In [37]:
```
import csv
import numpy as np
import pandas as pd
import scipy as sp
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
from sklearn.metrics import adjusted_rand_score
from sklearn.metrics import adjusted_mutual_info_score
from sklearn.metrics import normalized_mutual_info_score
from sklearn import datasets
from sklearn.datasets.base import Bunch
# importing clustering algorithms
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from sklearn.cluster import SpectralClustering
from  seaborn import clustermap
from sklearn.metrics import silhouette_samples

from pandas.api.types import is_numeric_dtype

from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import VarianceThreshold
```

In [2]:
```
n_samples = 1500
random_state = 10

import numpy as np

#Plotting packages
import matplotlib.pyplot as plt
import seaborn as sns

#Classification Algorithms
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

#Ensemble Methods
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import BaggingRegressor
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.ensemble import AdaBoostClassifier

#Mlxtend for visualizing classification decision boundaries
from mlxtend.plotting import plot_decision_regions

from sklearn.decomposition import PCA
from scipy.linalg import svd
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

In [3]:
```
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
from math import isnan
def hopkins(X):
    n=X.shape[0]#rows
    d=X.shape[1]#cols
    p=int(0.1*n)#considering 10% of points
    nbrs=NearestNeighbors(n_neighbors=1).fit(X)

    rand_X=sample(range(0,n),p)
    uj=[]
    wj=[]
    for j in range(0,p):
        u_dist,_=nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),
        uj.append(u_dist[0][1])#distances to nearest neighbors in random data
        w_dist,_=nbrs.kneighbors(X[rand_X[j]].reshape(1,-1),2,return_distance
        wj.append(w_dist[0][1])#distances to nearest neighbors in real data
    H=sum(uj)/(sum(uj)+sum(wj))
    if isnan(H):
        print(uj,wj)
        H=0

    return H
```

In [9]:
```
def silhouette(X,labels):
    n_clusters=np.size(np.unique(labels));
    sample_silhouette_values=silhouette_samples(X,labels)
    y_lower=10
    for i in range(n_clusters):
        ith_cluster_silhouette_values=sample_silhouette_values[labels==i]
        ith_cluster_silhouette_values.sort()
        size_cluster_i=ith_cluster_silhouette_values.shape[0]
        y_upper=y_lower+size_cluster_i
        color=cm.nipy_spectral(float(i)/n_clusters)
        plt.fill_betweenx(np.arange(y_lower,y_upper),0,ith_cluster_silhouette
        plt.text(-0.05,y_lower+0.5*size_cluster_i,str(i))#Compute the new y_l
        y_lower=y_upper+10# 10 for the 0 samples
    plt.title("Silhouette plot for K-Means Clustering.")
    plt.xlabel("Silhouette coefficient values")
    plt.ylabel("Cluster label")
    plt.show()
```

In [10]:
```
from scipy.special import comb
def rand_index(S, T):

    Spairs = comb(np.bincount(S), 2).sum()
    Tpairs = comb(np.bincount(T), 2).sum()

    A = np.c_[(S, T)]

    f_11 = sum(comb(np.bincount(A[A[:, 0] == i, 1]), 2).sum()
               for i in set(S))

    f_10 = Spairs - f_11
    f_01 = Tpairs - f_11
    f_00 = comb(len(A), 2) - f_11 - f_10 - f_01
    return (f_00 + f_11) / (f_00 + f_01 + f_10 + f_11)
```

In [11]:
```
values = pd.read_csv('https://raw.githubusercontent.com/snapponder/data/main/
types = pd.read_csv('https://raw.githubusercontent.com/snapponder/data/main/t
original = pd.read_csv('https://raw.githubusercontent.com/snapponder/data/mai

values.drop(columns=values.columns[values.columns.str.contains('Group', case=
values.head()

types.drop(columns=types.columns[types.columns.str.contains('unnamed', case=F
types.head()
fixed = np.arange(24)
Clusters = []
for type in types.values:
    if(type == "Proneural"):
        Clusters.append(0)
    elif(type == "Mesenchymal"):
        Clusters.append(1)
    elif(type == "G-CIMP"):
        Clusters.append(2)
    elif(type == "Classical"):
        Clusters.append(3)
    elif(type == "Neural"):
        Clusters.append(4)
    else:
        Clusters.append(3)
print(Clusters)
origclust = []
original.drop(columns=original.columns[original.columns.str.contains('tcga',
for cluster in original.values:
    origclust.append(cluster[0])
origclust.append(3)
print(original.values)
a = np.asarray(Clusters)
print(a)
print(origclust)
b = np.asarray(origclust)
print(b)

LABEL_COLOR_MAP = {0 : 'r',
                   1 : 'k',
                   2 : 'b',
                   3 : 'g',
                   4 : 'c',
                   5 : 'm',
                   6 : 'tab:brown',
                   7 : 'y',
                  }
```

```
[5]
[2]
[5]

[5]
[5]
```

```
[3]
[3]
[3]
[3]
[5]
[5]
[4]
[5]
[5]
[3]
[3]
[4]
[1]
[5]
```

In [12]:
```
c = np.asarray([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

print(c)
d = np.asarray([0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

e = np.asarray([1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2

f = np.asarray([3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3
```

```
[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 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 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]
```

In [13]:
```
Dataset = Bunch(data=values, target=types)
Glio_X = values.values
Glio_y = a
print(Glio_X.shape)
print(Glio_y.shape)

Glio_y_color = [LABEL_COLOR_MAP[l] for l in Glio_y]
```

```
(152, 200)
(152,)
```

In [14]:
```
print(Glio_X)
```

```
[[ 1.13  1.51  0.04 ... -0.11  0.11 -0.11]
 [-1.29 -0.81 -0.07 ...  1.21  0.99  0.98]
 [-0.73 -0.74 -1.19 ... -0.65 -0.18 -0.66]
 ...
 [ 1.56  0.58  0.24 ... -1.07 -0.1  -1.29]
 [ 1.35  0.93 -0.17 ... -1.02 -0.15 -0.42]
 [ 0.54  0.71  0.13 ... -1.48 -0.54 -1.21]]
```

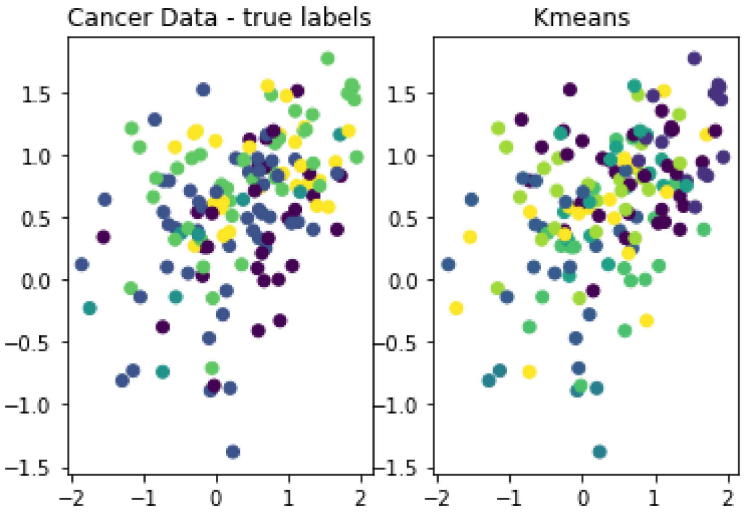
In [ ]:
```
sns.heatmap(Glio_X)
```

In [ ]:
```
pca = PCA(151)  # project from 64 to 2 dimensions
projected = pca.fit_transform(Glio_X.data)
sns.heatmap(projected)
```

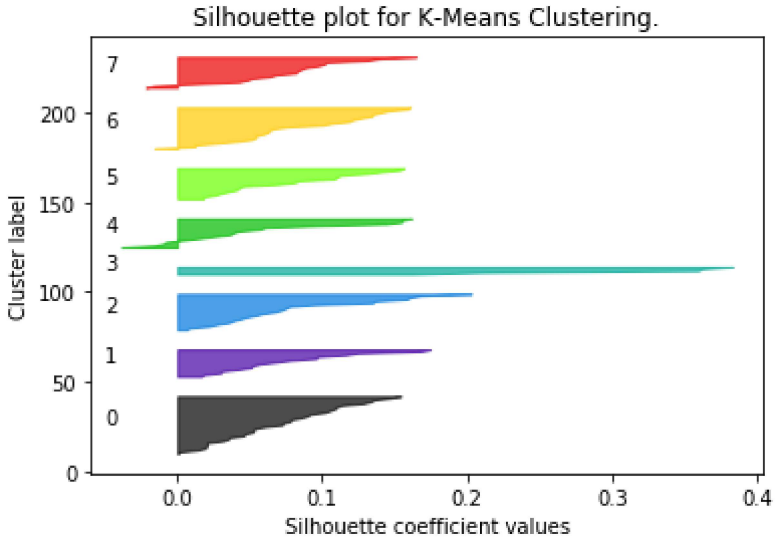
In [ ]:
```
score = np.zeros(151);
for i in range(2,151):
     kmeans = KMeans(n_clusters=i, random_state=random_state); #Initializing
     kmeans.fit_predict(Glio_X) #Clustering using KMeans
     score[i] = -kmeans.score(Glio_X) #Computing SSE
     print("SSE for k=",i,":", round(score[i],2)) #Printing SSE

plt.plot(range(2,151),score[2:151])
plt.scatter(range(2,151),score[2:151])
plt.xlabel('k')
plt.ylabel('SSE')
plt.title('SSE for different values of k')
plt.show()
```

In [43]:
```
kmeans= KMeans(n_init=151,  random_state=random_state, n_clusters = 8)
y_pred = kmeans.fit_predict(Glio_X)
print(y_pred)
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Glio_X[:, 0], Glio_X[:, 1],  c=Glio_y) # true clusters
plt.title('Cancer Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Glio_X[:, 0], Glio_X[:, 1], c=y_pred) # KMeans clusters
plt.title('Kmeans ')
plt.show()
```

```
[7 3 7 1 2 6 5 3 0 5 0 0 1 0 0 5 5 2 6 2 2 6 4 5 2 0 0 0 5 6 4 4 6 1 0 2 0
 1 1 5 5 0 2 6 0 6 0 3 5 6 2 7 2 4 0 6 7 7 2 5 0 1 0 5 5 2 0 5 4 4 6 4 2 2
 6 2 1 4 3 0 4 7 4 4 6 7 1 0 7 7 6 0 5 0 7 5 7 0 7 7 5 0 0 0 4 2 1 6 6 0 7
 6 2 4 6 1 0 4 2 2 6 6 1 6 6 0 7 7 3 1 1 0 4 0 7 1 7 2 0 4 5 4 0 2 5 6 1 0
 2 1 6 6]
```



In [44]:
```
silhouette(Glio_X,y_pred)
```



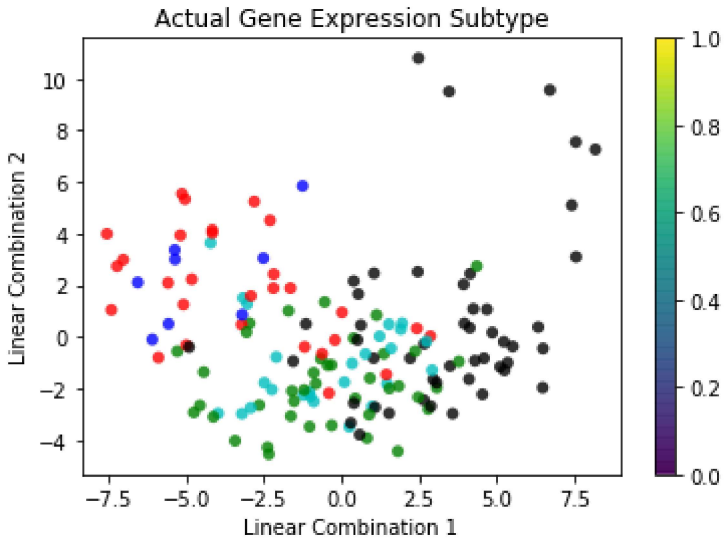
In [45]:
```
print("ARI is " + str(adjusted_rand_score(y_pred, Glio_y)))
print("RI is " + str(rand_index(y_pred, Glio_y)))
print("NMI is " + str(normalized_mutual_info_score(y_pred, Glio_y)))
```

```
ARI is 0.18970666683420748
RI is 0.7493028929940746
NMI is 0.31654524916032284
```

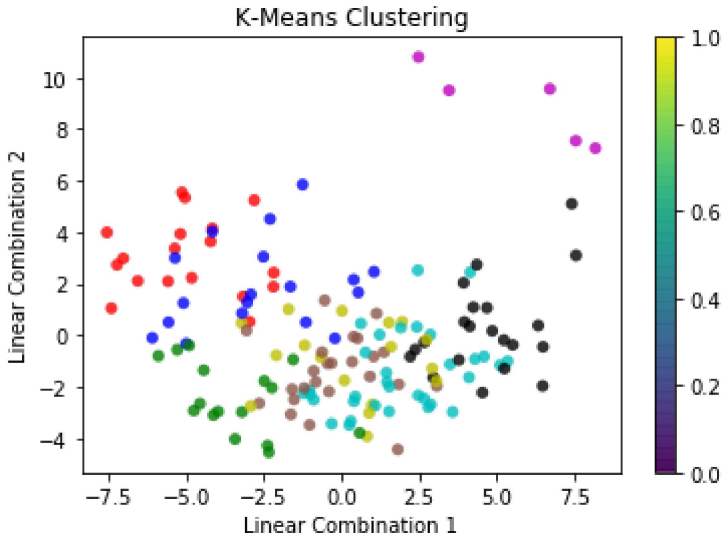
In [46]:
```
print("ARI is " + str(adjusted_rand_score(b, Glio_y)))
print("RI is " + str(rand_index(b, Glio_y)))
print("NMI is " + str(normalized_mutual_info_score(b, Glio_y)))
```

```
ARI is 0.008240111071814426
RI is 0.6229522481700941
NMI is 0.03775350934287529
```

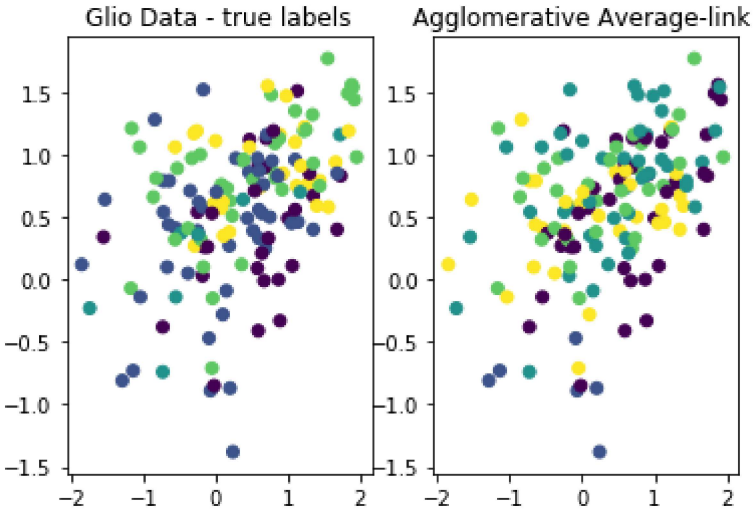
In [48]:
```
pca = PCA(2)  # project from 64 to 2 dimensions
projected = pca.fit_transform(Glio_X.data)

plt.scatter(projected[:, 0], projected[:, 1],
            c=Glio_y_color, edgecolor='none', alpha=0.8,
            cmap=plt.cm.get_cmap('hsv', 10))
plt.xlabel('Linear Combination 1')
plt.ylabel('Linear Combination 2')
plt.title('Actual Gene Expression Subtype')
plt.colorbar();
```



In [52]:
```
LABEL_COLOR_MAP = {0 : 'c',
                   1 : 'g',
                   2 : 'k',
                   3 : 'm',
                   4 : 'y',
                   5 : 'r',
                   6 : 'tab:brown',
                   7 : 'b',
                  }


y_pred_color = [LABEL_COLOR_MAP[l] for l in y_pred]

plt.scatter(projected[:, 0], projected[:, 1],
            c=y_pred_color, edgecolor='none', alpha=0.8,
            cmap=plt.cm.get_cmap('hsv', 10))
plt.xlabel('Linear Combination 1')
plt.ylabel('Linear Combination 2')
plt.title('K-Means Clustering')
plt.colorbar();
```

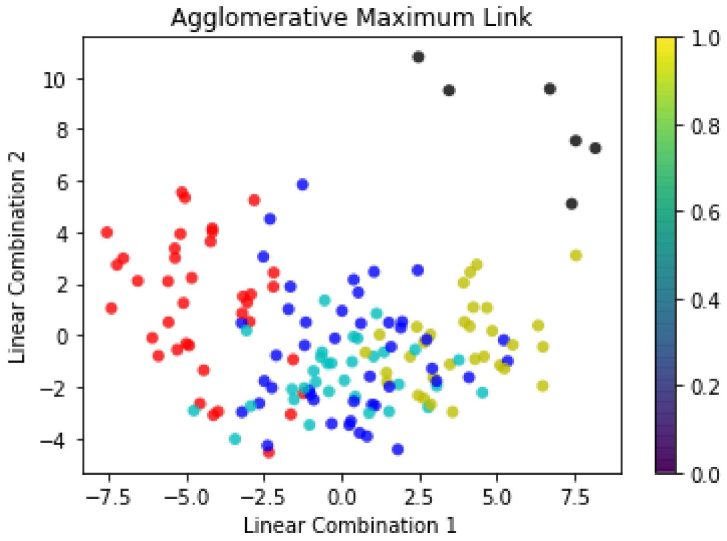


In [40]:
```
n_clusters = 5
average_linkage = AgglomerativeClustering(linkage="complete", n_clusters=n_cl
y_pred = average_linkage.fit_predict(Glio_X)

fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Glio_X[:, 0], Glio_X[:, 1], c=Glio_y) # true clusters
plt.title('Glio Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Glio_X[:, 0], Glio_X[:, 1], c=y_pred) # KMeans clusters
plt.title('Agglomerative Average-link ')
plt.show()

print(rand_index(y_pred, Glio_y))
```




```
0.701289647960962
```

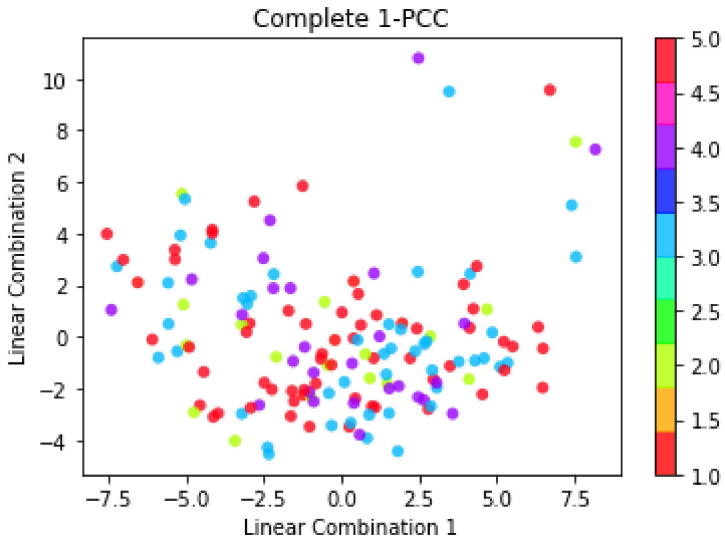
In [41]:
```
LABEL_COLOR_MAP = {0 : 'r',
                   1 : 'k',
                   2 : 'b',
                   3 : 'c',
                   4 : 'y',
                   5 : 'm',
                   6 : 'tab:brown',
                   7 : 'g',
                  }


y_pred_color = [LABEL_COLOR_MAP[l] for l in y_pred]

plt.scatter(projected[:, 0], projected[:, 1],
            c=y_pred_color, edgecolor='none', alpha=0.8,
            cmap=plt.cm.get_cmap('hsv', 10))
plt.xlabel('Linear Combination 1')
plt.ylabel('Linear Combination 2')
plt.title('Agglomerative Maximum Link')
plt.colorbar();
```



In [51]:
```
print(adjusted_rand_score(y_pred, Glio_y))
print(rand_index(y_pred, Glio_y))
print(normalized_mutual_info_score(y_pred, Glio_y))
```

```
0.18970666683420748
0.7493028929940746
0.31654524916032284
```

In [16]:
```
plt.scatter(projected[:, 0], projected[:, 1],
            c=b, edgecolor='none', alpha=0.8,
            cmap=plt.cm.get_cmap('hsv', 10))
plt.xlabel('Linear Combination 1')
plt.ylabel('Linear Combination 2')
plt.title('Complete 1-PCC')
plt.colorbar();
```

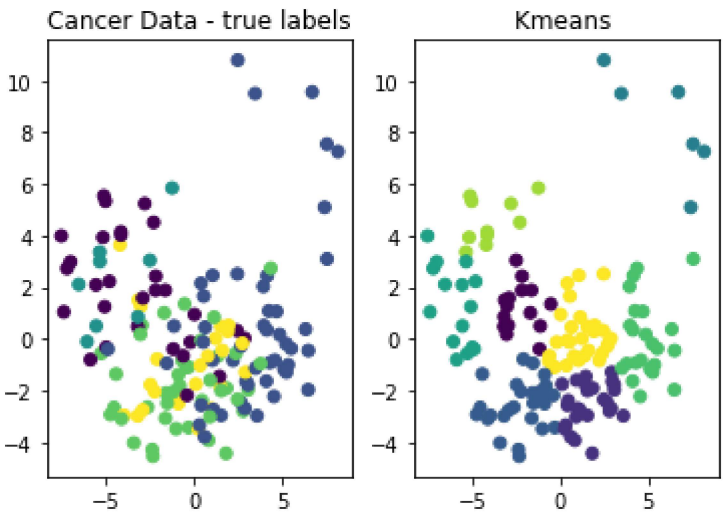


In [38]:
```
print(adjusted_rand_score(b, Glio_y))
print(normalized_mutual_info_score(b, Glio_y))
print(adjusted_mutual_info_score(b, Glio_y))
```

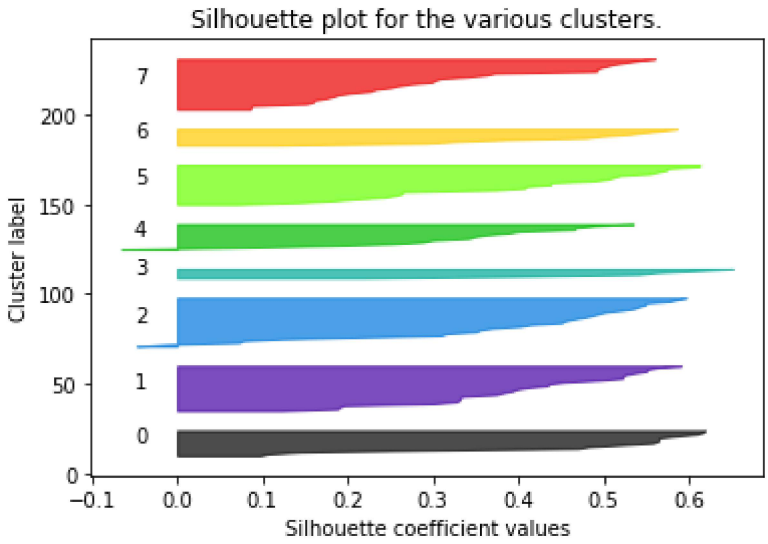
```
0.008240111071814426
0.6229522481700941
(array([1, 4, 5, 3, 5, 4, 1, 2, 2, 3, 3, 5, 5, 3, 4, 2, 3, 5, 3, 3, 5, 2,
       2, 3, 3, 3, 3, 4, 5, 5, 5, 5, 2, 3, 3, 5, 5, 5, 4, 4, 3, 3, 5, 5,
       5, 5, 1, 4, 5, 3, 2, 3, 3, 3, 5, 4, 4, 3, 5, 5, 5, 5, 2, 4, 3, 5,
       4, 5, 2, 5, 5, 3, 3, 5, 3, 5, 2, 1, 3, 3, 4, 2, 3, 5, 4, 2, 3, 4,
       5, 5, 5, 4, 3, 4, 4, 4, 5, 4, 4, 4, 5, 5, 3, 2, 3, 5, 5, 5, 5, 4,
       5, 2, 5, 5, 5, 3, 3, 3, 3, 5, 5, 4, 5, 5, 3, 3, 4, 1, 5, 5, 2, 2,
       3, 5, 5, 4, 3, 3, 2, 4, 3, 3, 3, 4, 5, 5, 3, 3, 5, 5, 5, 3]), array([0
1, 2, 3, 1, 3, 2, 1, 4, 4, 1, 0, 3, 3, 1, 0, 4, 1, 3, 1, 1, 3,
       4, 0, 1, 1, 0, 4, 0, 3, 3, 0, 3, 4, 1, 1, 4, 3, 1, 0, 0, 1, 1, 3,
       3, 3, 4, 1, 2, 3, 1, 4, 1, 4, 3, 3, 0, 2, 1, 3, 1, 4, 4, 0, 0, 1,
       4, 0, 0, 3, 3, 4, 3, 1, 3, 1, 3, 4, 1, 1, 1, 0, 4, 4, 3, 0, 0, 1,
       0, 1, 3, 4, 0, 3, 2, 0, 2, 3, 2, 0, 0, 1, 1, 0, 4, 1, 4, 3, 3, 1,
       2, 3, 1, 4, 3, 3, 1, 3, 1, 1, 3, 4, 1, 1, 1, 1, 1, 1, 1, 3, 3, 4,
       4, 1, 1, 1, 0, 3, 1, 0, 0, 3, 1, 1, 0, 0, 3, 4, 3, 4, 3, 0]))
```

In [59]:
```
kmeans= KMeans(n_init=151, n_clusters = 5, random_state=random_state)
y_pred = kmeans.fit_predict(projected)
print(y_pred)
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(projected[:, 0], projected[:, 1],  c=Glio_y) # true clusters
plt.title('Cancer Data - true labels')
plt.subplot(1,2,2)
plt.scatter(projected[:, 0], projected[:, 1], c=y_pred) # KMeans clusters
plt.title('Kmeans ')
plt.show()
```

```
[7 3 6 4 5 1 4 3 1 0 5 7 2 2 1 6 6 7 1 5 5 7 2 6 3 5 1 7 6 2 0 7 1 2 5 5 1
 2 2 4 0 1 5 7 1 7 2 3 6 1 5 0 5 7 1 2 6 4 1 0 7 2 2 0 4 5 2 4 0 7 2 1 7 5
 7 5 2 7 3 1 1 4 7 1 2 4 4 1 6 7 2 2 4 1 0 4 4 1 0 0 6 1 1 7 1 5 2 0 2 1 4
 0 5 2 7 2 7 1 7 5 2 7 4 7 7 5 7 7 3 2 2 7 7 5 0 1 0 5 5 0 6 1 5 5 4 7 2 7
 5 2 2 2]
```



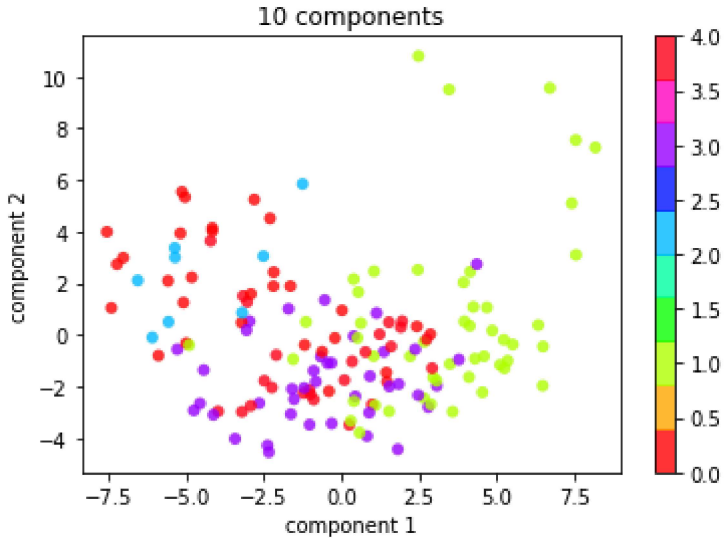
In [60]:
```
silhouette(projected,y_pred)
```



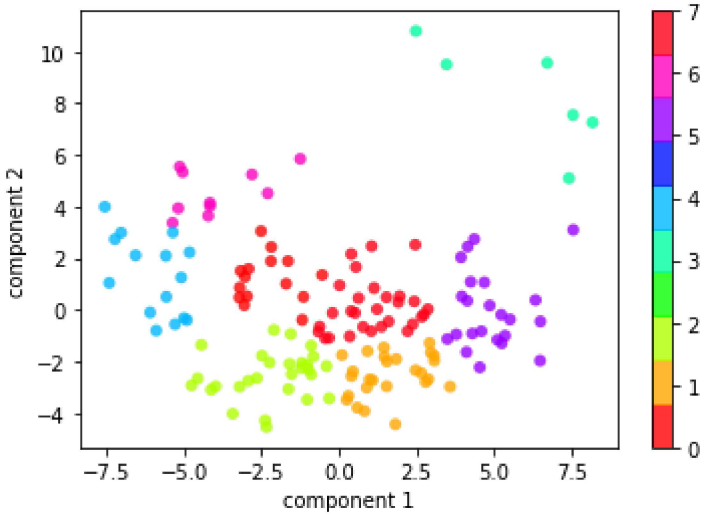
In [61]:
```
print(rand_index(y_pred, Glio_y))
```

```
0.7384977344022308
```

In [80]:
```
pca = PCA(10)  # project from 64 to 2 dimensions
projected = pca.fit_transform(Glio_X.data)

plt.scatter(projected[:, 0], projected[:, 1],
            c=Glio_y, edgecolor='none', alpha=0.8,
            cmap=plt.cm.get_cmap('hsv', 10))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.title('10 components')
plt.colorbar();
```



In [63]:
```
plt.scatter(projected[:, 0], projected[:, 1],
            c=y_pred, edgecolor='none', alpha=0.8,
            cmap=plt.cm.get_cmap('hsv', 10))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```

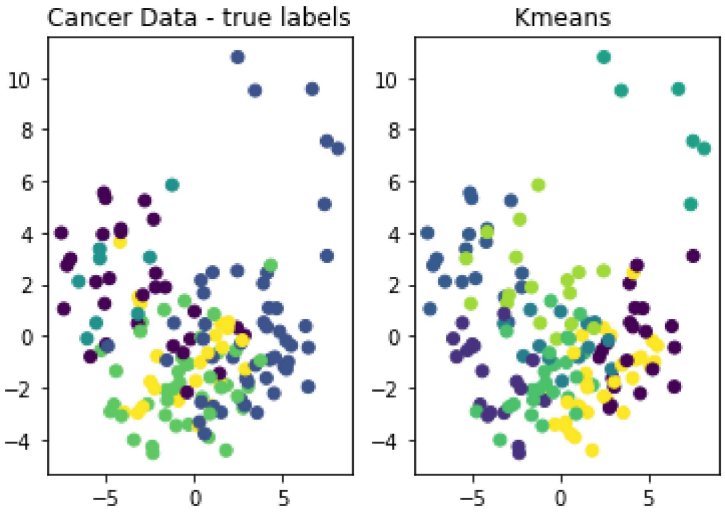


In [64]:
```
print(rand_index(b, Glio_y))
print(normalized_mutual_info_score(b, Glio_y))
```

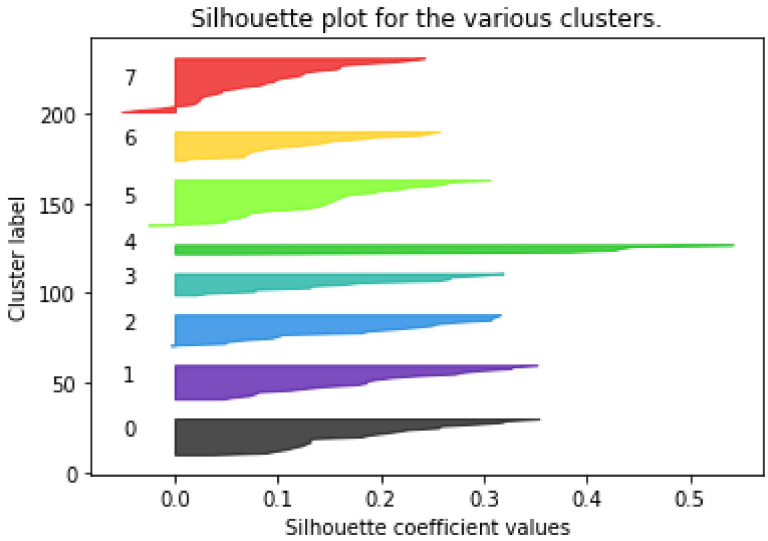
```
0.6229522481700941
```

In [65]:
```
kmeans= KMeans(n_init=151,  random_state=random_state)
y_pred = kmeans.fit_predict(projected)
print(y_pred)
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(projected[:, 0], projected[:, 1],  c=Glio_y) # true clusters
plt.title('Cancer Data - true labels')
plt.subplot(1,2,2)
plt.scatter(projected[:, 0], projected[:, 1], c=y_pred) # KMeans clusters
plt.title('Kmeans ')
plt.show()
```

```
[6 4 6 1 7 5 2 4 7 2 7 7 1 7 7 2 2 0 7 0 0 5 3 2 4 7 7 7 2 5 3 3 5 1 7 0 7
 1 1 2 2 7 0 5 5 5 1 4 2 0 0 6 0 3 0 5 6 1 0 2 6 1 1 2 2 0 7 2 3 3 5 3 0 0
 5 0 5 3 4 0 7 6 3 7 5 1 1 7 6 6 5 1 2 7 1 2 1 7 6 6 2 7 7 7 3 7 1 5 5 7 6
 5 0 5 5 1 6 7 0 0 5 5 1 3 5 7 6 6 4 1 5 7 3 7 6 7 6 0 7 3 2 5 7 0 2 5 1 6
 0 1 5 5]
```



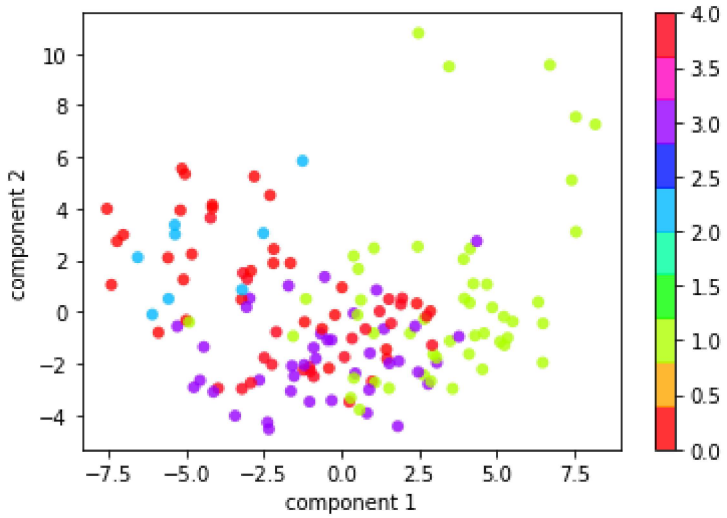
In [66]:
```
silhouette(projected,y_pred)
```



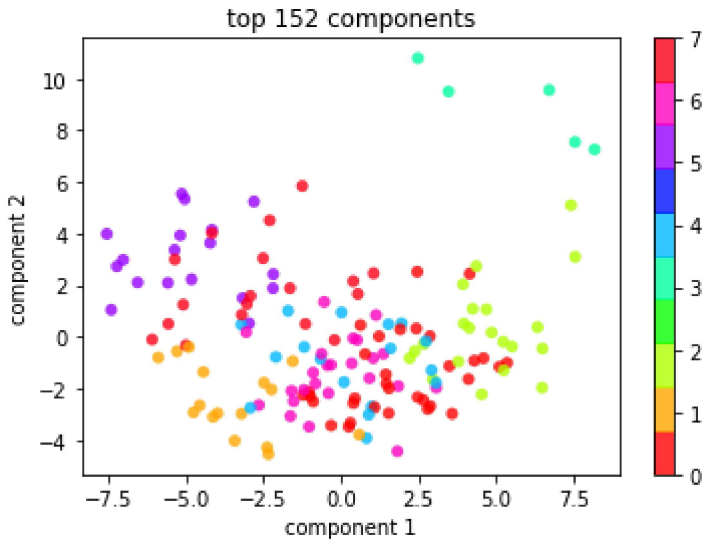
In [67]:
```
print(rand_index(y_pred, Glio_y))
```

```
0.7486057859881492
```

In [91]:
```
pca = PCA(152)  # project from 64 to 2 dimensions
projected = pca.fit_transform(Glio_X.data)

plt.scatter(projected[:, 0], projected[:, 1],
            c=Glio_y, edgecolor='none', alpha=0.8,
            cmap=plt.cm.get_cmap('hsv', 10))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```



In [92]:
```
plt.scatter(projected[:, 0], projected[:, 1],
            c=y_pred, edgecolor='none', alpha=0.8,
            cmap=plt.cm.get_cmap('hsv', 10))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.title('top 152 components')
plt.colorbar();
```

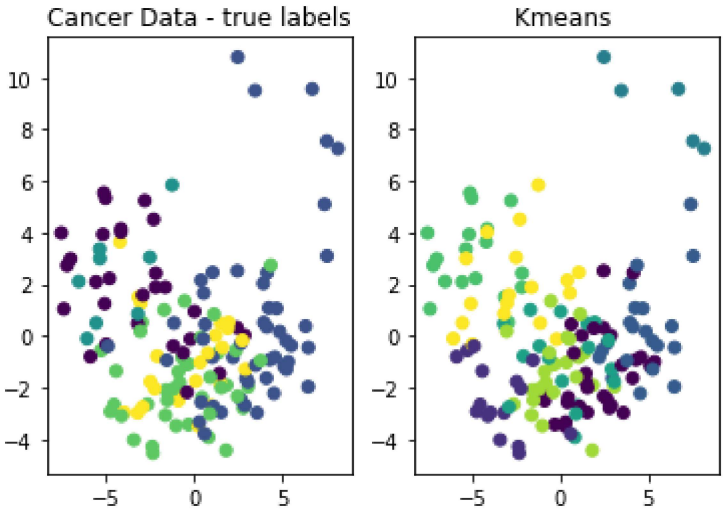


In [93]:
```
print(rand_index(b, Glio_y))
```

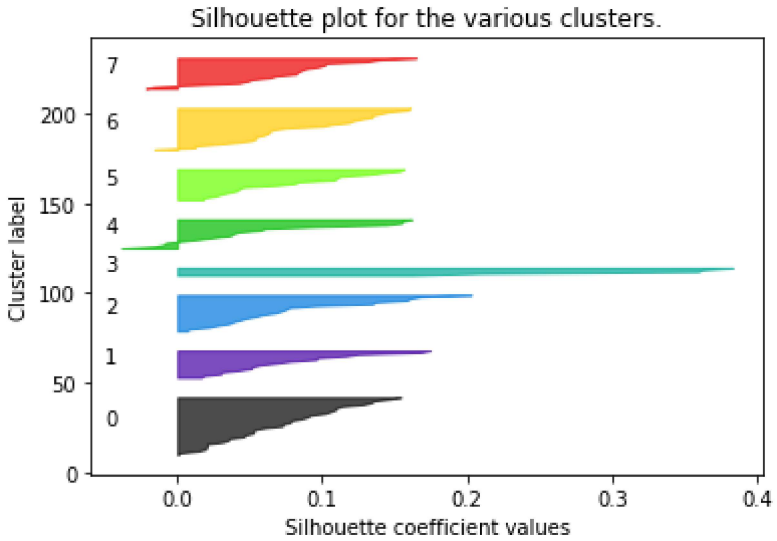
```
0.6229522481700941
```

In [72]:
```
kmeans= KMeans(n_init=151,  random_state=random_state)
y_pred = kmeans.fit_predict(projected)
print(y_pred)
fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(projected[:, 0], projected[:, 1],  c=Glio_y) # true clusters
plt.title('Cancer Data - true labels')
plt.subplot(1,2,2)
plt.scatter(projected[:, 0], projected[:, 1], c=y_pred) # KMeans clusters
plt.title('Kmeans ')
plt.show()
```

```
[7 3 7 1 2 6 5 3 0 5 0 0 1 0 0 5 5 2 6 2 2 6 4 5 2 0 0 0 5 6 4 4 6 1 0 2 0
 1 1 5 5 0 2 6 0 6 0 3 5 6 2 7 2 4 0 6 7 7 2 5 0 1 0 5 5 2 0 5 4 4 6 4 2 2
 6 2 1 4 3 0 4 7 4 4 6 7 1 0 7 7 6 0 5 0 7 5 7 0 7 7 5 0 0 0 4 2 1 6 6 0 7
 6 2 4 6 1 0 4 2 2 6 6 1 6 6 0 7 7 3 1 1 0 4 0 7 1 7 2 0 4 5 4 0 2 5 6 1 0
 2 1 6 6]
```



In [73]:
```
silhouette(projected,y_pred)
```



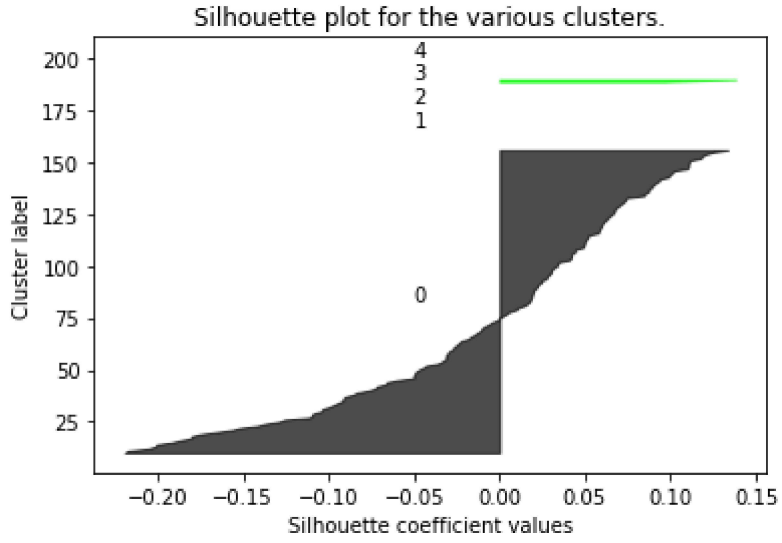
In [74]:
```
print(rand_index(y_pred, Glio_y))
```

```
0.7493028929940746
```

In [14]:
```
print("Glio:")
n_clusters = 5;
spectral = SpectralClustering(n_clusters=n_clusters, random_state=5)
y_pred = spectral.fit_predict(Glio_X)
```

```
Glio:
```

In [15]:
```
silhouette(Glio_X,y_pred)
```



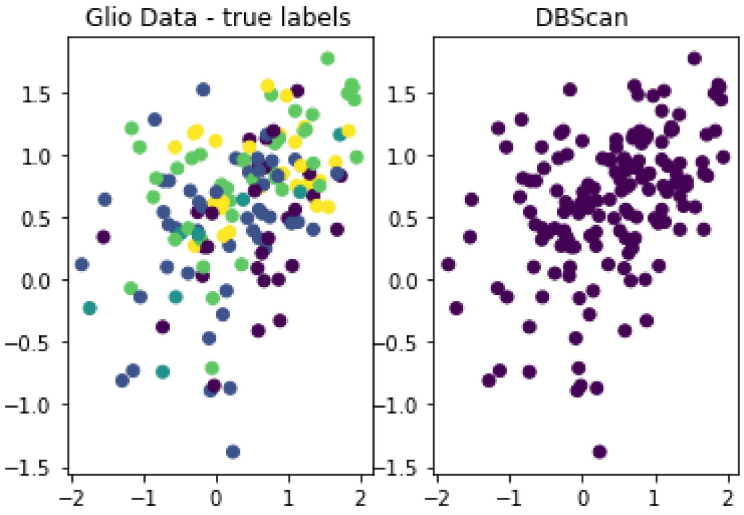
In [16]:
```
print(rand_index(y_pred, Glio_y))
```

```
0.268299058905542
```

In [28]:
```
n_clusters = 5
single_linkage = AgglomerativeClustering(linkage="single", n_clusters=n_clust
y_pred = single_linkage.fit_predict(Glio_X)

print(rand_index(y_pred, Glio_y))
```

```
0.26202509585221334
```

In [19]:
```
dbscan = DBSCAN(eps=10000000, min_samples=10)
y_pred = dbscan.fit_predict(Glio_X)

fig, ax = plt.subplots()
plt.subplot(1,2,1)
plt.scatter(Glio_X[:, 0], Glio_X[:, 1], c=Glio_y) # true clusters
plt.title('Glio Data - true labels')
plt.subplot(1,2,2)
plt.scatter(Glio_X[:, 0], Glio_X[:, 1], c=y_pred) # KMeans clusters
plt.title('DBScan ')
plt.show()

print(rand_index(y_pred, Glio_y))
```




```
0.23658069013593586
```

In [ ]: