In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
ir= pd.read_csv("C:\\Users\\dhima\\anaconda3\\6th Week\\Iris.csv")
ir1=ir.copy()
ir1
```

Out[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [2]:
```
ir1.isnull().sum()
```

Out[2]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [3]:
```
y_true=ir1['Species']
y_true
```

Out[3]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

Ques 1. Apply PCA and select first two directions to convert the data in to 2D. (Exclude the attribute “Species” for PCA)

In [4]:
```
ir2=ir1.iloc[0:,1:5]
ir2
```

Out[4]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [5]:
```
#from sklearn.preprocessing import StandardScaler
#scaler = StandardScaler()
#ir2_ss=ir2.copy()
#StandardScaler().fit(ir2_ss)
#ir2_ss=scaler.fit_transform(ir2_ss)
#ir2_ss
```

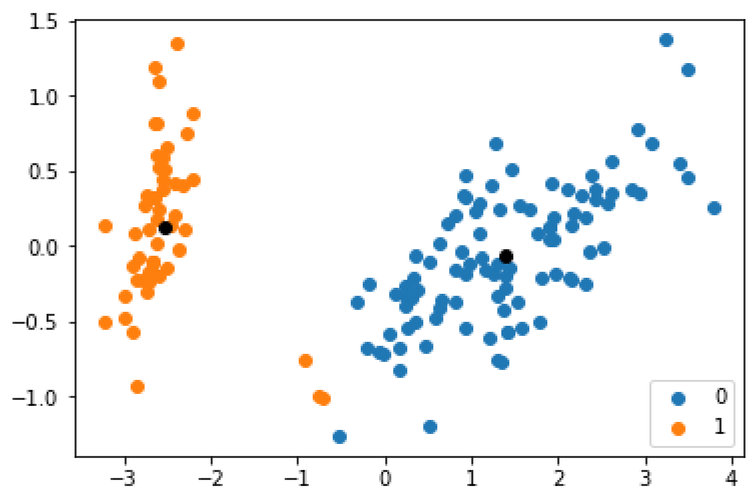
In [6]:
```
from sklearn.decomposition import PCA

pca = PCA(2)

df = pca.fit_transform(ir2)
pca.n_components_
df.shape
```

Out[6]:
```
(150, 2)
```

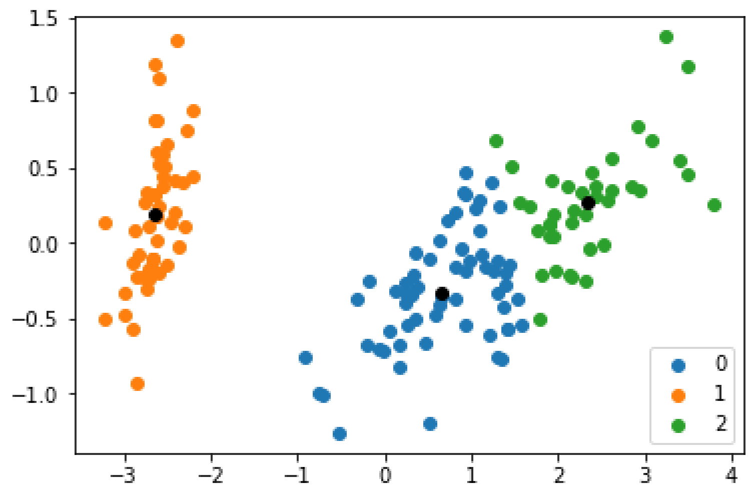
2. Apply K-means (K=3) clustering on the reduced data. Plot the data points in these clusters (use different colors for each cluster). Obtain the sum of squared distances of samples to their closest cluster center. (Use kmeans.fit to train the model and kmeans.labels_ to obtaine the cluster labels).

In [7]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

clus=[2,3,4,5,6,7]

for y,z in enumerate(clus):

    kmeans = KMeans(n_clusters=z, max_iter=600, algorithm = 'auto')
    #predict the labels of clusters.
    label = kmeans.fit_predict(df)
    print(label)

    # Final locations of the centroid
    print(kmeans.cluster_centers_)

    #filter rows of original data
    #filtered_label0 = df[label == 0]
    #filtered_label1 = df[label == 1]
    #filtered_label2 = df[label == 2]

    #plotting the results
    #plt.scatter(filtered_label0[:,0] , filtered_label0[:,1] , color = 'blue')
    #plt.scatter(filtered_label1[:,0] , filtered_label1[:,1] , color = 'black')
    #plt.scatter(filtered_label2[:,0] , filtered_label2[:,1] , color = 'red')


    #Getting the Centroids
    centroids = kmeans.cluster_centers_
    centroids

    #Getting unique labels
    u_labels = np.unique(label)

    #plotting the results:
    for i in u_labels:
        plt.scatter(df[label == i , 0] , df[label == i , 1] , label = i)
    plt.scatter(centroids[:,0] , centroids[:,1] , color = 'k')
    plt.legend()
    plt.show()
```

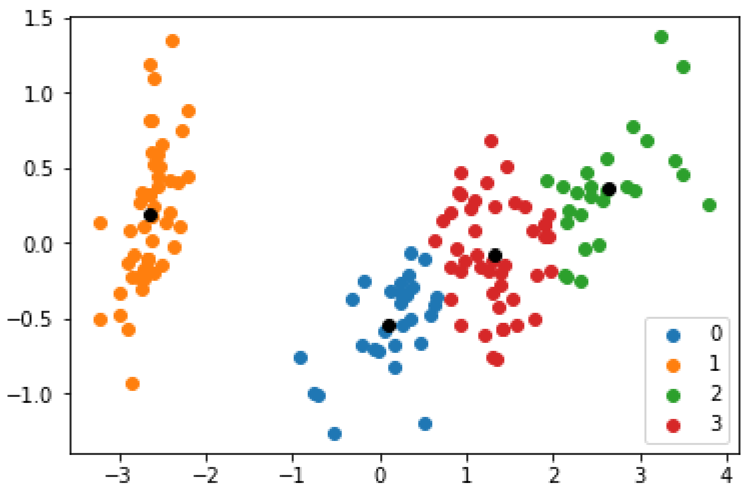
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0]
[[ 1.38566031 -0.0697412 ]
 [-2.53601981  0.12763956]]
```



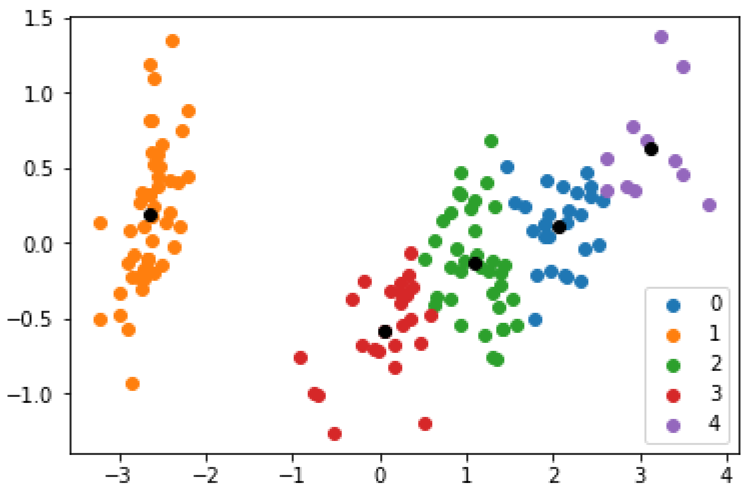
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
[[ 0.66443351 -0.33029221]
 [-2.64084076  0.19051995]
 [ 2.34645113  0.27235455]]
```



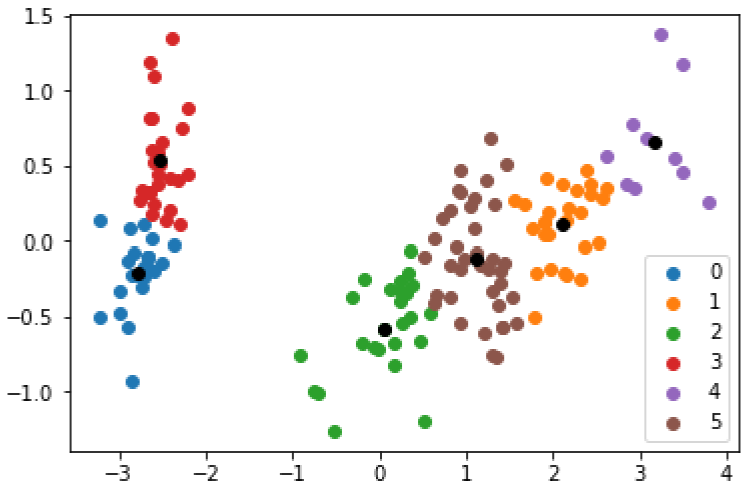
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 0 3 0 3 0 3 0 0 0 0 3 0 3 0 0 3 0 3 0 3 3
 3 3 3 3 3 0 0 0 0 3 0 3 3 3 0 0 0 3 0 0 0 0 0 3 0 0 2 3 2 3 2 2 0 2 2 2 3
 3 2 3 3 3 3 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 3 2 2 3 3 2 2 2 3 2 2 3 3 3
 3 3]
[[ 0.10568813 -0.54728468]
 [-2.64084076  0.19051995]
 [ 2.63931015  0.35508094]
 [ 1.31436318 -0.07465502]]
```



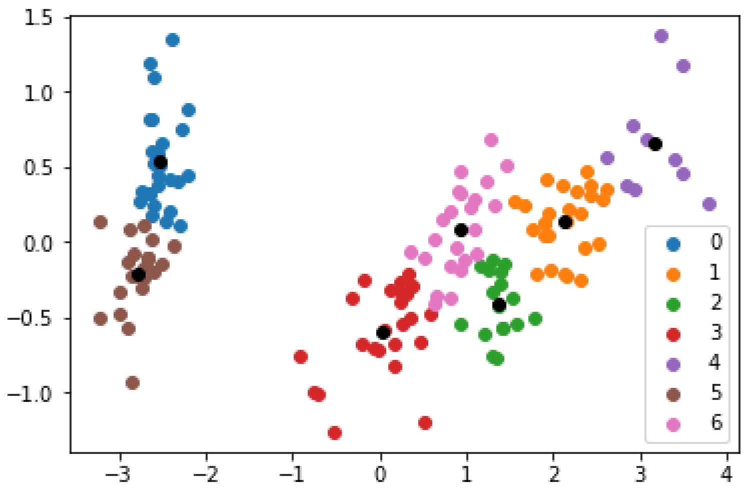
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 0 3 2 2 2 3 2 3 3 2 3 2 3 2 2 3 2 3 2 3 2 2
 2 2 2 0 2 3 3 3 3 2 3 2 2 2 3 3 3 2 3 3 3 3 3 2 3 3 0 2 4 0 0 4 3 4 0 4 0
 0 0 2 2 0 0 4 4 2 0 2 4 2 0 4 2 2 0 0 4 4 0 2 0 4 0 0 2 0 0 0 2 0 0 0 2 0
 0 2]
[[ 2.06727994  0.11279799]
 [-2.64084076  0.19051995]
 [ 1.09911077 -0.13540537]
 [ 0.04592561 -0.57809549]
 [ 3.12820371  0.62644023]]
```



```
[3 0 0 0 3 3 0 3 0 0 3 0 0 0 3 3 3 3 3 3 3 3 0 3 0 0 3 3 3 0 0 3 3 3 0 0 3
 0 0 3 3 0 0 3 3 0 3 0 3 0 5 5 5 2 5 5 5 2 5 2 2 5 2 5 2 5 5 2 5 2 5 2 5 5
 5 5 5 1 5 2 2 2 2 5 2 5 5 5 2 2 2 5 2 2 2 2 2 5 2 2 1 5 1 1 1 4 2 4 1 4 1
 1 1 5 5 1 1 4 4 5 1 5 4 5 1 4 5 5 1 1 4 4 1 5 1 4 1 1 5 1 1 1 5 1 1 1 5 1
 1 5]
[[-2.78123098 -0.20587391]
 [ 2.10996229  0.10678856]
 [ 0.04592561 -0.57809549]
 [-2.52124909  0.5281888 ]
 [ 3.17937563  0.65489073]
```



```
[0 5 5 5 0 0 5 0 5 5 0 5 5 5 0 0 0 0 0 0 0 0 5 0 5 5 0 0 0 5 5 0 0 0 5 5 0
 5 5 0 0 5 5 0 0 5 0 5 0 5 6 6 6 3 6 6 6 3 6 3 3 6 3 6 3 6 6 3 2 3 6 6 2 6
 6 6 6 1 6 3 3 3 3 2 3 6 6 6 3 3 3 6 3 3 3 3 3 6 3 3 1 2 1 1 1 4 3 4 1 4 1
 1 1 2 2 1 1 4 4 2 1 2 4 2 1 4 2 2 1 1 4 4 1 2 2 4 1 1 2 1 1 1 2 1 1 1 2 1
 1 2]
[[-2.52124909  0.5281888 ]
 [ 2.12266699  0.13018292]
 [ 1.3596994  -0.41706022]
 [ 0.03297307 -0.59939848]
 [ 3.17937563  0.65489073]
 [-2.78123098 -0.20587391]
 [ 0.91968246  0.08764898]]
```



Obtain the purity score for the Kmeans clustering (K=3).

In [8]:
```
import numpy as np
from sklearn import metrics

def purity_score(y_true, y_pred):
    # compute contingency matrix (also called confusion matrix)
    contingency_matrix = metrics.cluster.contingency_matrix(y_true, y_pred)
    # return purity
    return np.sum(np.amax(contingency_matrix, axis=0)) / np.sum(contingency_matri
```

In [9]:
```
kmeans = KMeans(n_clusters=z, max_iter=600, algorithm = 'auto')
#predict the labels of clusters.
label3 = kmeans.fit_predict(df)

purity_score(y_true,label3)
```

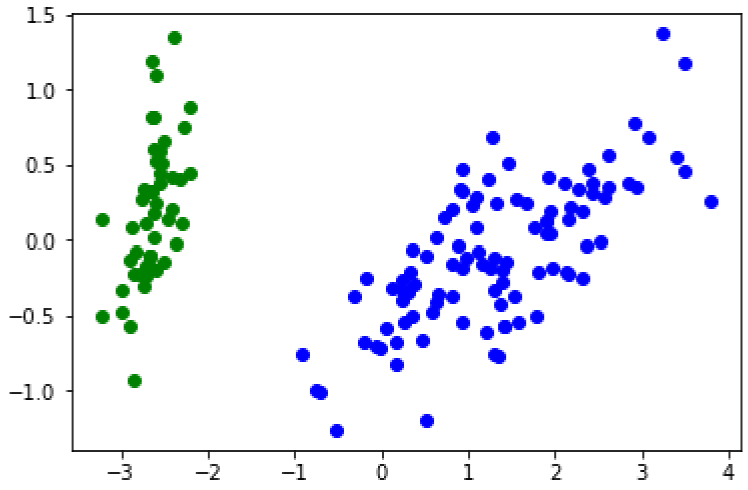
Out[9]:
```
0.9066666666666666
```

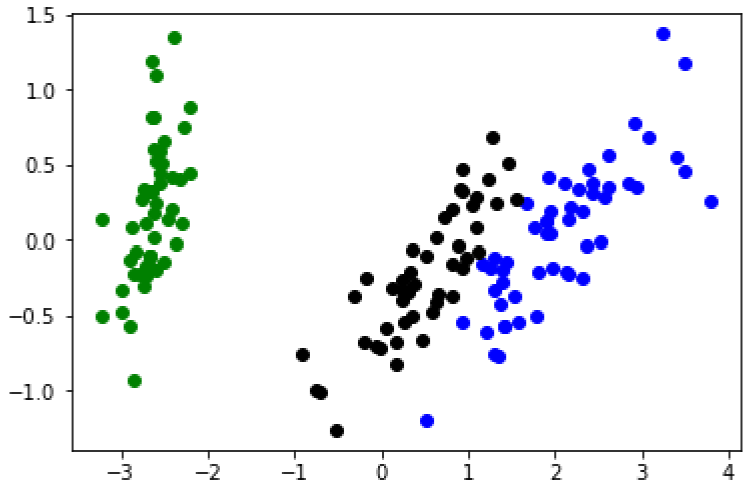
3. Build a GMM with 3 components (use GMM.fit) on the reduced data. Use this GMM to cluster the data points (Use GMM.predict). Plot the points in these clusters.

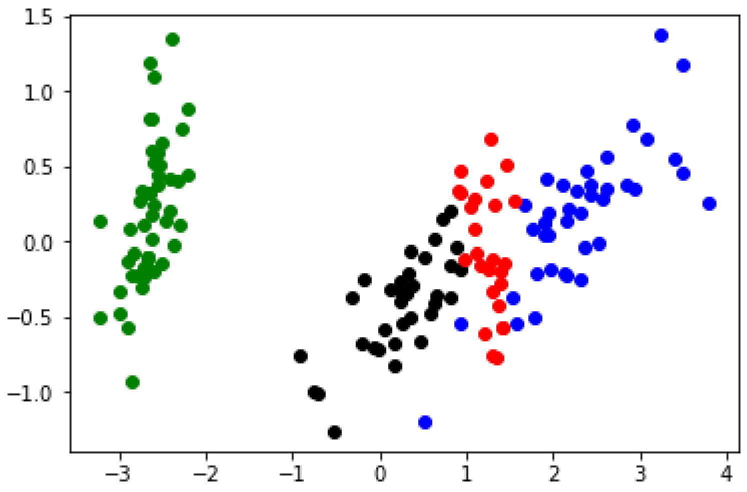
In [10]:
```
from sklearn.mixture import GaussianMixture

gmm = GaussianMixture(n_components=3)
gmm.fit(df)

print(gmm.means_)
print('\n')
print(gmm.covariances_)
```

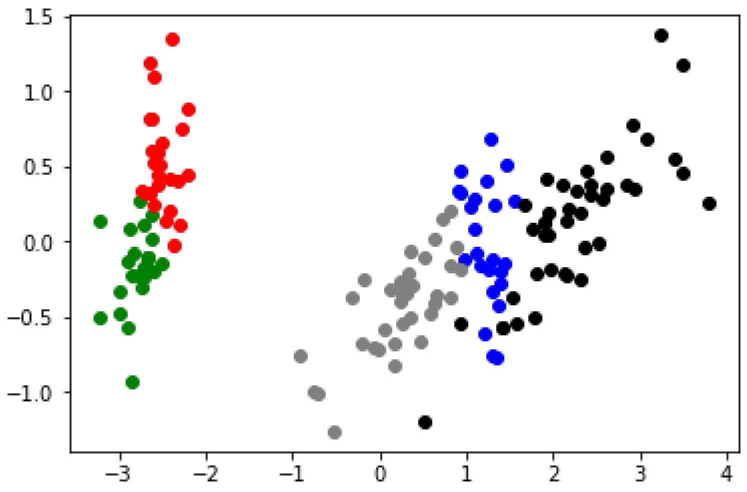
```
[[ 0.5234348  -0.22332857]
 [-2.64084076  0.19051995]
 [ 2.03944957  0.02028171]]


[[[0.37131895 0.21751015]
  [0.21751015 0.18683677]]

 [[0.04777148 0.05590782]
  [0.05590782 0.21472456]]

 [[0.55435437 0.29321815]
  [0.29321815 0.23386779]]]
```

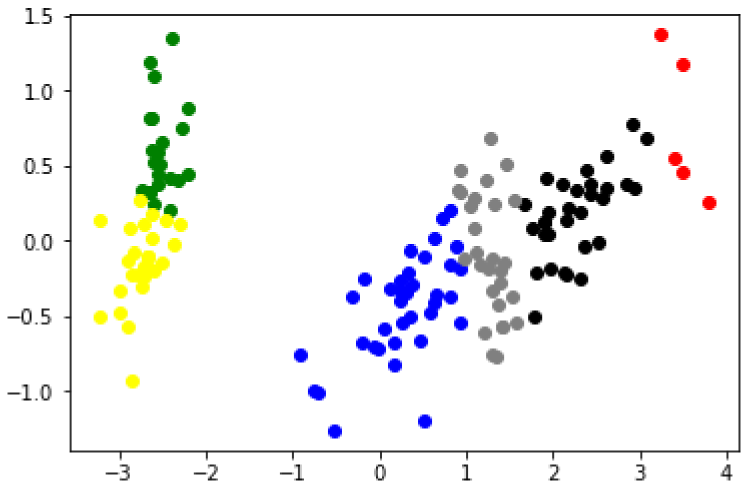
In [11]:
```
for z in range(2,8,1):

    gmm = GaussianMixture(n_components=z)
    gmm.fit(df)

    #predictions from gmm
    labels = gmm.predict(df)
    frame = pd.DataFrame(df)
    frame['cluster'] = labels
    frame.columns = ['SepalLengthCm', 'SepalWidthCm', 'cluster']

    color=['blue','green','black','red','grey','yellow','cyan']
    for k in range(0,z):
        data = frame[frame["cluster"]==k]
        plt.scatter(data["SepalLengthCm"],data["SepalWidthCm"],c=color[k])
    plt.show()
```

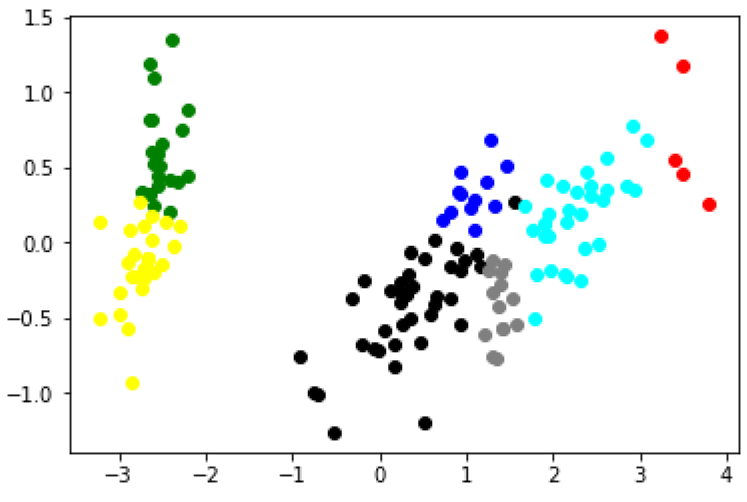





1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4


1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4





4. Obtain the purity score for the GMM clustering (K=3).

In [12]:
```
gmm = GaussianMixture(n_components=3)
gmm.fit(df)

#predictions from gmm
labels3 = gmm.predict(df)
purity_score(y_true,labels3)
```

Out[12]:
```
0.98
```

In [ ]: