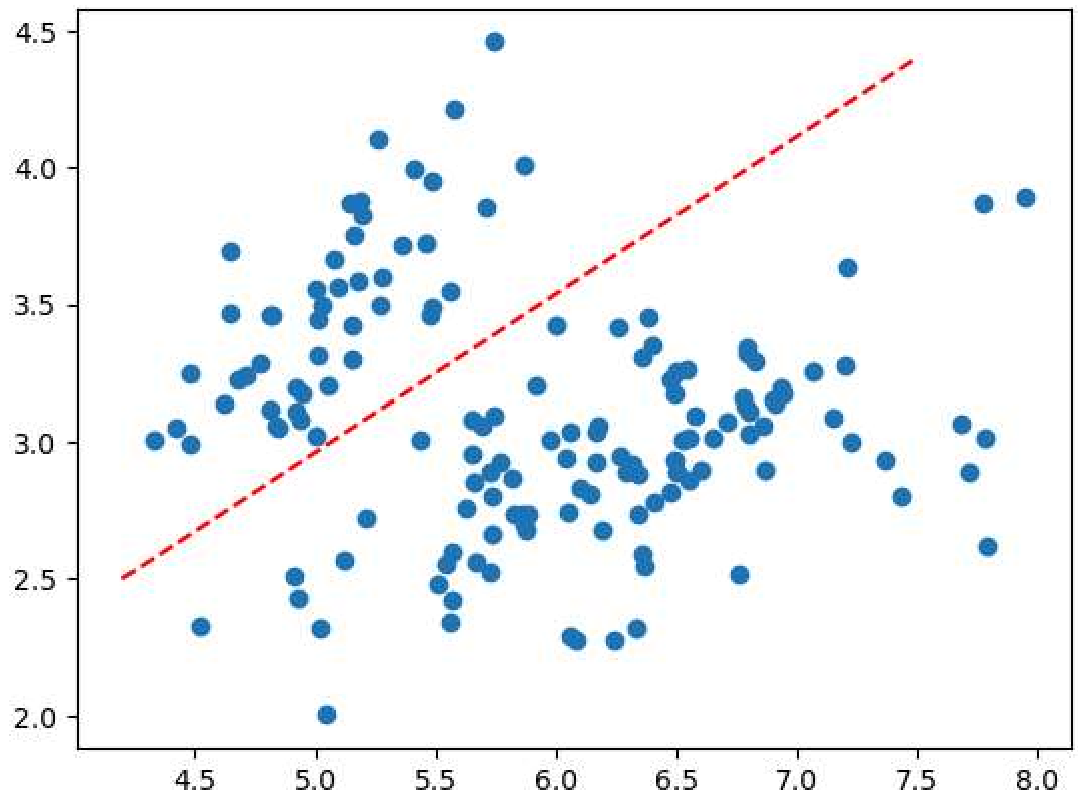
1 Clustering

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('Data/iris.txt')
iris  = data[:,0:2:1]
plt.scatter(iris[:,0],iris[:,1])
plt.plot([4.2,7.5],[2.5,4.4],"r",linestyle='--')
```

Out[1]: [<matplotlib.lines.Line2D at 0x1490f84a4d0>]

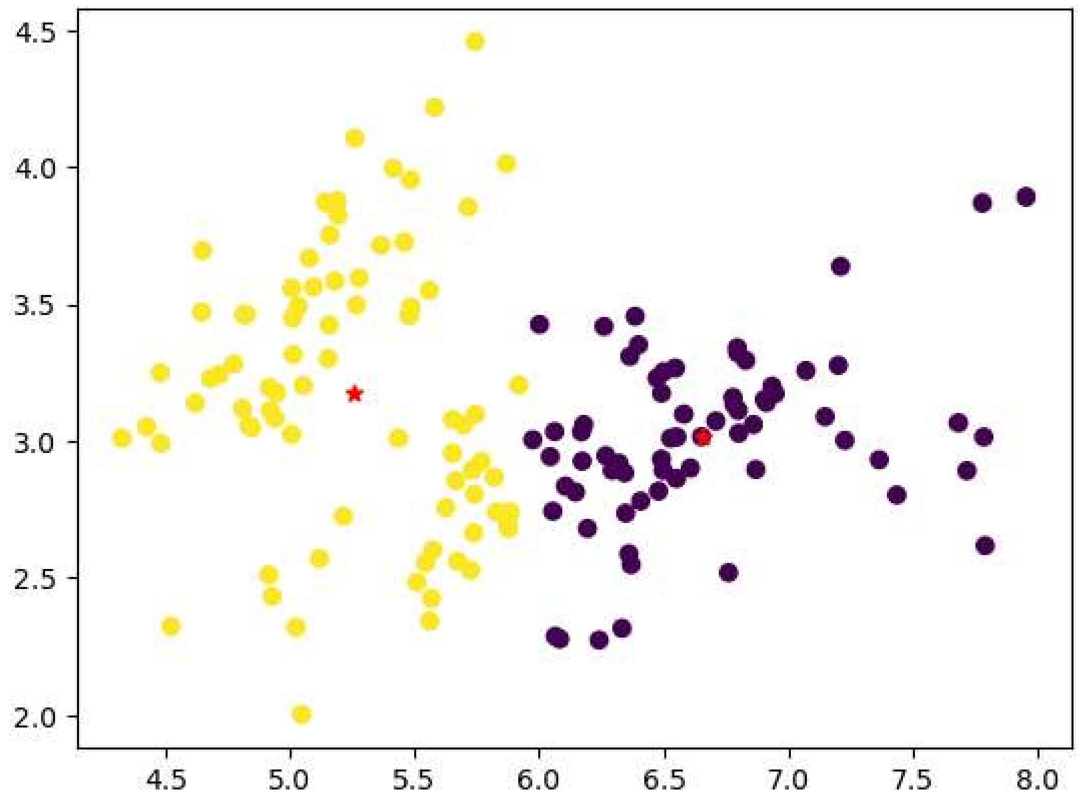


I think there are 2 clusters here, and I draw a red dot line to split them.

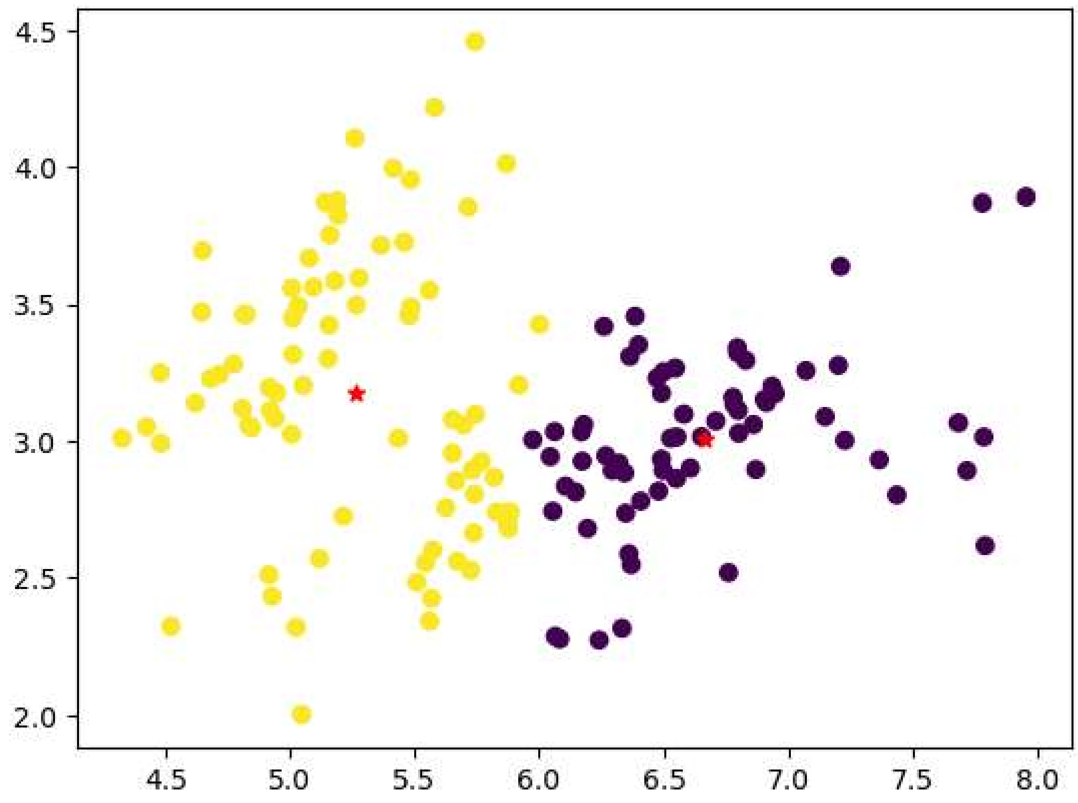
In [2]:
```
import mltools.plot as plot
import mltools as ml
```

In [48]:
```
def test_K_of_model(k):
    for i in range(5):
        z,c,s = ml.cluster.kmeans(iris,K=k)
        plt.scatter(iris[:,0],iris[:,1],c=z)
        plt.scatter(c[:,0],c[:,1],marker='*',color ="red")
        plt.show()
        print(s)
```

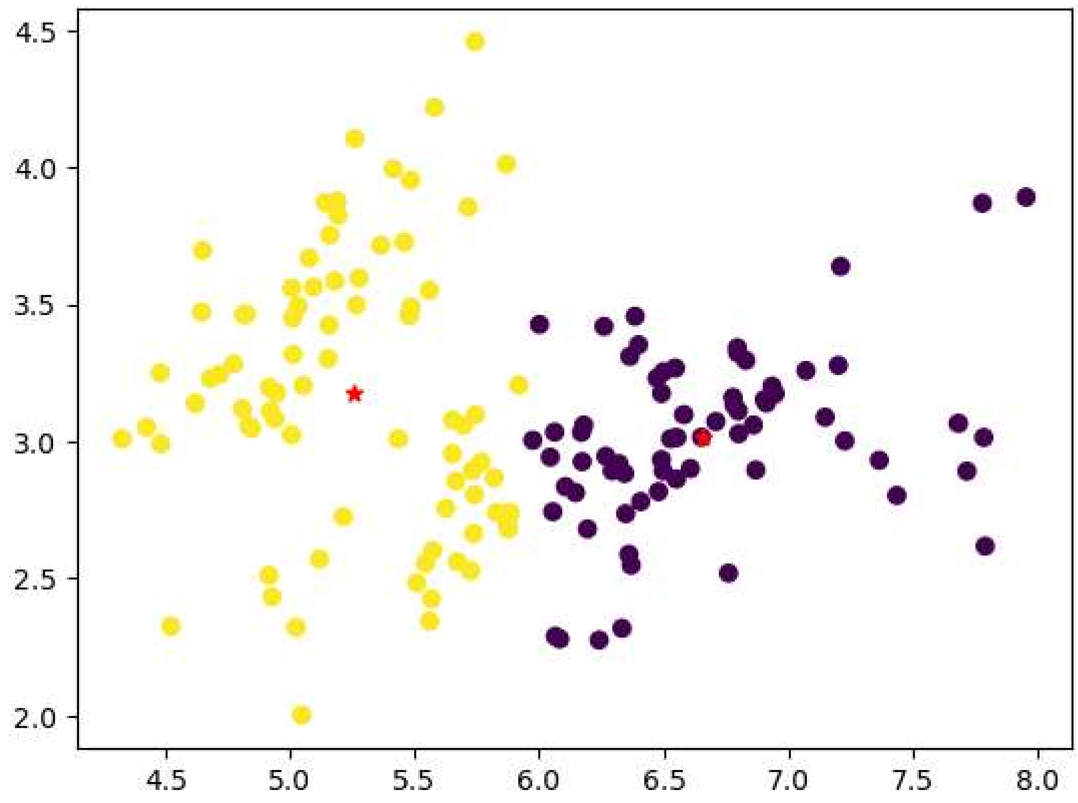
In [49]:
```
for k in [2,5,20]:
    test_K_of_model(k)
```



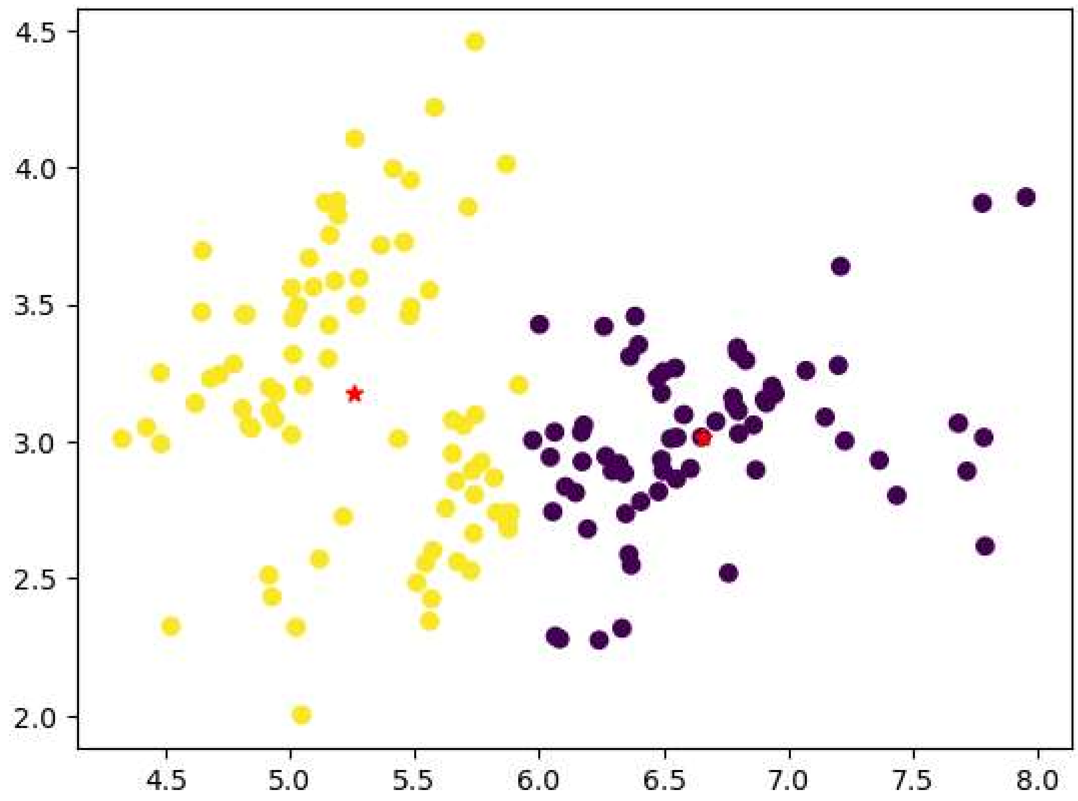

```
57.877648396983034
```



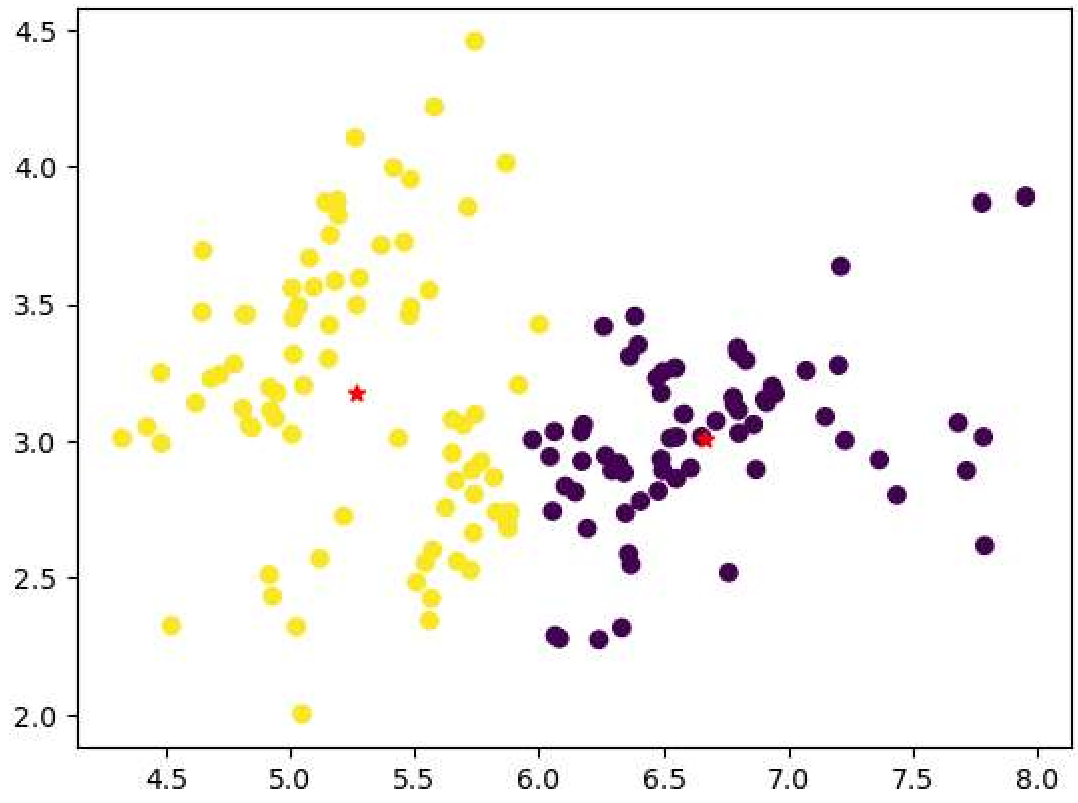
57.87966196118197



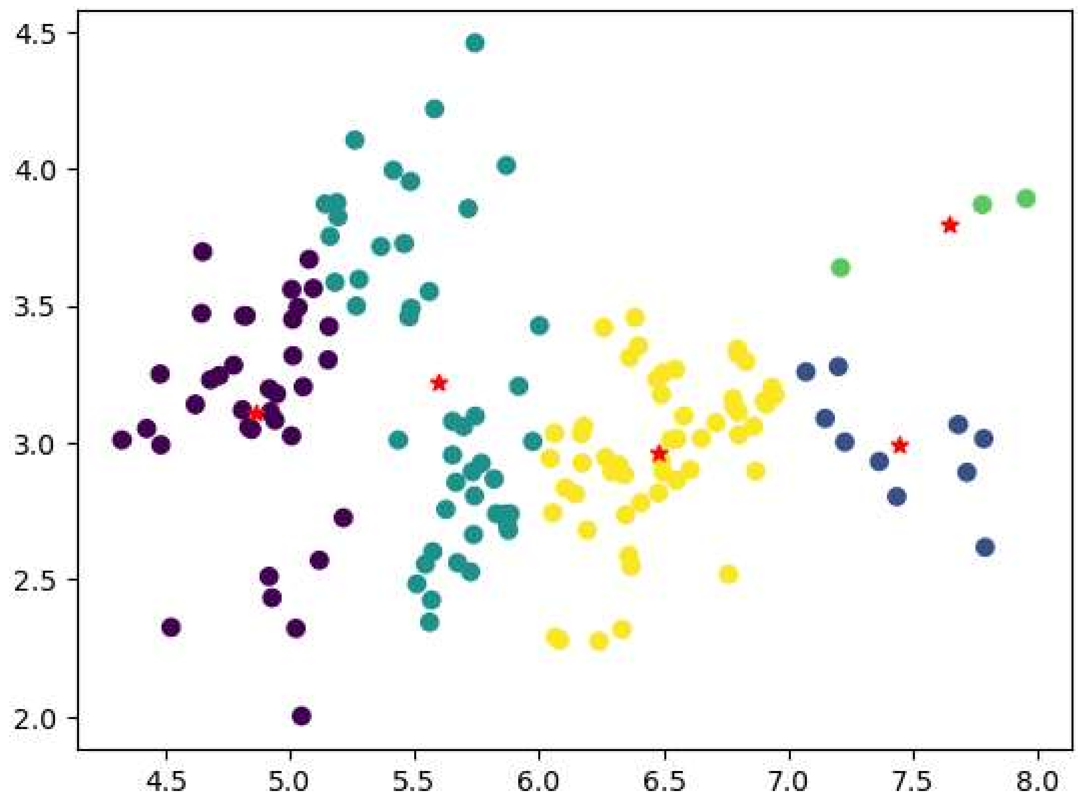
57.877648396983034



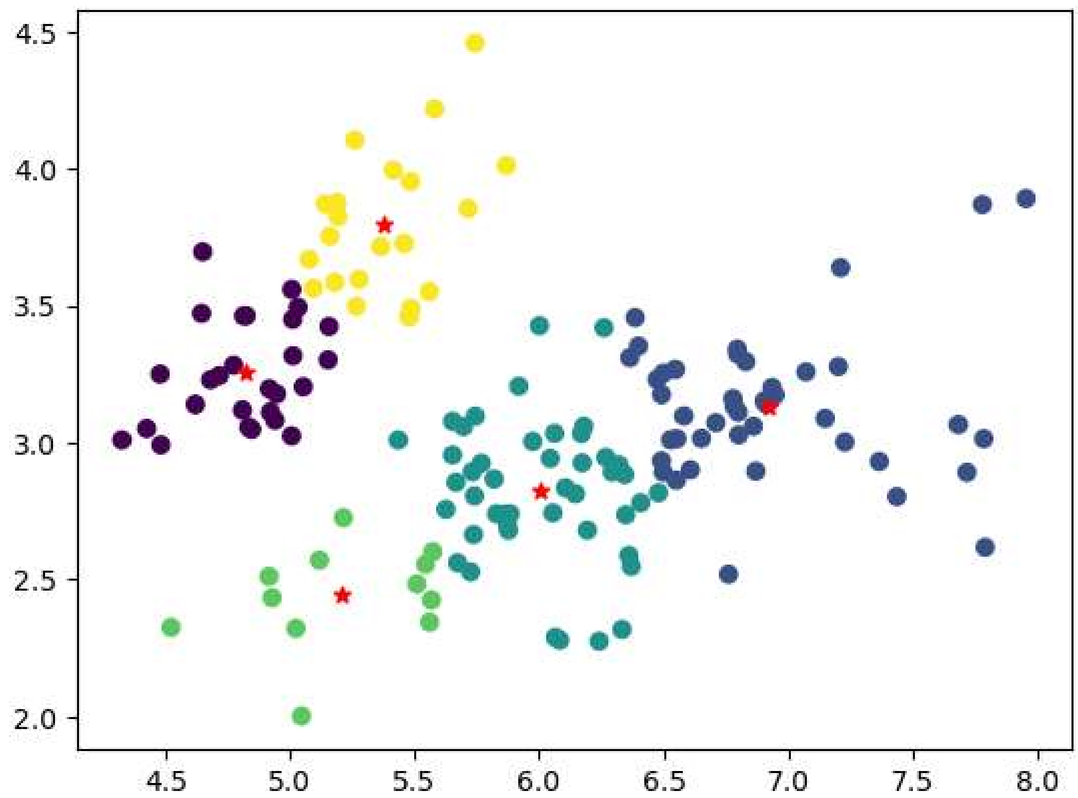
57.877648396983034



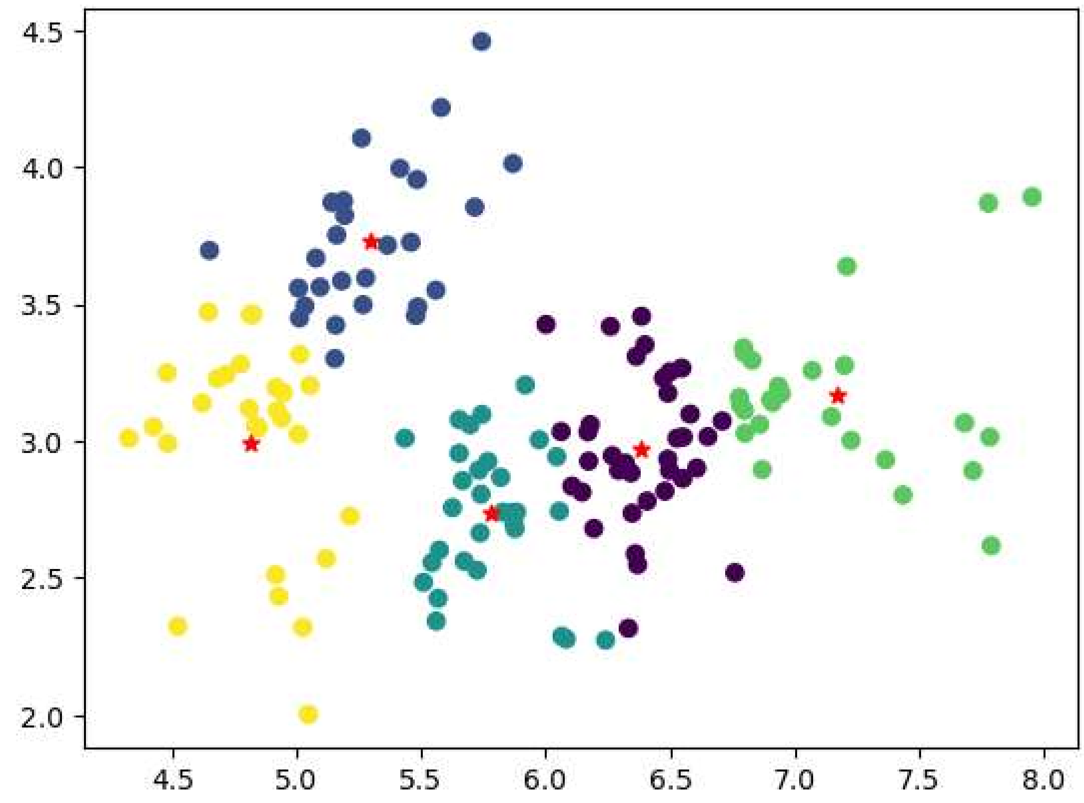
57.87966196118197



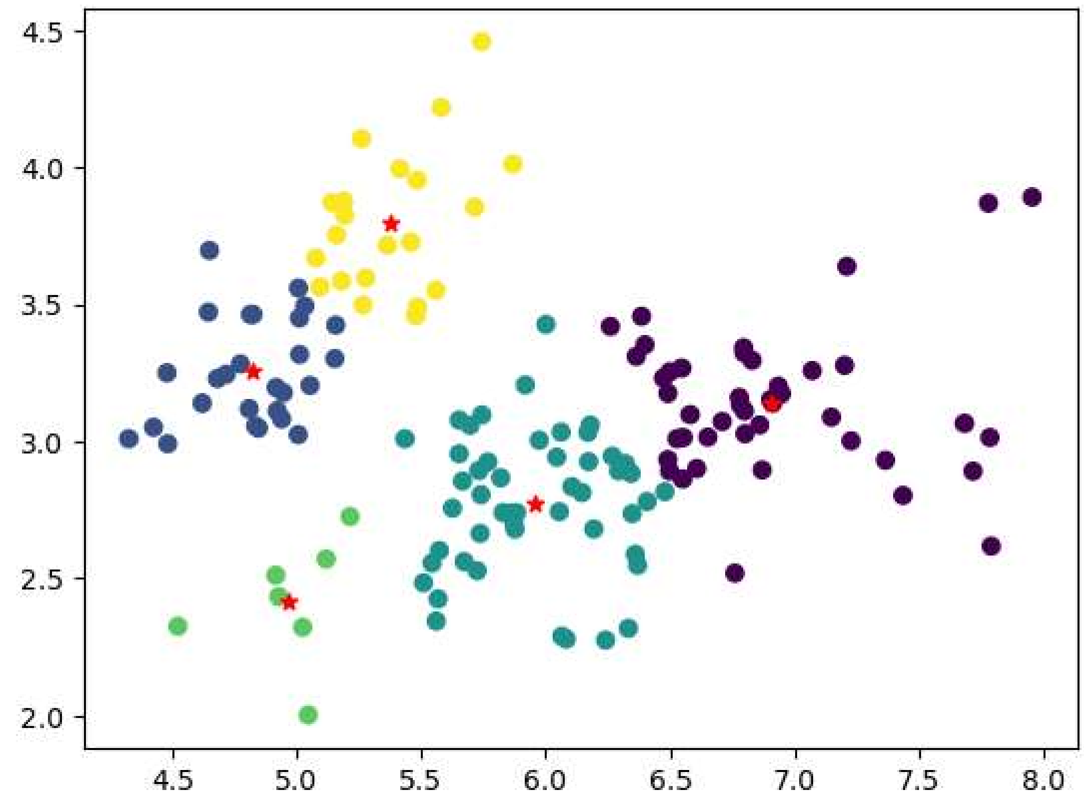
34.03707034981381



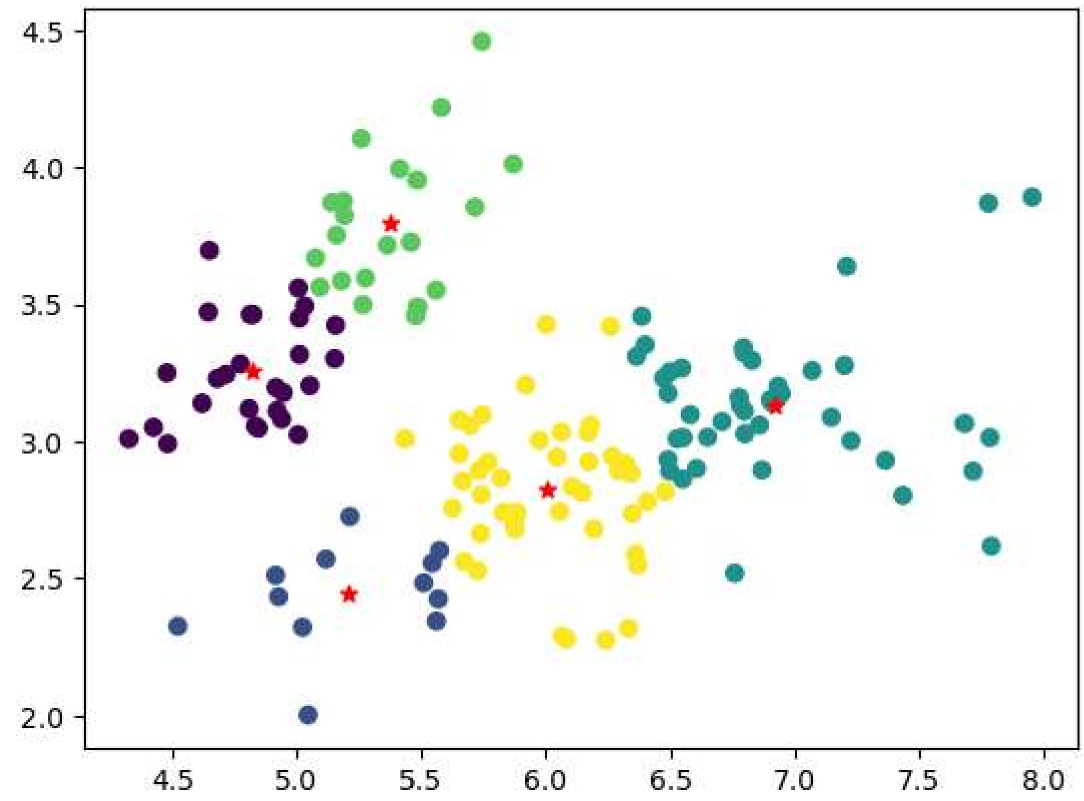
23.438643757297644



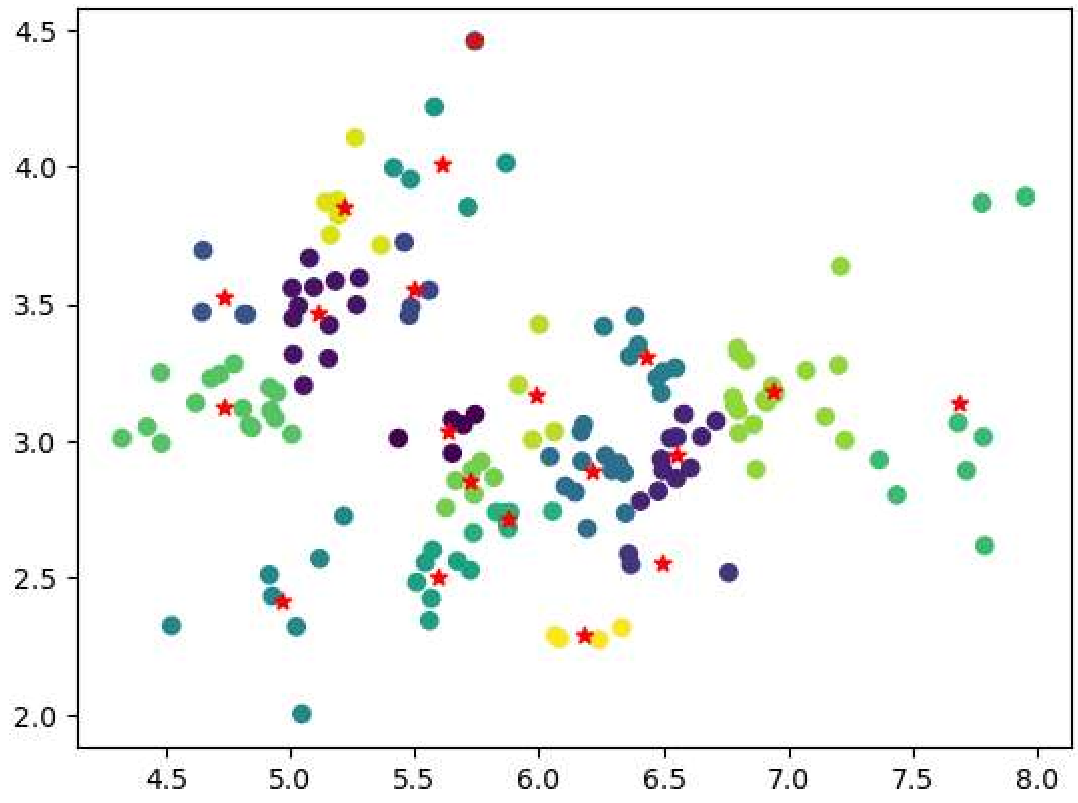
21.67284238824455



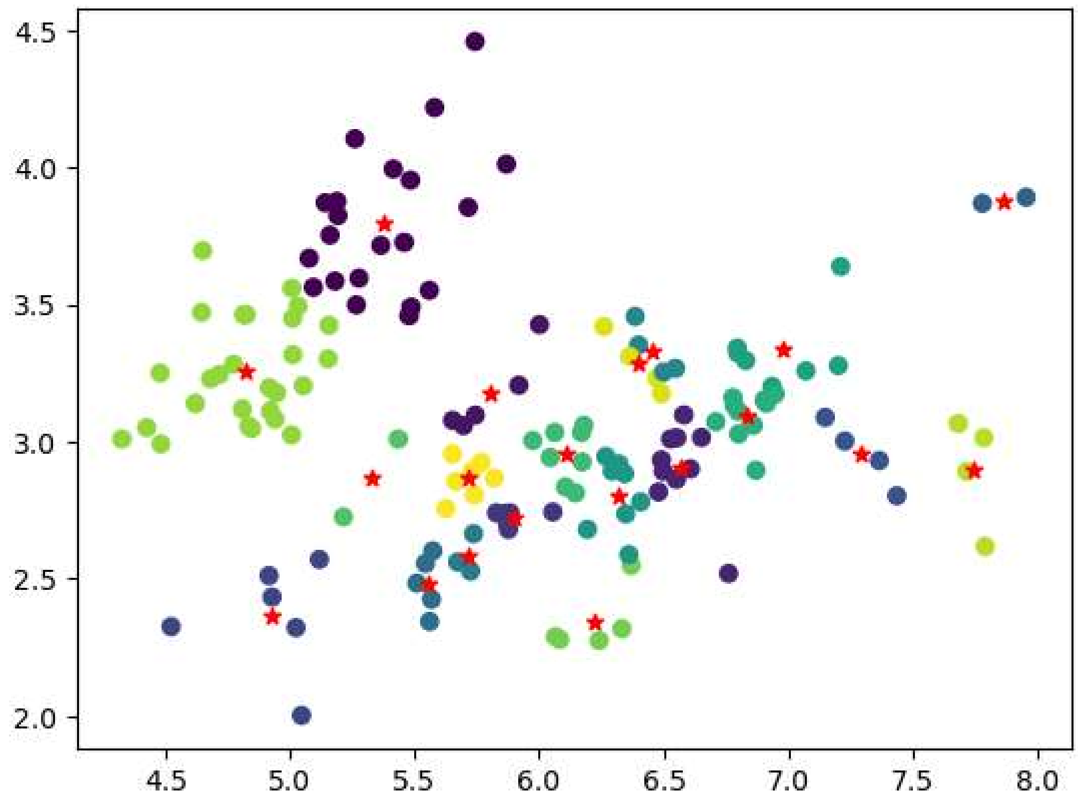
23.889675639615003



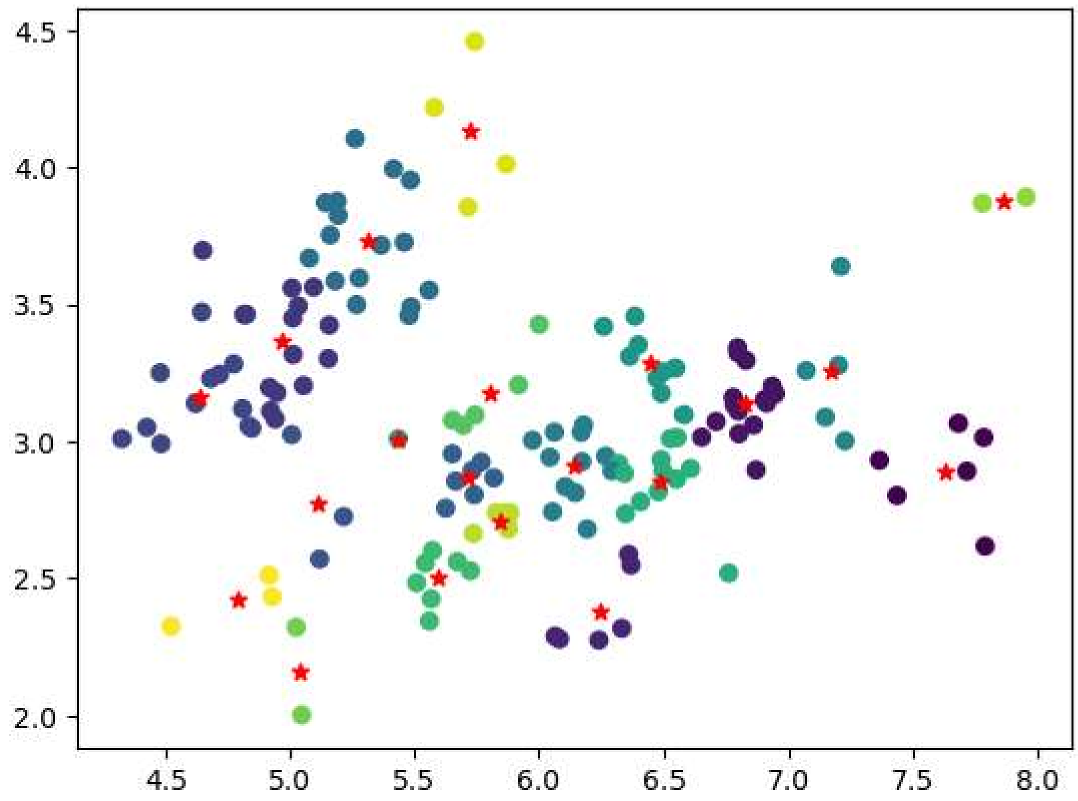
23.438643757297644



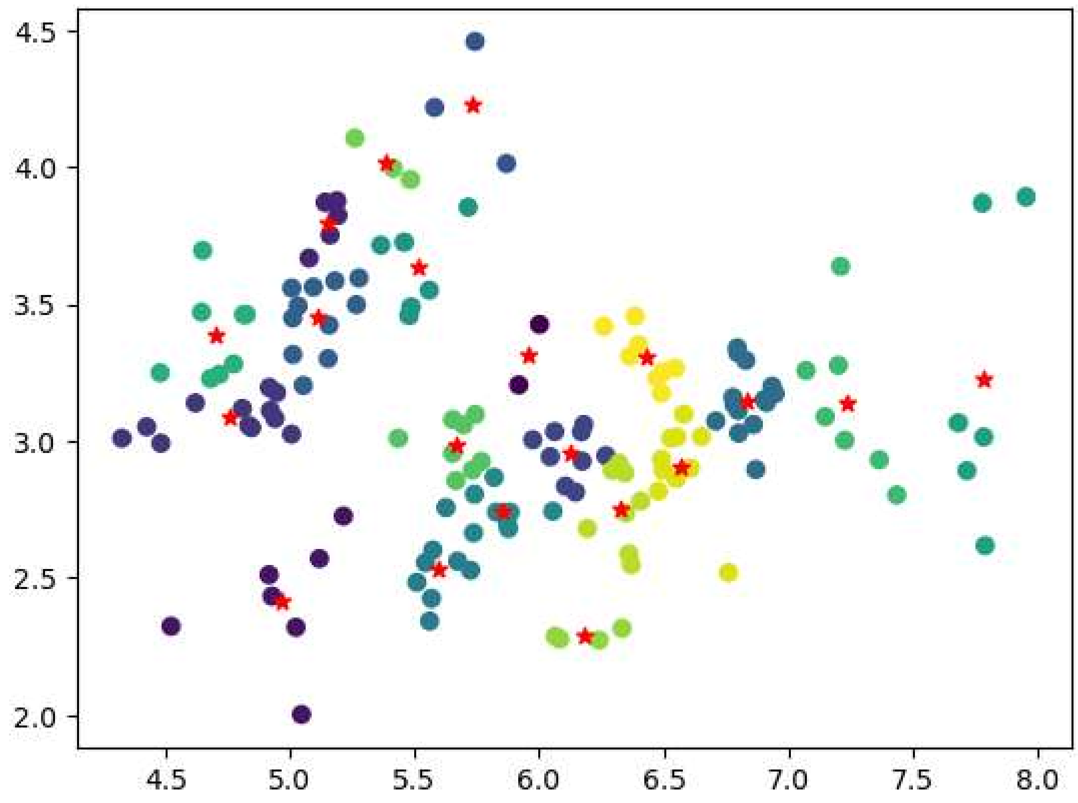

```
6.026975976988429
```



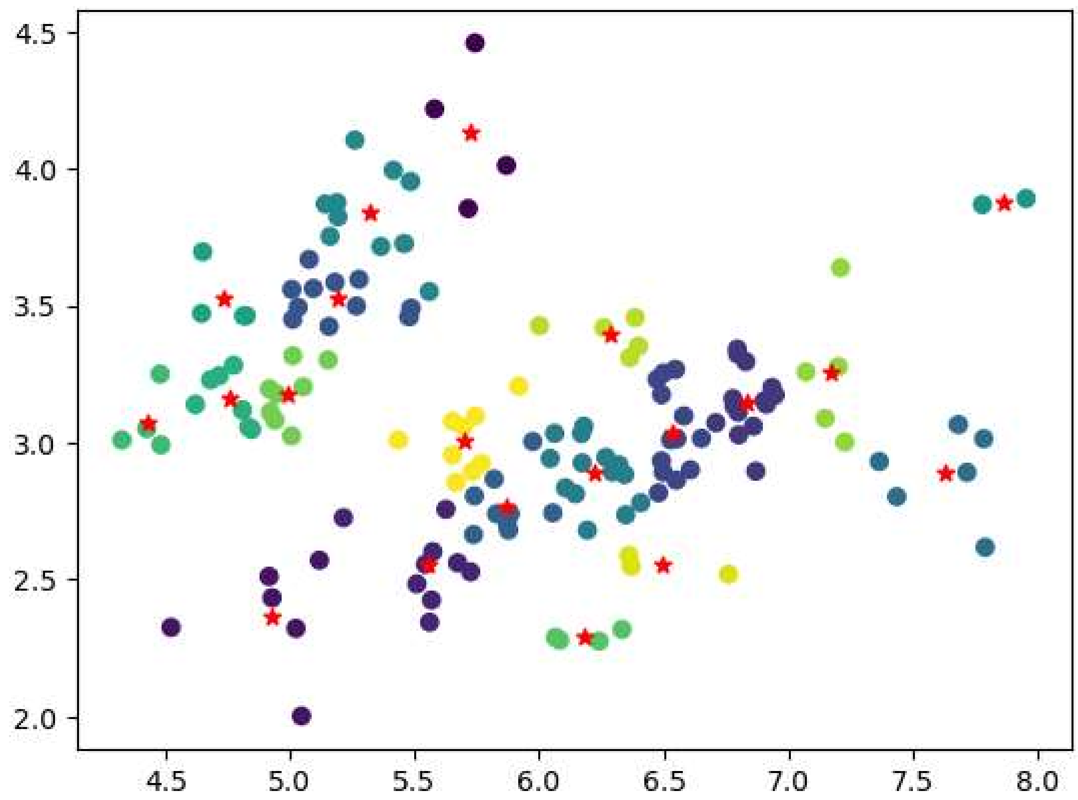

```
6.904042210904514
```




```
4.9634637967437385
```



5.636163874165257

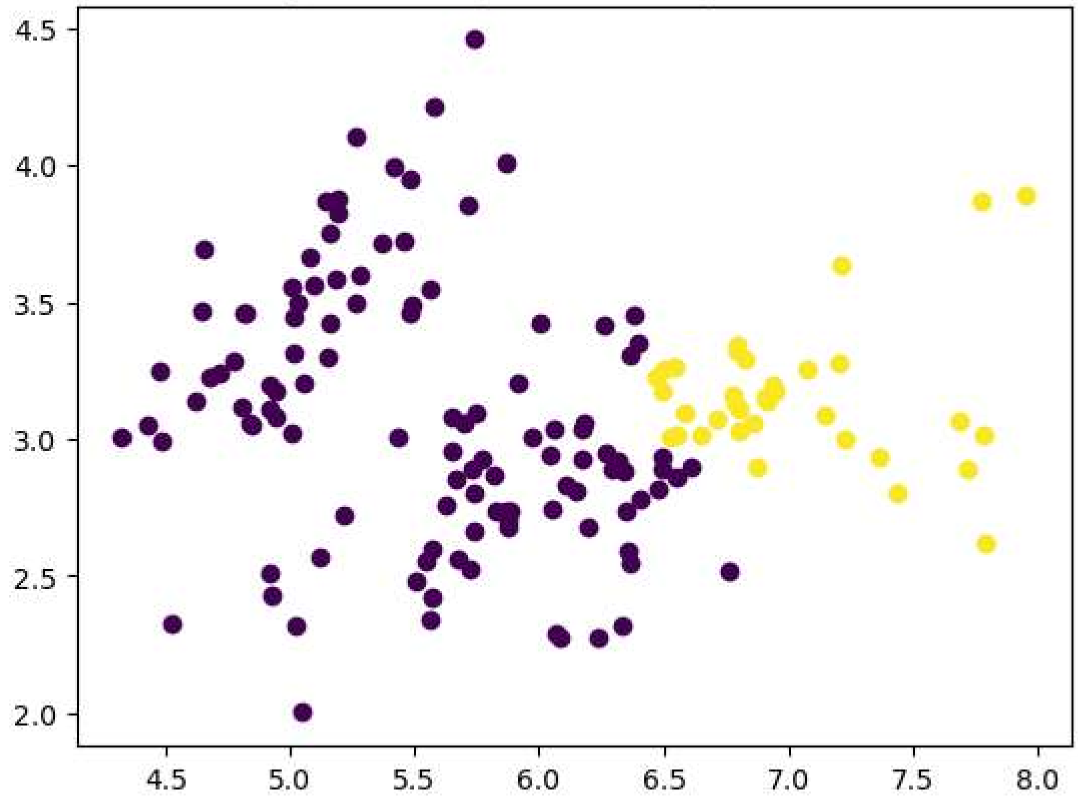


4.26953908350041

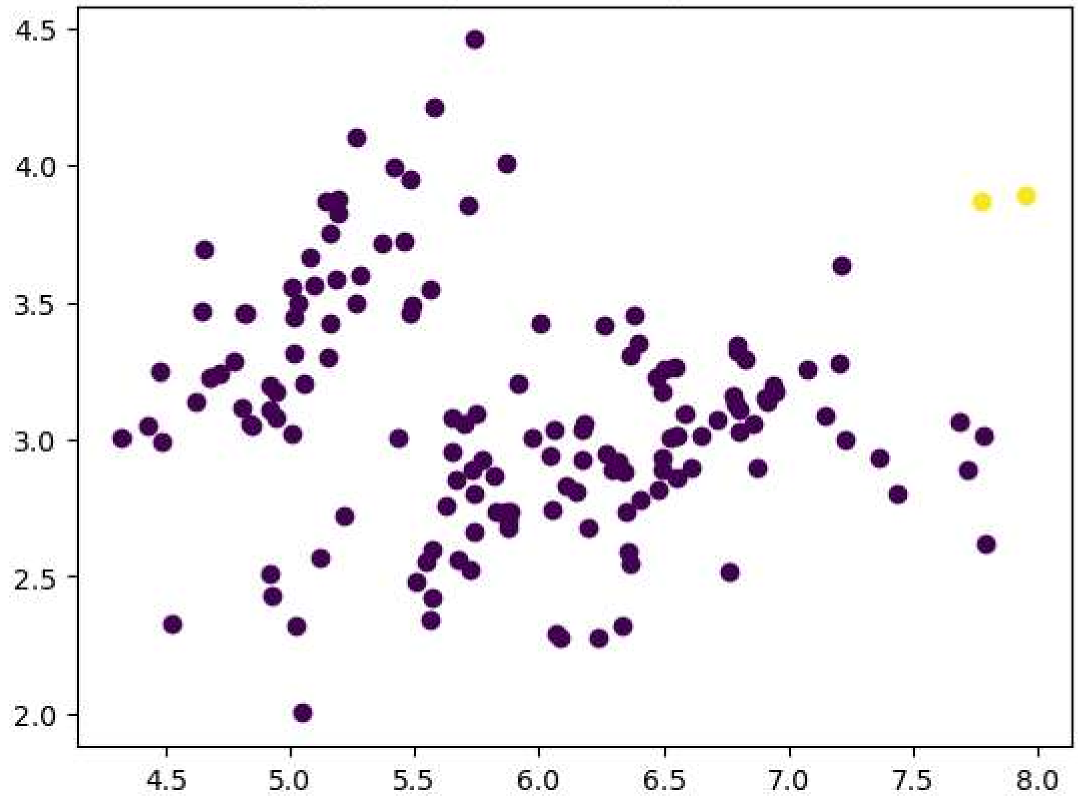
In [4]:
```
def agglomerative_with_K(k):
    z,j=ml.cluster.agglomerative(iris,k,"max")

    plt.title("Complete Link (max distance) of {} clusters".format(k))
    plot.plotClassify2D(None,iris,z)
    #plt.scatter(iris[:,0],iris[:,1],c=z,s = z)
    plt.show()
    z,j=ml.cluster.agglomerative(iris,k,"min")
    plt.title("Single Link (min distance) of {} clusters".format(k))
    #plt.scatter(iris[:,0],iris[:,1],c=z,s = z)
    plot.plotClassify2D(None,iris,z)

    plt.show()
```

In [5]:
```
for k in [2,5,20]:
    agglomerative_with_K(k)
```

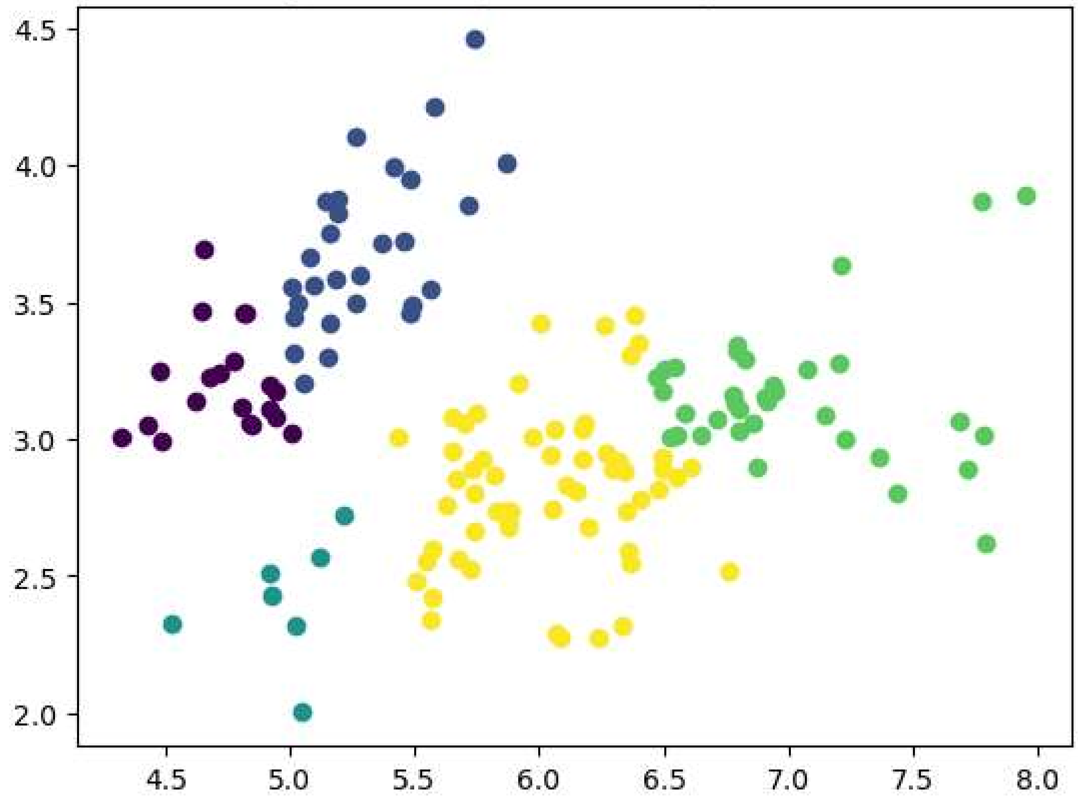
Complete Link (max distance) of 2 clusters

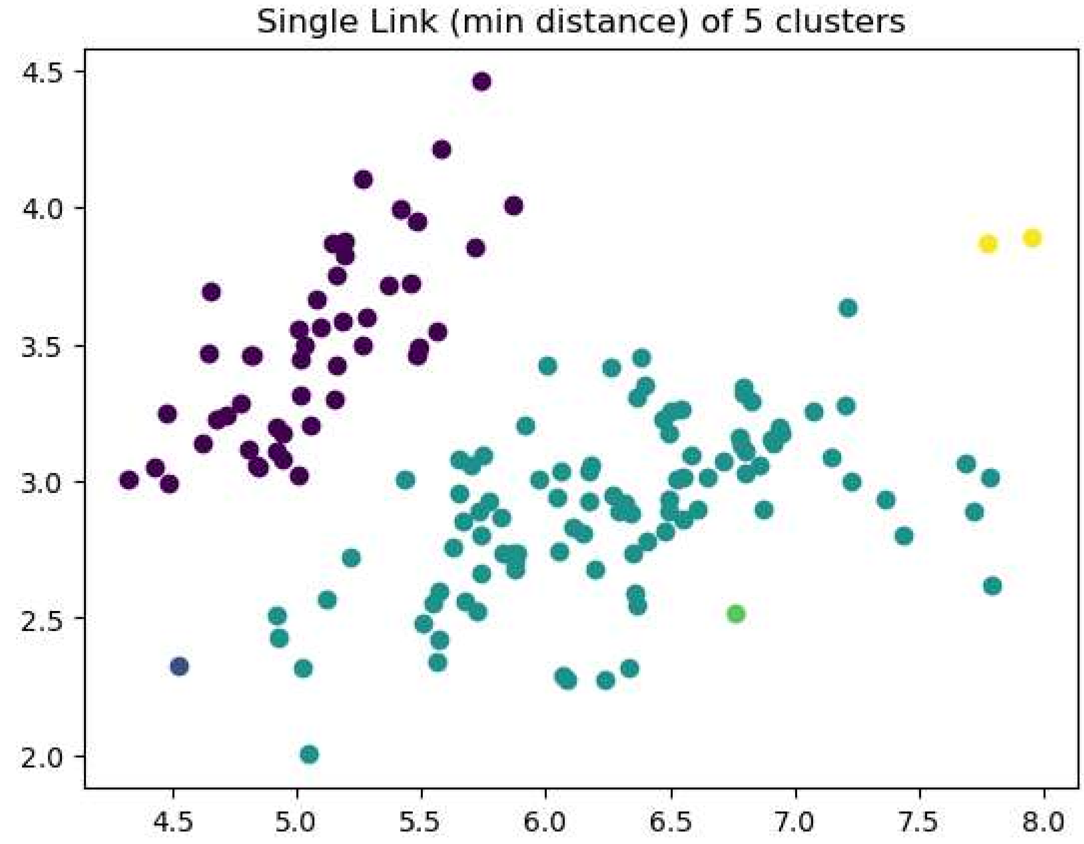


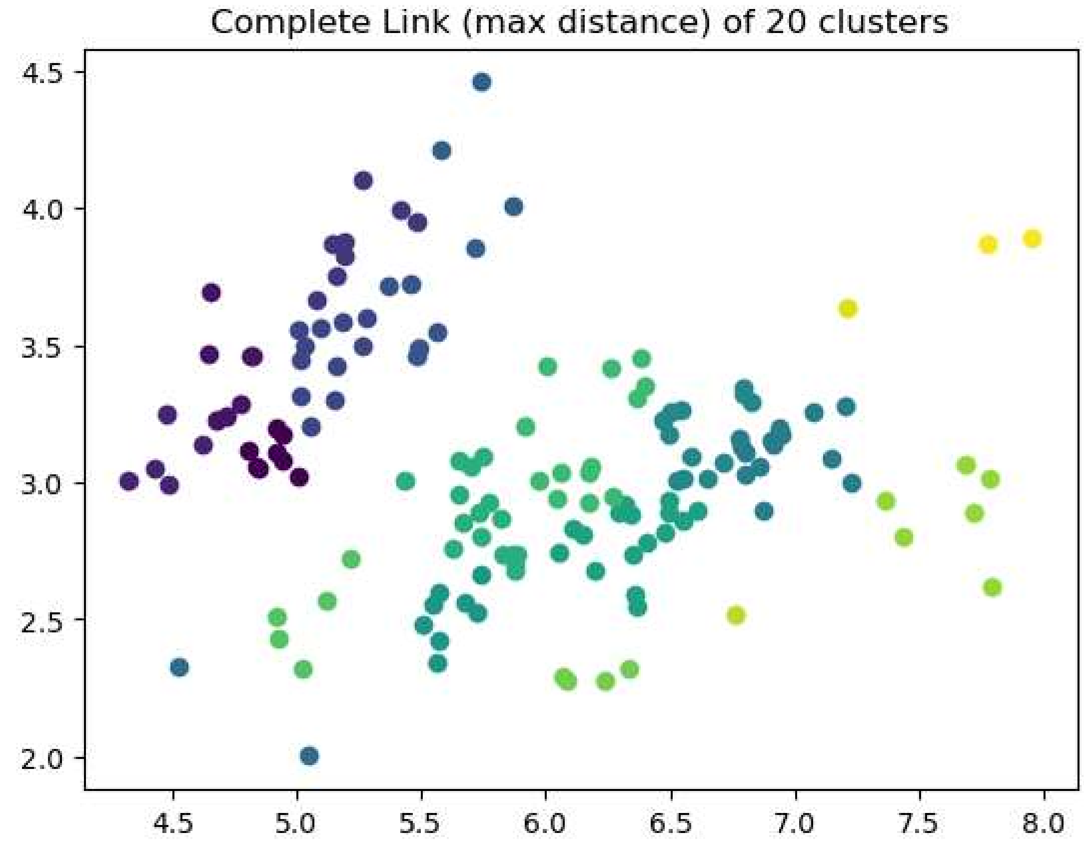
Single Link (min distance) of 2 clusters

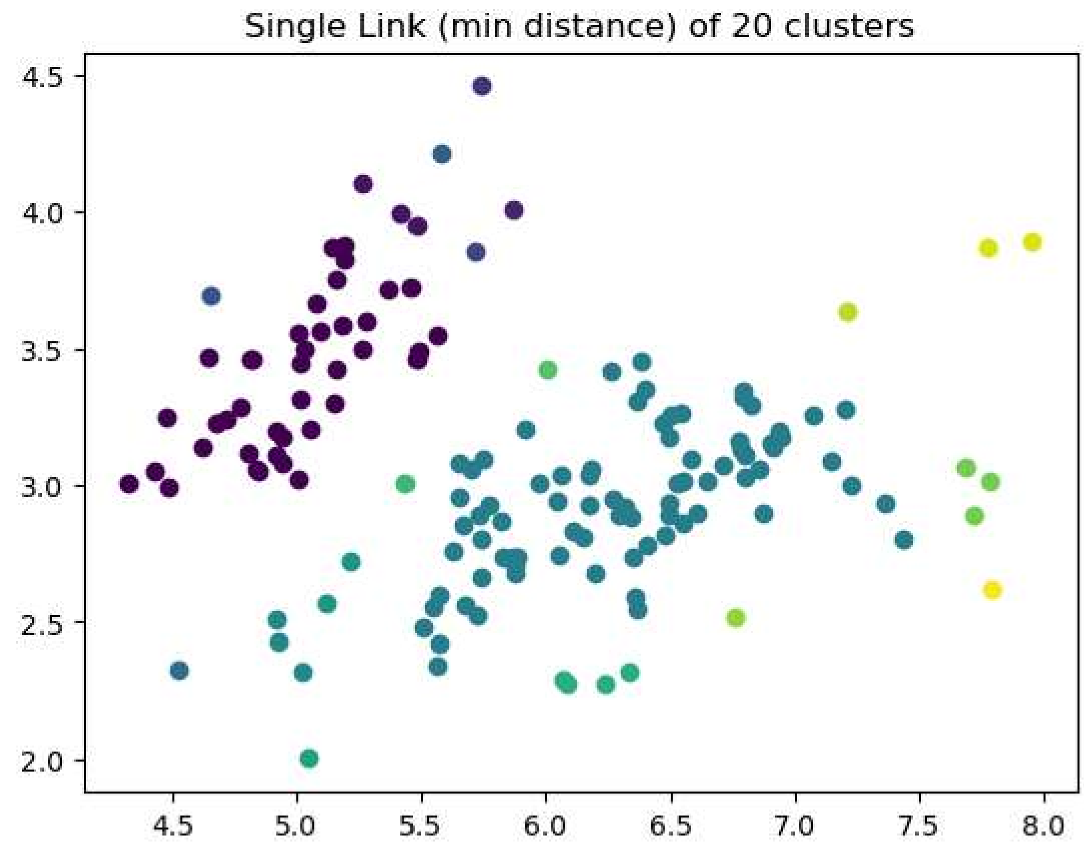


Complete Link (max distance) of 5 clusters







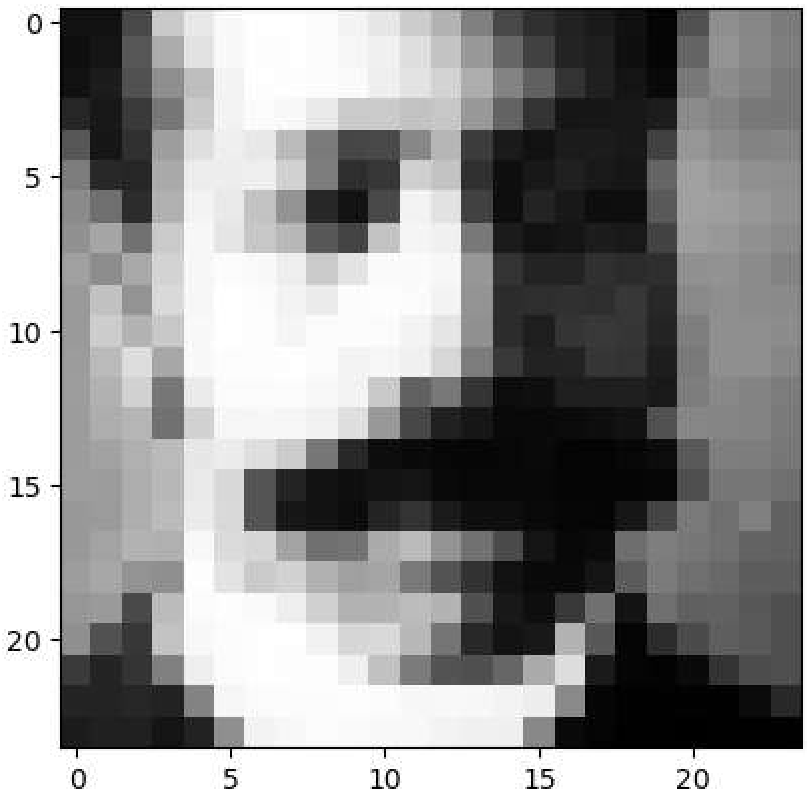


I think the similarity of these two clustering models are that the performance gets better as the number of K improved. But we can hardly find same solution in the two models. In Kmeans, the results is stable when number of cluster is low, and agglomerative models can easily be influenced by outliners. But when the number of cluster improves, the agglomerative models outperforms the Kmeans models.

2 EigenFaces

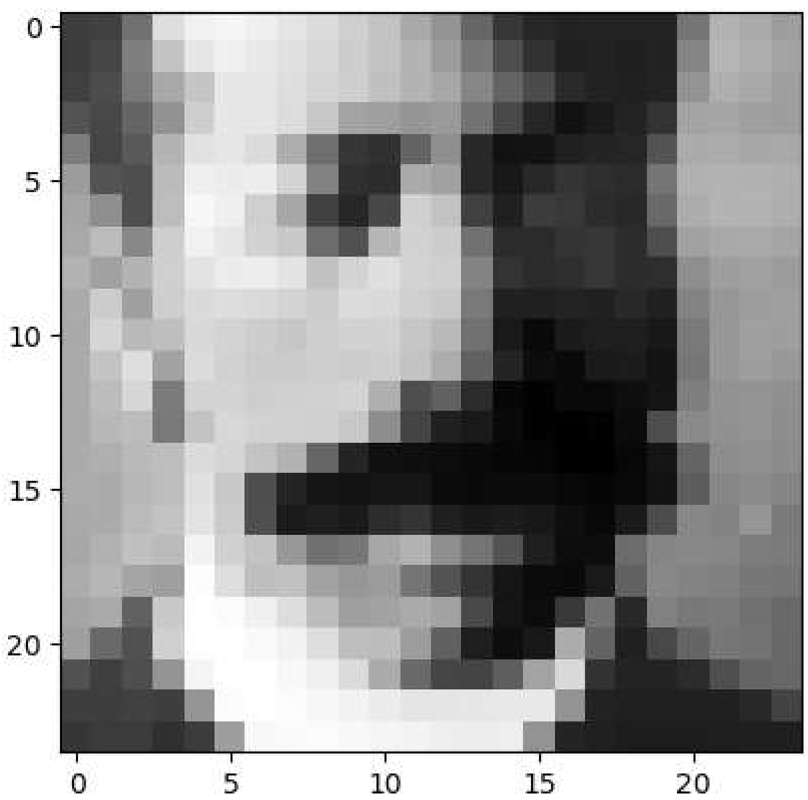
In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
X = np.genfromtxt("data/faces.txt", delimiter=None) # load face dataset
plt.figure()
i = 1
img = np.reshape(X[i,:],(24,24))
plt.imshow( img.T , cmap="gray")
```

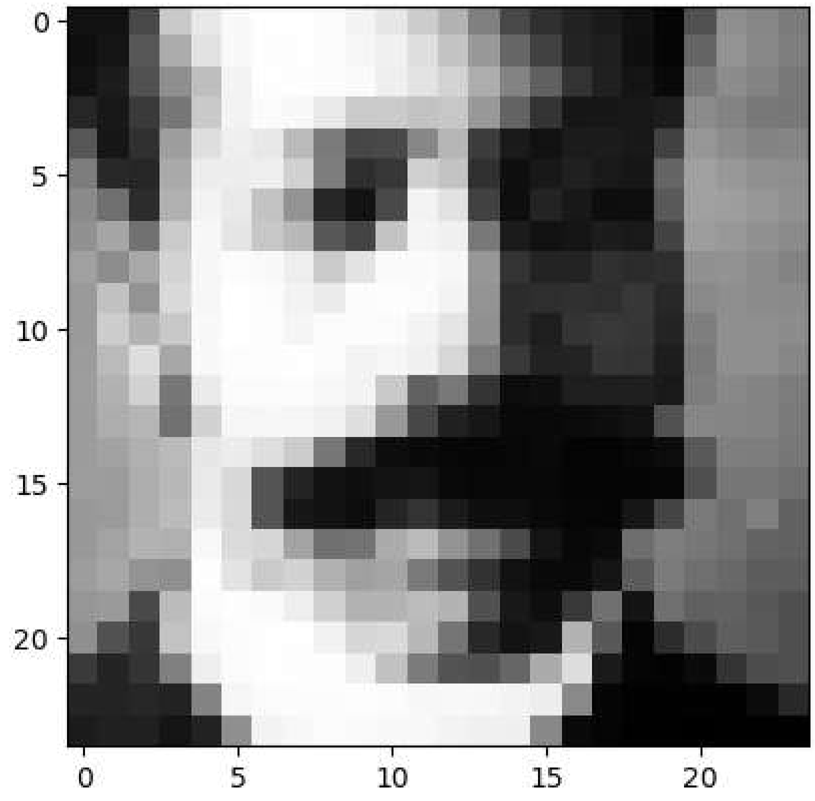
Out[2]: <matplotlib.image.AxesImage at 0x14911a9fdc0>



In [6]:
```
mu = np.mean(X, axis=0)
X0 = X - mu
img = np.reshape(X0[i,:],(24,24))
plt.imshow( img.T , cmap="gray")
plt.figure()
img2 = np.reshape(X[i,:],(24,24))
plt.imshow( img2.T , cmap="gray")
```

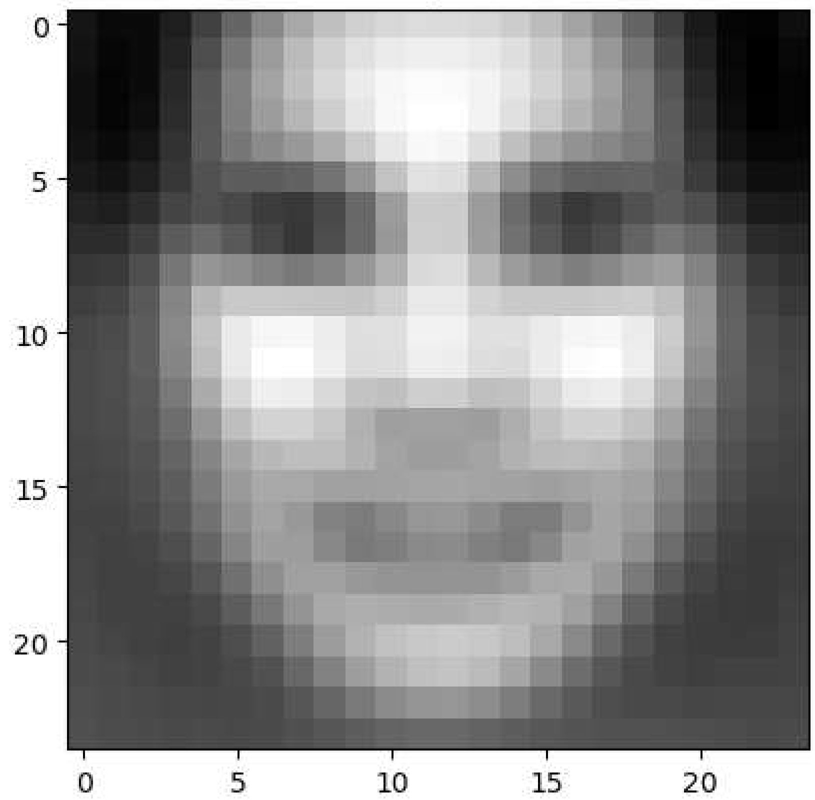
Out[6]: <matplotlib.image.AxesImage at 0x14923f0fb20>





In [7]:
```
img = np.reshape(mu,(24,24))
plt.imshow( img.T , cmap="gray")
plt.title("average face")
```

Out[7]: Text(0.5, 1.0, 'average face')



In [8]:
```
from scipy.linalg import svd

U, s, V = svd(X0, full_matrices=False)
W = np.dot(U, np.diag(s))

print ('Shape of W = (%d, %d)' % W.shape, 'Shape of V = (%d, %d)' % V.shape)
```

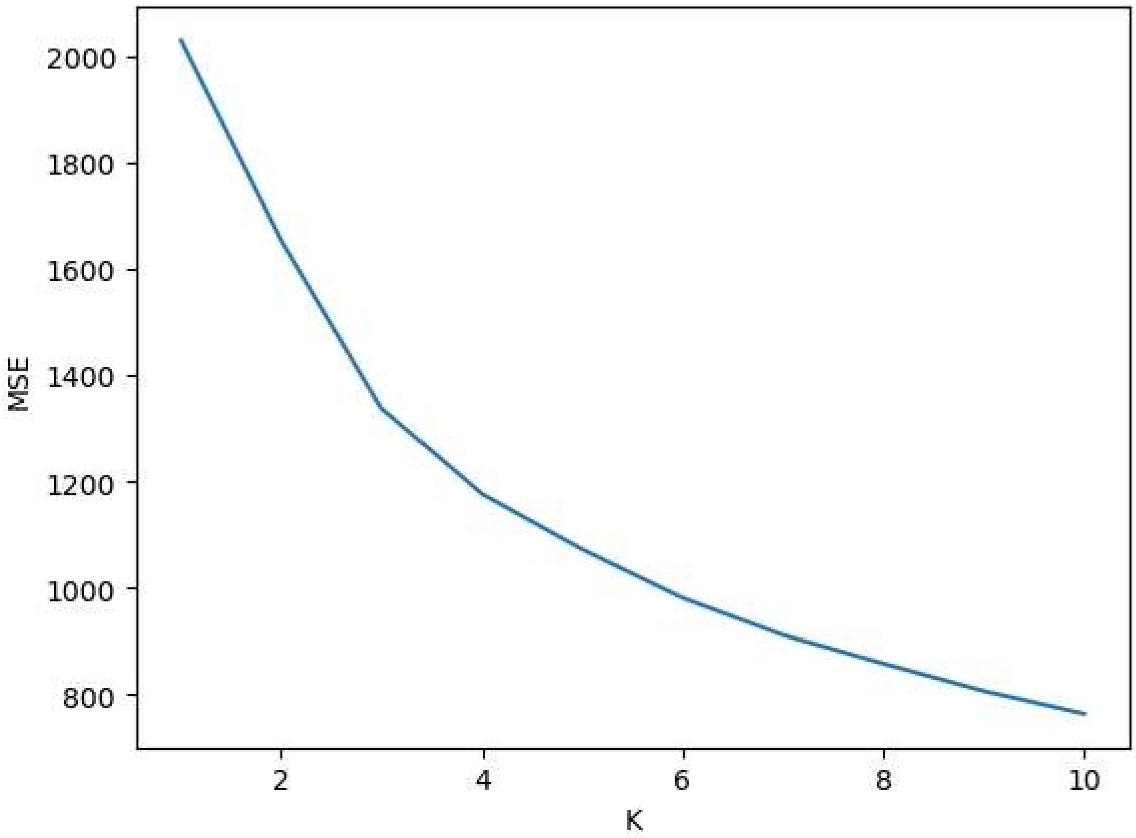
Shape of W = (4916, 576) Shape of V = (576, 576)

In [24]:
```
def cal_mse_of_k(k):
    X0k  = np.dot(W[:,:k],V[:k,:])
    mse = np.mean((X0-X0k)**2)
    return mse
```

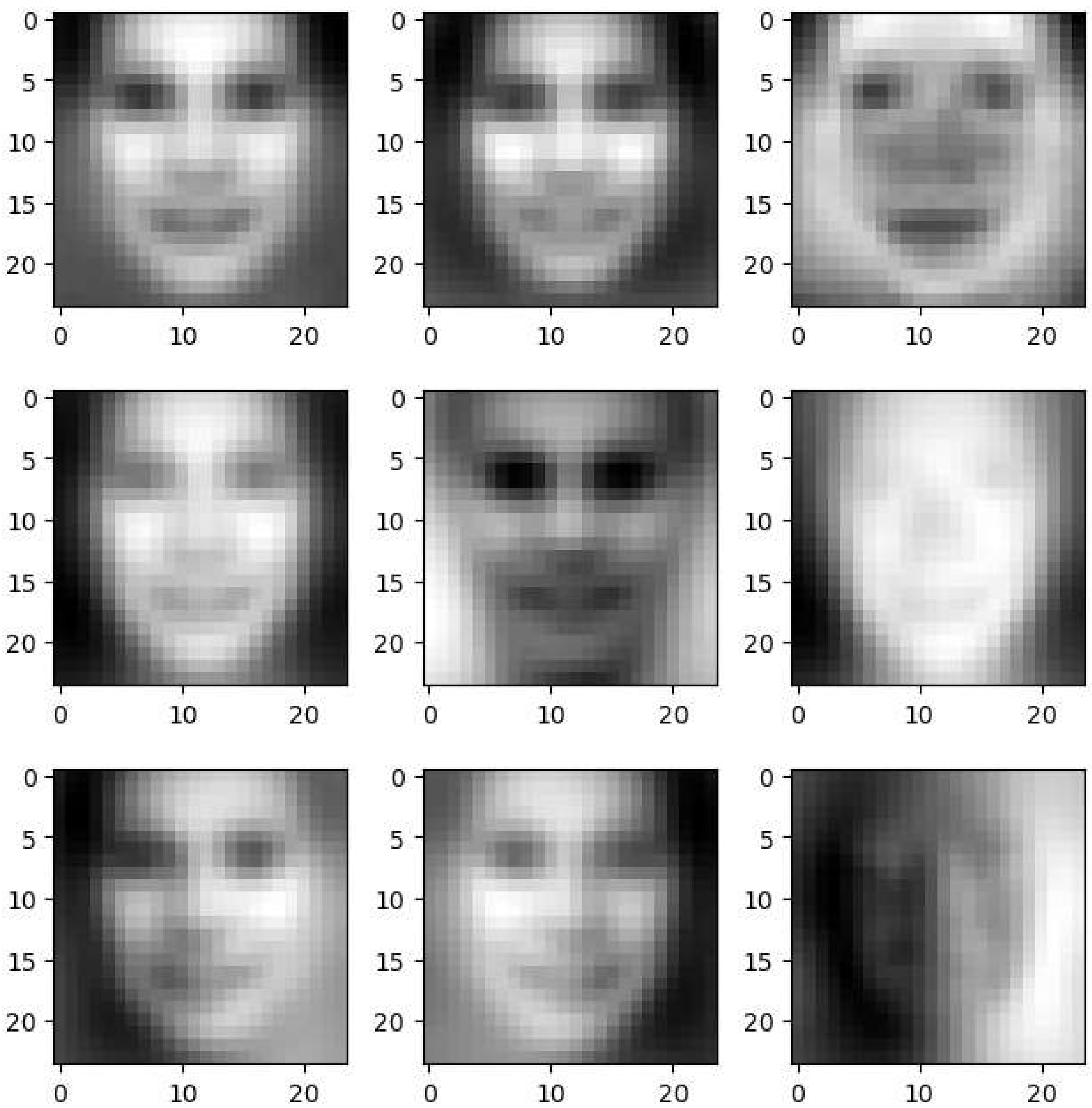
In [25]:
```
k_list = []
Mse = []
for i in range(1,11):
    k_list.append(i)
    Mse.append(cal_mse_of_k(i))
```

In [29]:
```
plt.plot(k_list,Mse)
plt.xlabel("K")
plt.ylabel("MSE")
```

Out[29]: Text(0, 0.5, 'MSE')

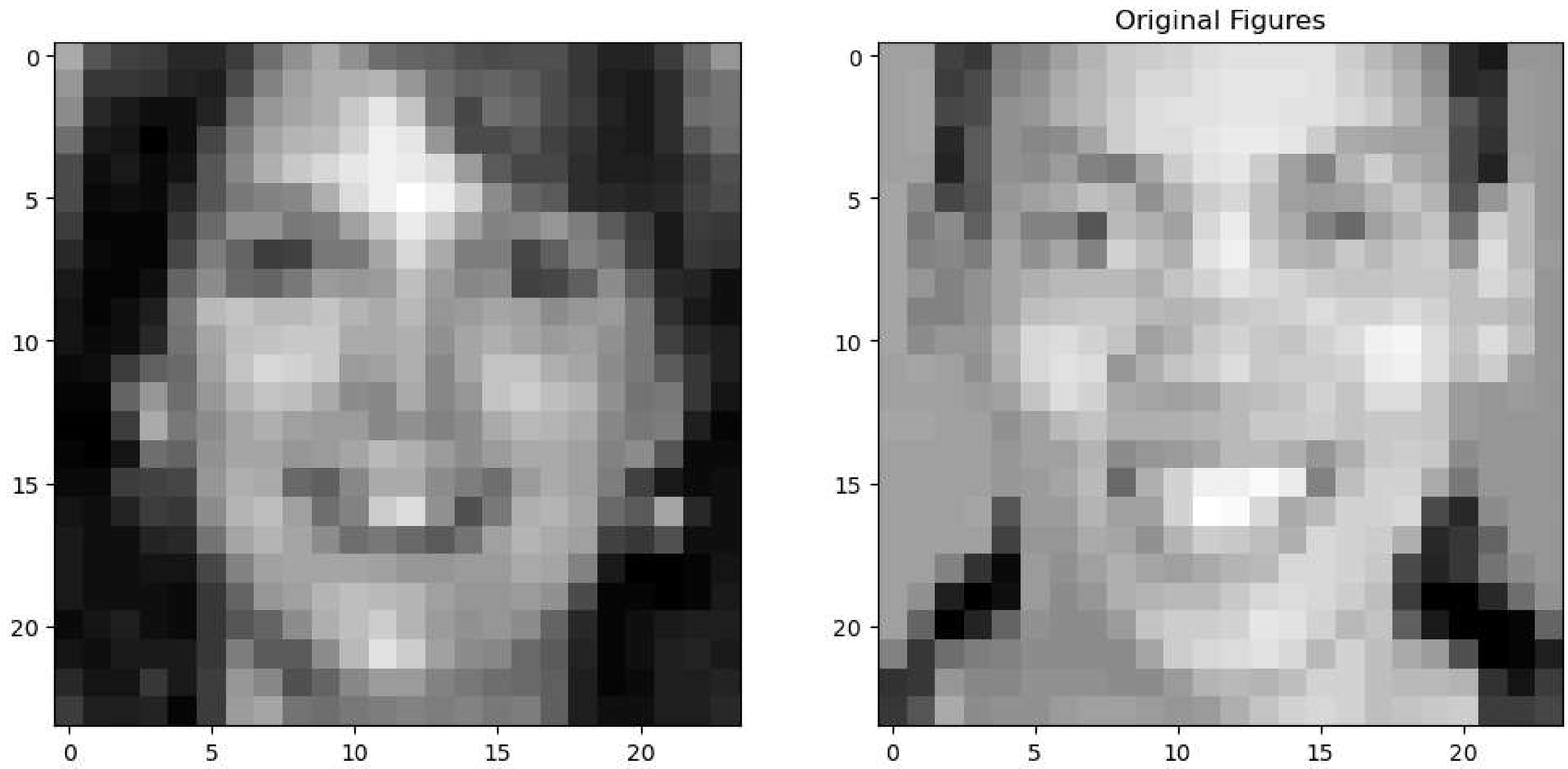


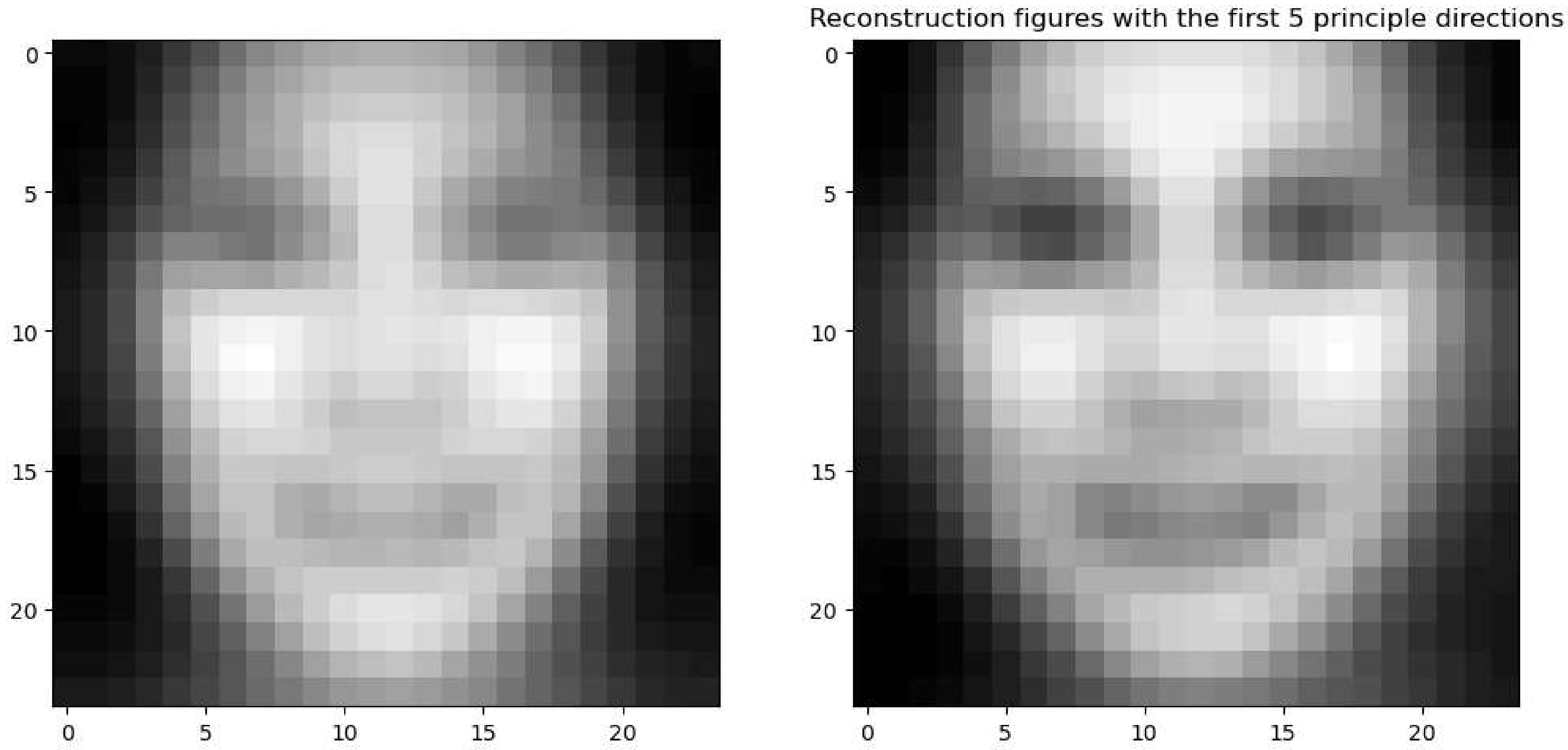
In [51]:
```python
for j in range(3):
    a = 2 * np.median(np.abs(W[:, j]))  # The scalar

    # Compting the "direction" as a function of the mean.
    p1 = mu - a * V[j,:]
    p2 = mu + a * V[j,:]
    fig, (ax1, ax2,ax3) = plt.subplots(1, 3)

    img = np.reshape(p1,(24,24))
    ax1.imshow( img.T , cmap="gray")

    img2 = np.reshape(p2,(24,24))
    ax2.imshow( img2.T , cmap="gray")

    img3 = np.reshape(p1-p2,(24,24))
    ax3.imshow( img3.T , cmap="gray")
    plt.tight_layout()

    plt.show()
```

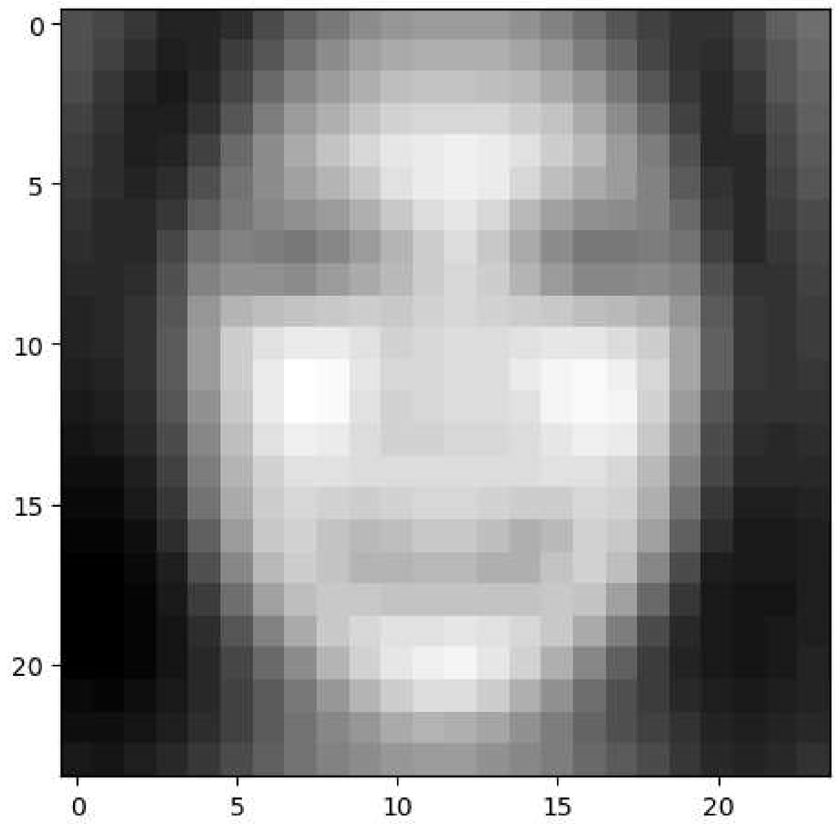


In [73]:
```python
idx = np.random.choice(X.shape[0], 2, replace=False)
idx = [88, 14]
```

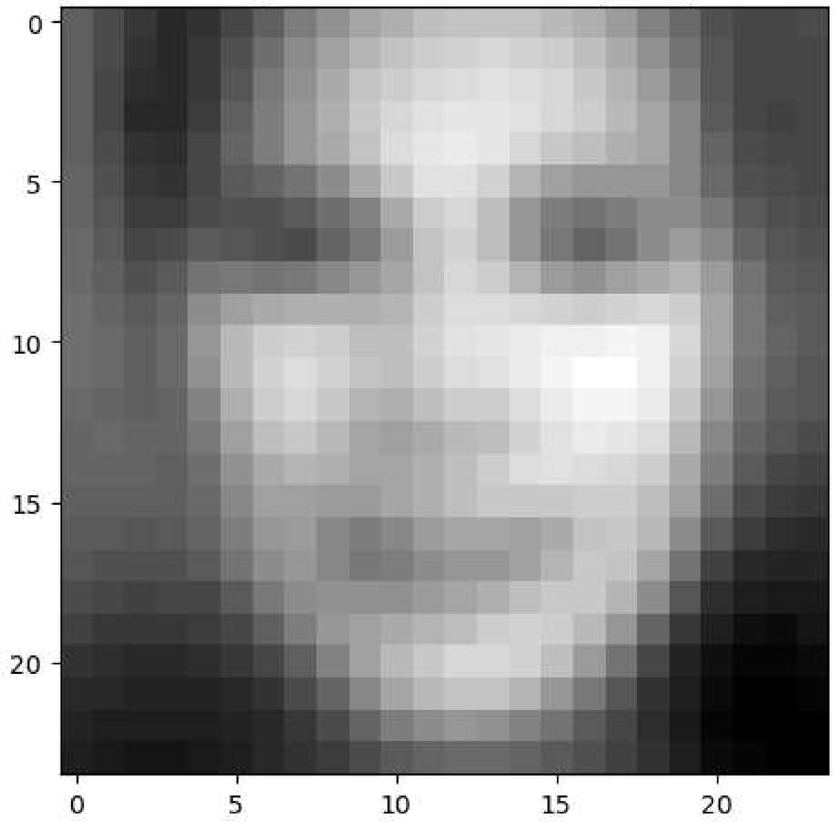
```python
k_list = [5,10,50,100]

f, ax = plt.subplots(1, 2, figsize=(12, 15))
plt.title("Original Figures")

for j in range(len(idx)):

        i = idx[j]

        img = X[i]  # DON'T FORGET TO ADD THE MU

        img = np.reshape(img,(24,24))  # reshape flattened data into a 24*24 patch

        # We've seen the imshow method in the previous discussion :)
        ax[j].imshow( img.T , cmap="gray")

for k in k_list:
    f, ax = plt.subplots(1, 2, figsize=(12, 15))
    plt.title("Reconstruction figures with the first {} principle directions".format(k))
    for j in range(len(idx)):

        i = idx[j]

        img = np.dot(W[i, :k], V[:k]) + mu  # DON'T FORGET TO ADD THE MU

        img = np.reshape(img,(24,24))  # reshape flattened data into a 24*24 patch

        # We've seen the imshow method in the previous discussion :)
        ax[j].imshow( img.T , cmap="gray")
```

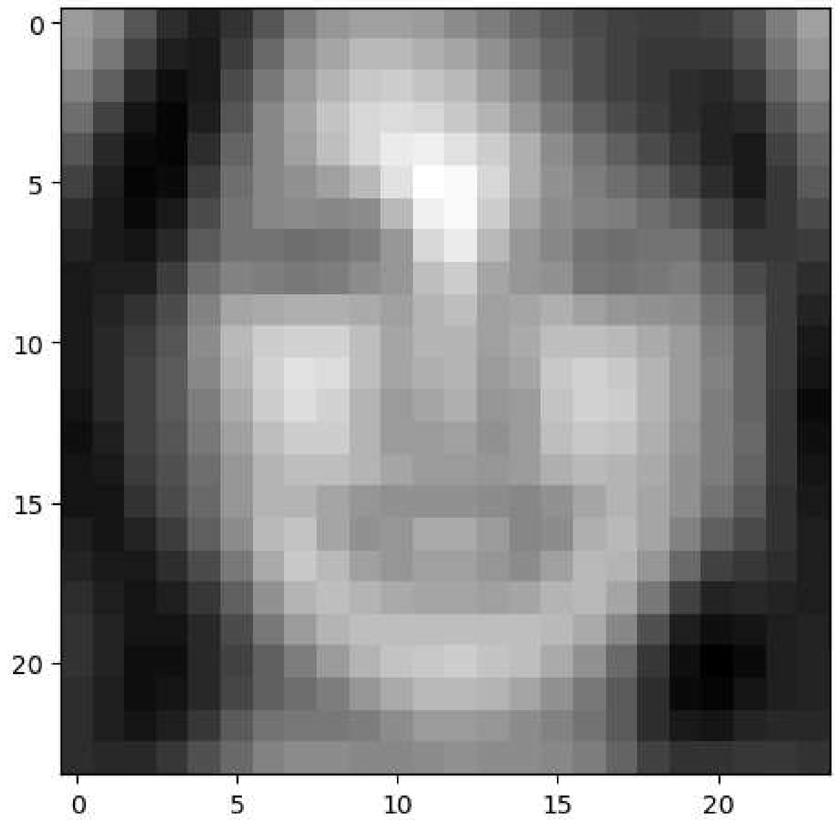




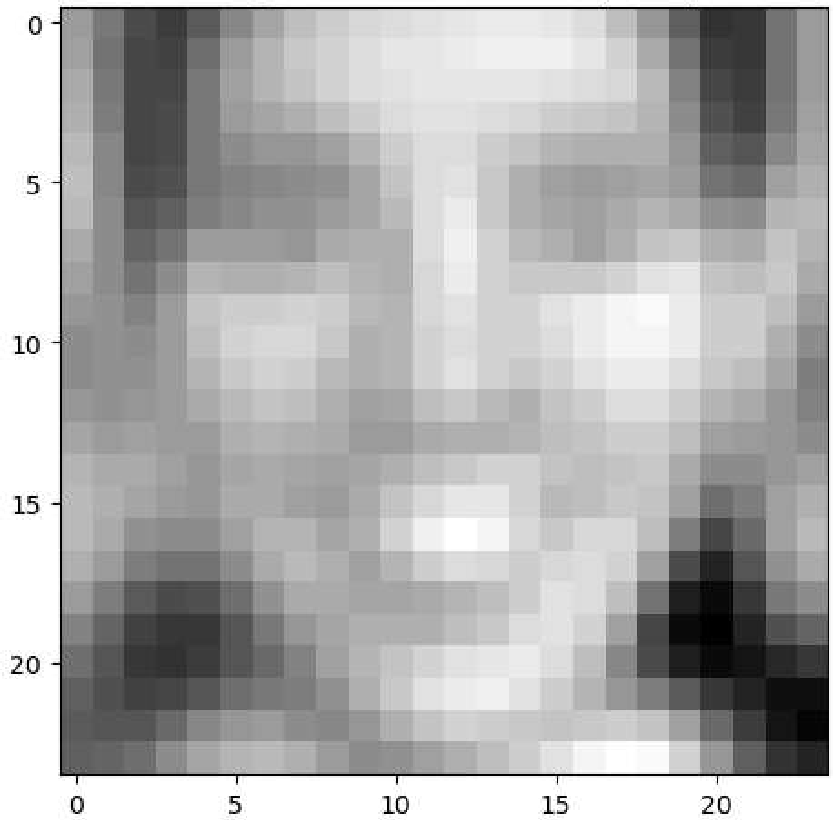


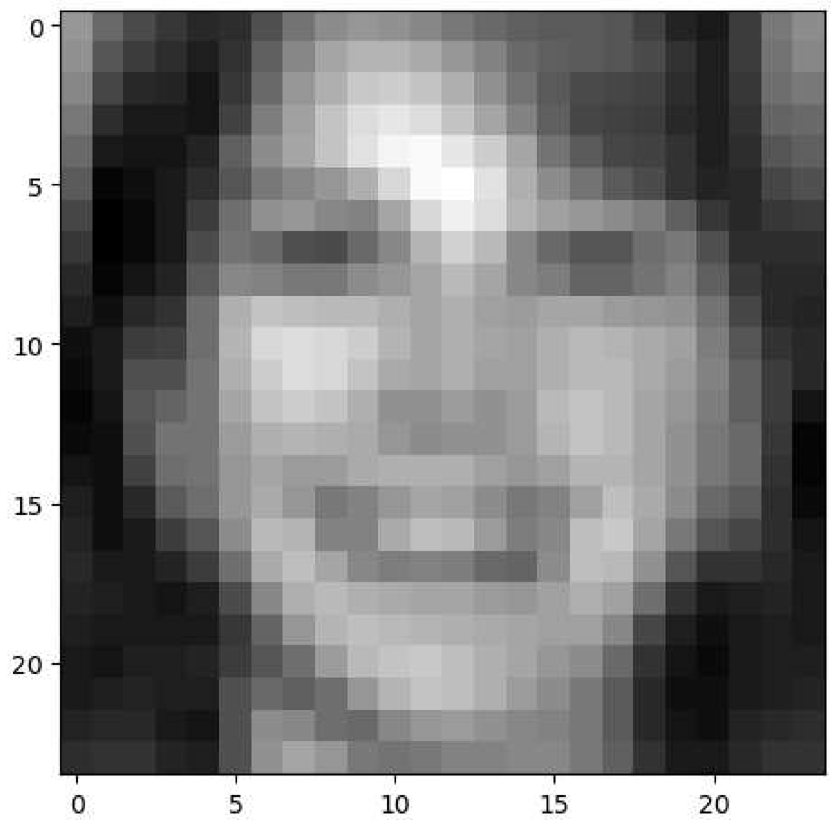
Reconstruction figures with the first 10 principle directions





Reconstruction figures with the first 50 principle directions

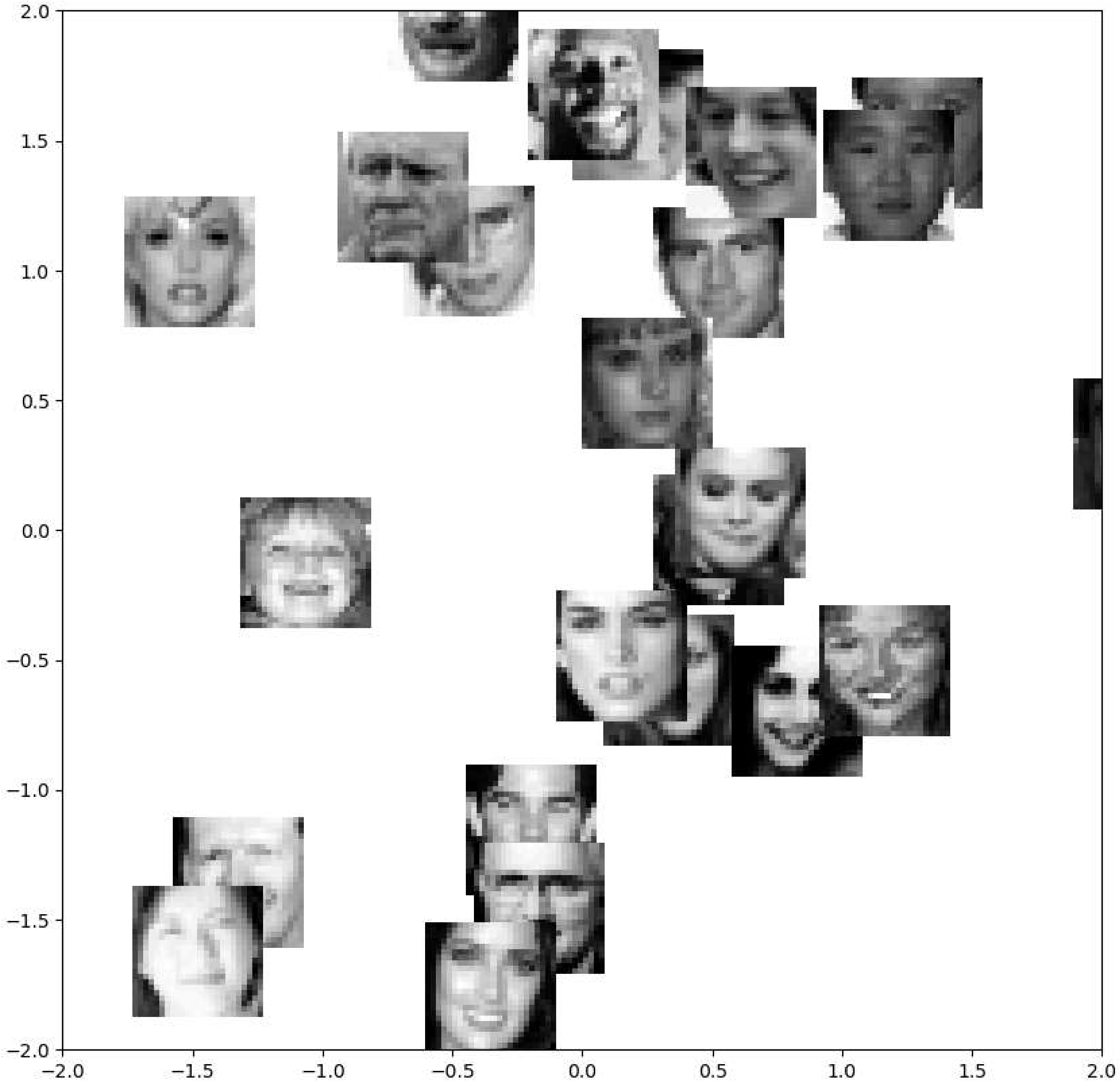




Reconstruction figures with the first 100 principle directions

In [76]:
```
import mltools as ml
f, ax = plt.subplots(1, 1, figsize=(15, 10))

idx = np.random.choice(X.shape[0], 25, replace=False)

coord, params = ml.transforms.rescale(W[:,0:2])
for i in idx:
    loc = (coord[i, 0], coord[i, 0] + 0.5, coord[i,1], coord[i, 1] + 0.5)
```

```
    img = np.reshape(X[i,:], (24,24))
    ax.imshow(img.T , cmap="gray", extent=loc)

ax.axis([-2, 2, -2, 2])
plt.show()
```



Statement of Collaboration
I finished my homework by myself.

```
In [ ]:
```