In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
from numpy import asarray
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# Import data to dataframe
df = pd.read_csv(r'C:/Users/jeric/OneDrive/Documents/classFiles/DSC630_AD/data/als_data.csv', index_col=False)

record_count = len(df)
print("Record Count: "+str(record_count))

#View Data
df.head()
```

```
Record Count: 2223
```

Out[1]:

	ID	Age_mean	Albumin_max	Albumin_median	Albumin_min	Albumin_range	ALSFRS_slope	ALSFRS_Total_max	ALSFRS_Total_median	ALSFI
0	1	65	57.0	40.5	38.0	0.066202	-0.965608	30	28.0	
1	2	48	45.0	41.0	39.0	0.010453	-0.921717	37	33.0	
2	3	38	50.0	47.0	45.0	0.008929	-0.914787	24	14.0	
3	4	63	47.0	44.0	41.0	0.012111	-0.598361	30	29.0	
4	5	63	47.0	45.5	42.0	0.008292	-0.444039	32	27.5	

Step 1: Remove any data that is not relevant to the patient’s ALS condition.

In [2]:
```
data = df.drop(['ID', 'SubjectID' ], axis=1)
data.head()
```

Out[2]:

	Age_mean	Albumin_max	Albumin_median	Albumin_min	Albumin_range	ALSFRS_slope	ALSFRS_Total_max	ALSFRS_Total_median	ALSFRS_T
0	65	57.0	40.5	38.0	0.066202	-0.965608	30	28.0	
1	48	45.0	41.0	39.0	0.010453	-0.921717	37	33.0	
2	38	50.0	47.0	45.0	0.008929	-0.914787	24	14.0	
3	63	47.0	44.0	41.0	0.012111	-0.598361	30	29.0	
4	63	47.0	45.5	42.0	0.008292	-0.444039	32	27.5	

Step 2: Apply a standard scalar to the data.

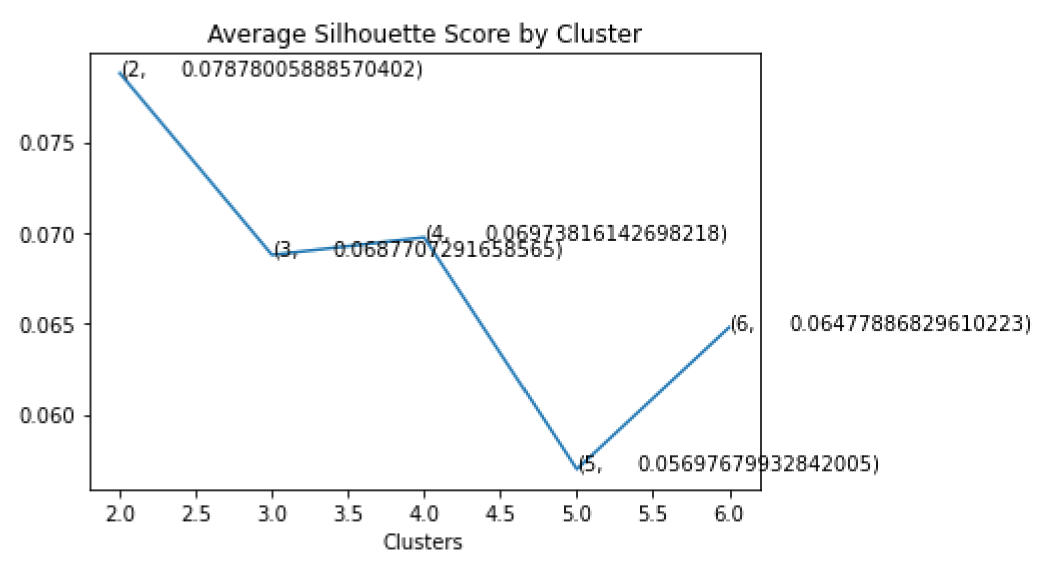
In [3]:
```
cols = data.columns
sc = StandardScaler()
scaler = StandardScaler()
scaled = pd.DataFrame(sc.fit_transform(data), columns=cols)
scaled.head()
```

Out[3]:

	Age_mean	Albumin_max	Albumin_median	Albumin_min	Albumin_range	ALSFRS_slope	ALSFRS_Total_max	ALSFRS_Total_median	ALSFRS_T
0	0.917137	3.089417	-1.300781	-0.866550	5.480929	-0.381450	-0.318520	0.134960	
1	-0.574879	-0.622016	-1.112401	-0.553303	-0.347725	-0.310907	0.998995	0.888863	
2	-1.452535	0.924415	1.148162	1.326179	-0.507103	-0.299769	-1.447819	-1.975969	-
3	0.741606	-0.003443	0.017880	0.073191	-0.174361	0.208801	-0.318520	0.285741	
4	0.741606	-0.003443	0.583021	0.386438	-0.573670	0.456831	0.057913	0.059570	

Step 3: Create a plot of the cluster silhouette score versus the number of clusters in a K-means cluster.

In [59]:
```
from sklearn.cluster import KMeans

X = scaled
range_n_clusters = [2, 3, 4, 5, 6]
silhouette_avg_n_clusters = []

for n_clusters in range_n_clusters:
    clusterer = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = clusterer.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    silhouette_avg_n_clusters.append(silhouette_avg)


y = silhouette_avg_n_clusters
x = range(2,7)

fig = plt.figure()
ax = fig.add_subplot(111)

plt.plot(x, y)
plt.title("Average Silhouette Score by Cluster")
plt.xlabel("Clusters")

for i,j in zip(x,y):
    ax.annotate('%s)' %j, xy=(i,j), xytext=(30,0), textcoords='offset points')
    ax.annotate('(%s,' %i, xy=(i,j))

plt.show()
```



Step 4: Use the plot created in (3) to choose an optimal number of clusters for K-means. Justify your choice.

I will be choosing a cluster count of 2, as this had the highest silhouette score, meaning the similarity of the data points within the two clusters are more similar than other the other clusters where k > 2.

Step 5: Fit a K-means model to the data with the optimal number of clusters chosen in part (4).

In [61]:
```
kmeans = KMeans(
    init="random",
    n_clusters=2,
    n_init=10,
    max_iter=300,
    random_state=42
    )

clustering_kmeans = kmeans.fit(scaled)
print(clustering_kmeans)
```

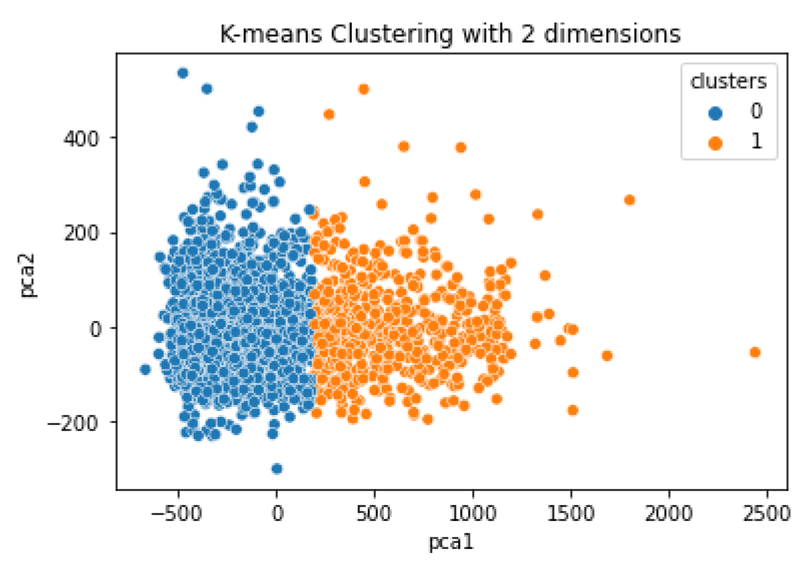
```
KMeans(init='random', n_clusters=2, random_state=42)
```

Step 6: Fit a PCA transformation with two features to the scaled data.

In [66]:
```
pca_num_components = 2

scaled['clusters'] = clustering_kmeans.fit_predict(data)
reduced_data = PCA(n_components=pca_num_components).fit_transform(data)
results = pd.DataFrame(reduced_data,columns=['pca1','pca2'])
```

Step 7: Make a scatterplot of the PCA transformed data coloring each point by its cluster value.

In [67]:
```
sns.scatterplot(x="pca1", y="pca2", hue=scaled['clusters'], data=results)
plt.title('K-means Clustering with 2 dimensions')

plt.show()
```



Step 8: Summarize your results and make a conclusion.

When clustering the data, our highest silhouette score is 0.078, which is not very close to 1. When we use PCA and plot the first principal component group and second the second, we can see two distinct clusters. This tells us that the data can be clustered. The next step would be to determine the features of the clusters to see what we can find out about the distinct clusters.

In [ ]: