Homework 7

YIMIN LI

K-means clustering by hand

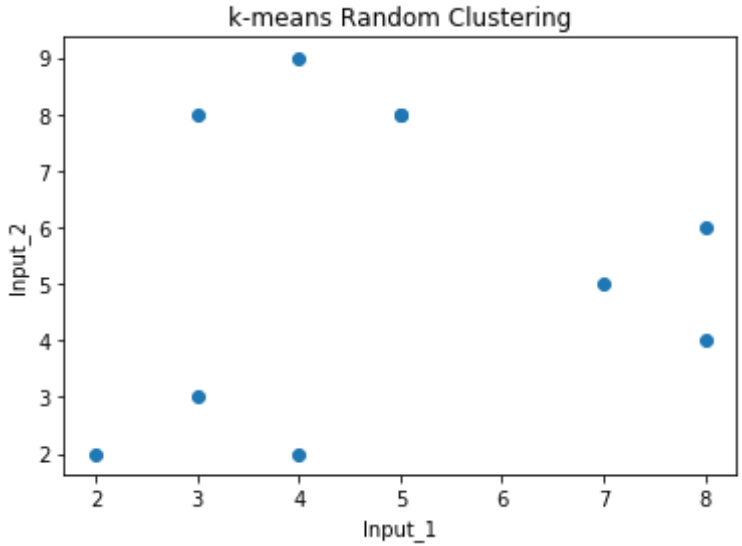
In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

1.

In [138]:
```
input1 = np.array([5,8,7,8,3,4,2,3,4,5])
input2 = np.array([8,6,5,4,3,2,2,8,9,8])
k_label = np.random.choice(3, 10, replace=True)
df = pd.DataFrame({'input1':input1, 'input2':input2, 'klabel':k_label})
```

In [139]:
```
plt.scatter(df['input1'], df['input2'])
plt.xlabel('Input_1')
plt.ylabel('Input_2')
plt.title("k-means Random Clustering")
```

Out[139]:
```
Text(0.5, 1.0, 'k-means Random Clustering')
```



2.

In [140]:
```
df
```

Out[140]:

	input1	input2	klabel
0	5	8	2
1	8	6	1
2	7	5	0
3	8	4	1
4	3	3	2
5	4	2	1
6	2	2	2
7	3	8	2
8	4	9	0
9	5	8	2

In [141]:
```
def fit_k(k, df):
    temp_df = df.copy()
    for i in range(df.shape[0]):
        centroids = {}
        for c in range(k):
            centroidx = temp_df[temp_df['klabel'] == c]['input1'].mean
()
            centroidy = temp_df[temp_df['klabel'] == c]['input2'].mean
()
            centroids[c] = (centroidx, centroidy)

        labels = []
        for index, row in temp_df.iterrows():
            min_distance = 1000
            for key, value in centroids.items():
                distance = ((row['input1'] - value[0]) ** 2 + (row['inp
ut2'] - value[1])**2)**0.5
                if distance < min_distance:
                    min_distance = distance
                    min_key = key
            labels.append(min_key)
        temp_df['klabel'] = labels
    return temp_df
```

In [142]:
```
df_3 = fit_k(3, df)
df_3
```

Out[142]:

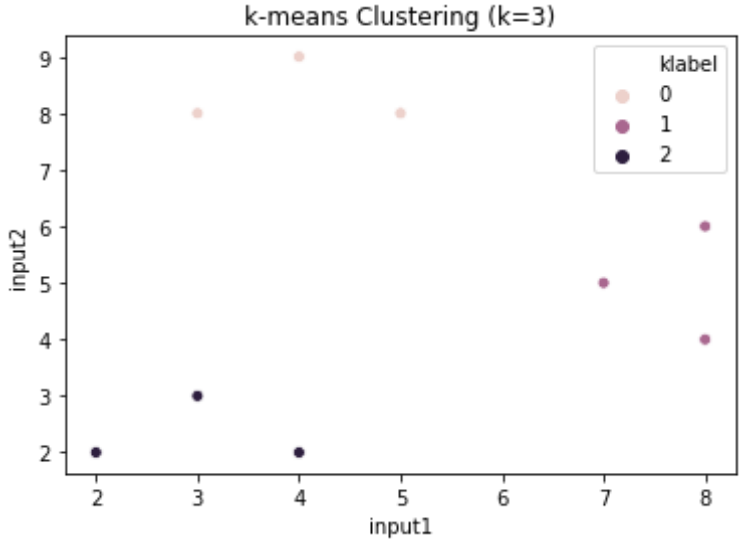
	input1	input2	klabel
0	5	8	0
1	8	6	1
2	7	5	1
3	8	4	1
4	3	3	2

	input1	input2	klabel
5	4	2	2
6	2	2	2
7	3	8	0
8	4	9	0
9	5	8	0

3.

In [145]:
```
sns.scatterplot(x='input1', y='input2', data=df_3, hue='klabel')
plt.title('k-means Clustering (k=3)')
```

Out[145]: Text(0.5, 1.0, 'k-means Clustering (k=3)')

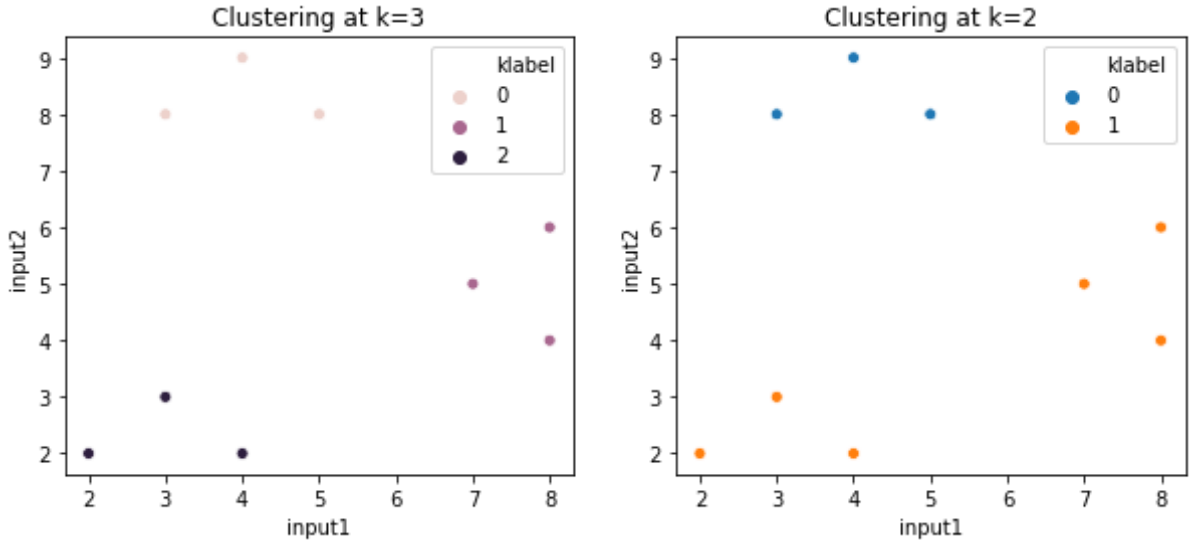


4.

In [148]:
```
df_2 = fit_k(2, df)
```

```
plt.figure(figsize=(10, 4))
ax = plt.subplot(121)
ax.set_title('Clustering at k=3')
sns.scatterplot(x='input1', y='input2', data=df_3, hue='klabel')
ax = plt.subplot(122)
ax.set_title('Clustering at k=2')
sns.scatterplot(x='input1', y='input2', data=df_2, hue='klabel')
```

Out[148]: <matplotlib.axes._subplots.AxesSubplot at 0x179a8c9e208>



5.

k-means with k=3 seems fit better compared with k-means with k=2. When k=2, the points are not that close. When k=3, the within-cluster distance is minimized and the distance between clusters is maximized. Hence, k=3 seems better than k=2.

Dimension reduction

6.

In [175]:
```
df_wiki = pd.read_csv("https://raw.githubusercontent.com/macss-model20/problem-set-7/master/data/wiki.csv")
```

In [176]:
```
df_columns = df_wiki.columns
df_wiki = StandardScaler().fit_transform(df_wiki)
pca = PCA(n_components=2)
pca_trans = pca.fit_transform(df_wiki)
df_pca = pd.DataFrame(data=pca_trans, columns=['PC1', 'PC2'])
df_pca
```

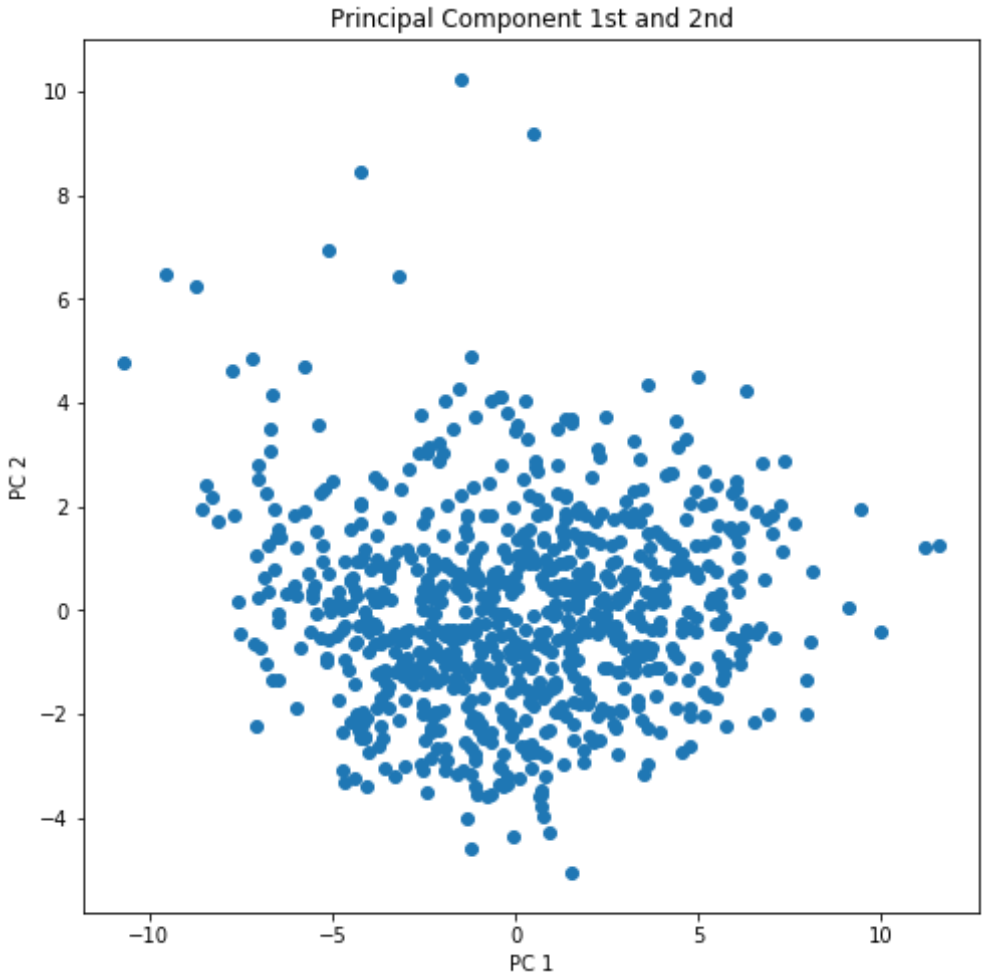
Out[176]:

	PC1	PC2
0	-0.150216	-1.982812
1	-3.314020	-0.792386
2	-4.682483	-0.312602
3	1.774200	1.986270
4	7.254695	2.013940
...	...	...
795	0.227143	1.475297
796	4.434784	-0.932011
797	1.449455	-0.170831
798	-2.888282	2.721546
799	-7.000656	2.806011

800 rows × 2 columns

In [177]:
```
plt.figure(figsize=(8, 8))
plt.title('Principal Component 1st and 2nd')
plt.scatter(df_pca['PC1'], df_pca['PC2'])
plt.xlabel("PC 1")
```

```
plt.ylabel("PC 2")
plt.show()
```



In [178]:
```
df_pca_2 = pd.DataFrame(data=pca.fit(df_wiki).components_.T, index=df_c
olumns, columns=['PC1', 'PC2'])
df_pca_2
```

Out[178]:

	PC1	PC2

	PC1	PC2
age	-0.021805	0.088377
gender	-0.035086	-0.149465
phd	-0.030501	0.030441
yearsexp	-0.034190	0.062362
userwiki	0.081363	0.134398
pu1	0.192827	0.008274
pu2	0.190588	0.017669
pu3	0.210863	0.028777
peu1	0.061228	-0.271741
peu2	0.113719	-0.222371
peu3	0.100219	-0.068466
enj1	0.145666	-0.151014
enj2	0.131110	-0.227605
qu1	0.178057	-0.038124
qu2	0.163778	-0.066422
qu3	0.157956	-0.033477
qu4	-0.060797	-0.103440
qu5	0.183365	0.010911
vis1	0.171153	-0.025208
vis2	0.114559	-0.056222
vis3	0.175351	0.197632
im1	0.160432	0.111112
im2	0.077810	-0.059768
im3	0.160803	0.044012
sa1	0.121658	-0.229922

	PC1	PC2
sa2	0.117590	-0.226762
sa3	0.120376	-0.242323
use1	0.181477	0.197830
use2	0.147852	0.218632
use3	0.218809	0.155153
use4	0.214558	0.160865
use5	0.206539	0.029824
pf1	0.102338	0.114368
pf2	0.103448	0.018598
pf3	0.109632	0.094166
jr1	0.080867	-0.136971
jr2	0.062216	-0.106299
bi1	0.226193	0.056371
bi2	0.230924	0.083429
inc1	0.104667	-0.245438
inc2	0.095802	-0.202023
inc3	0.081402	-0.220984
inc4	0.089707	-0.202021
exp1	0.208592	0.070546
exp2	0.195043	-0.029555
exp3	0.144023	-0.126411
exp4	0.099873	0.228499
exp5	0.110628	0.076094
domain_Sciences	0.021982	-0.014540
domain_Health.Sciences	-0.017158	-0.015467

	PC1	PC2
domain_Engineering_Architecture	0.051309	0.171477
domain_Law_Politics	-0.094775	-0.014883
uoc_position_Associate	0.010922	0.013143
uoc_position_Assistant	0.007123	0.002298
uoc_position_Lecturer	-0.018041	-0.023591
uoc_position_Instructor	0.004251	-0.003754
uoc_position_Adjunct	-0.007849	-0.005303

In [179]:
```
df_pca_2.nlargest(5, 'PC1')
```

Out[179]:

	PC1	PC2
bi2	0.230924	0.083429
bi1	0.226193	0.056371
use3	0.218809	0.155153
use4	0.214558	0.160865
pu3	0.210863	0.028777

In [180]:
```
df_pca_2.nlargest(5, 'PC2')
```

Out[180]:

	PC1	PC2
exp4	0.099873	0.228499
use2	0.147852	0.218632
use1	0.181477	0.197830
vis3	0.175351	0.197632
domain_Engineering_Architecture	0.051309	0.171477

As shown in the tables above, the top5 features strongly correlated to PC1 are bi2, bi1, use3, use4 and pu3 while the top5 features strongly correlated to PC2 are exp4, use2, use1, vis3 and domain_engineering_architecture.

7.

In [181]:
```
ls_ratio = list(pca.explained_variance_ratio_)
ls_ratio
print("The first two principal components explained " + str(sum(ls_rati
o) * 100) + "% of the variance")
```

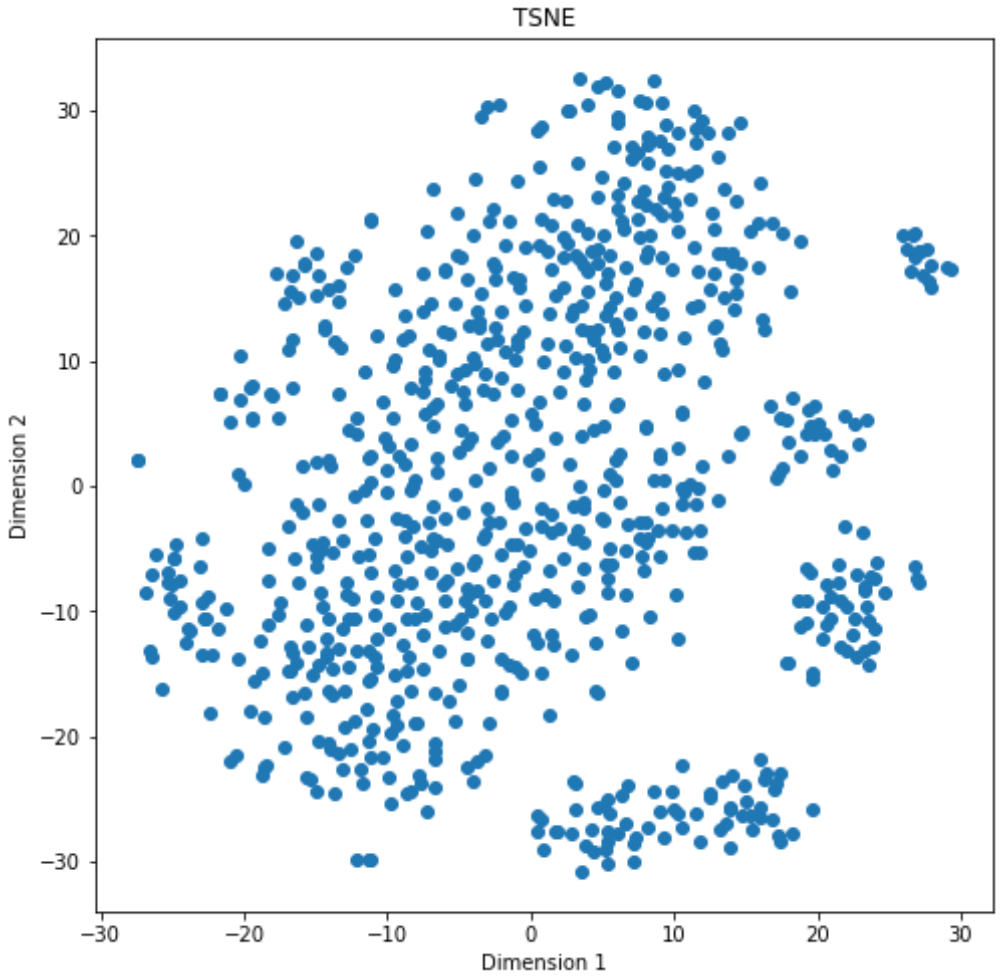
```
The first two principal components explained 29.18310228662659% of the
variance
```

8.

In [182]:
```
tsne = TSNE(n_components=2)
df_tsne = pd.DataFrame(data = tsne.fit_transform(df_wiki), columns = [
'dim1', 'dim2'])
```

In [189]:
```
plt.figure(figsize=(8, 8))
plt.scatter(df_tsne['dim1'], df_tsne['dim2'])
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('TSNE')
```

Out[189]:
```
Text(0.5, 1.0, 'TSNE')
```



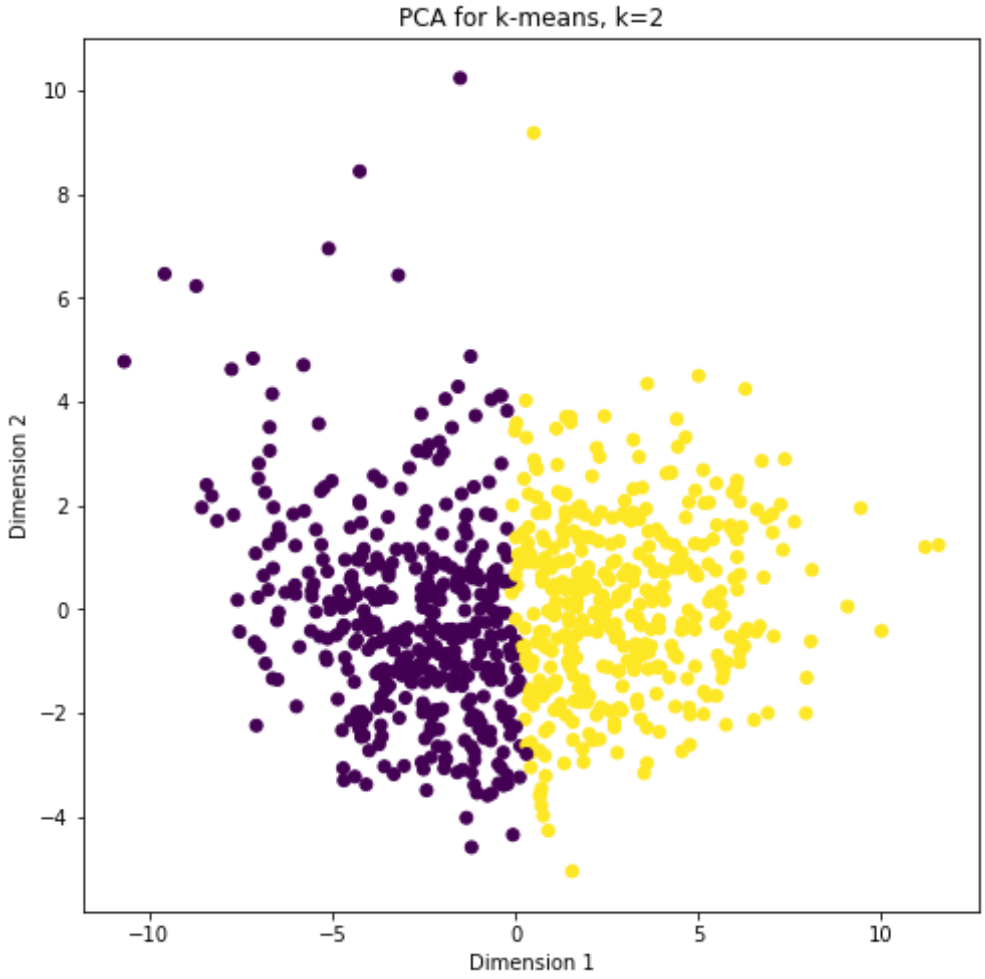
As is shown in the graph, there is a main clustering in the middle while there are small clustering around the main clustering. This may indicate that TSNE might have better clustering prediction that PCA as shown in the prior graph.

Clustering

9.

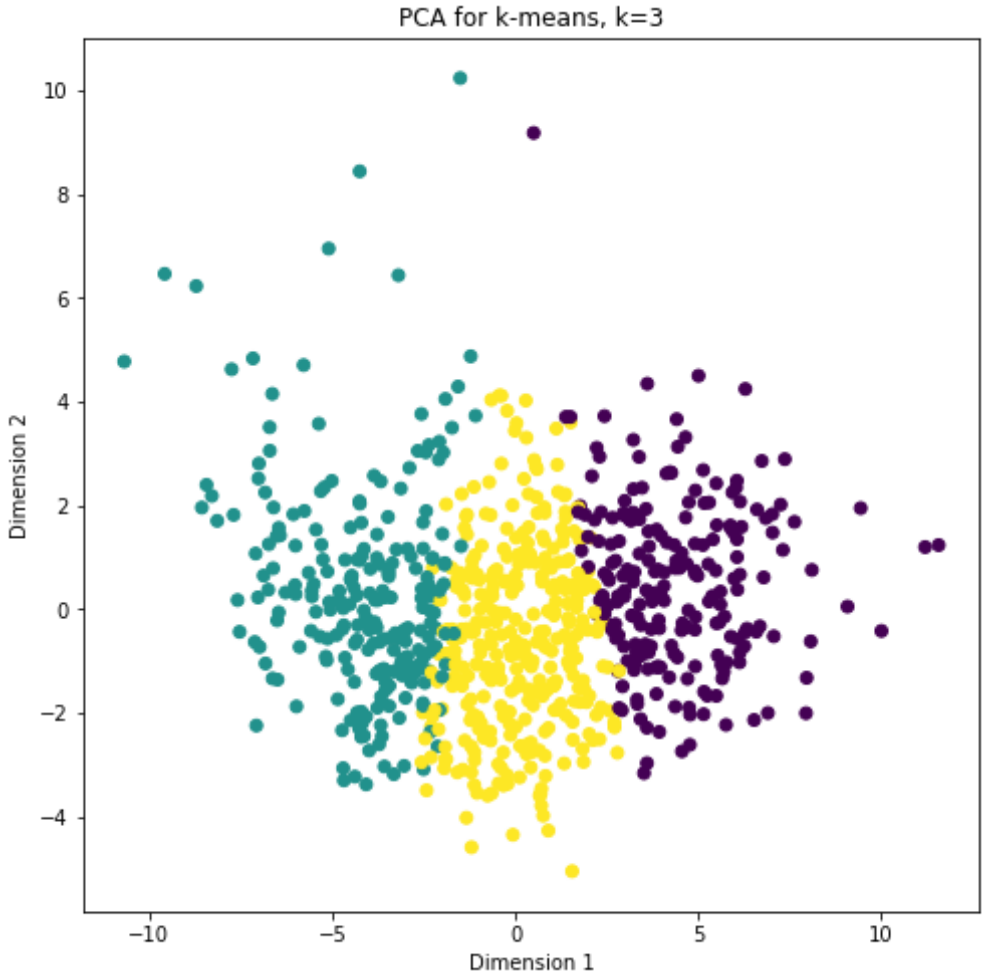
In [201]:
```
kmeans_2 = KMeans(n_clusters=2).fit(df_wiki)
plt.figure(figsize=(8, 8))
plt.scatter(df_pca['PC1'], df_pca['PC2'], c=kmeans_2.labels_)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('PCA for k-means, k=2')
```

Out[201]:
```
Text(0.5, 1.0, 'PCA for k-means, k=2')
```



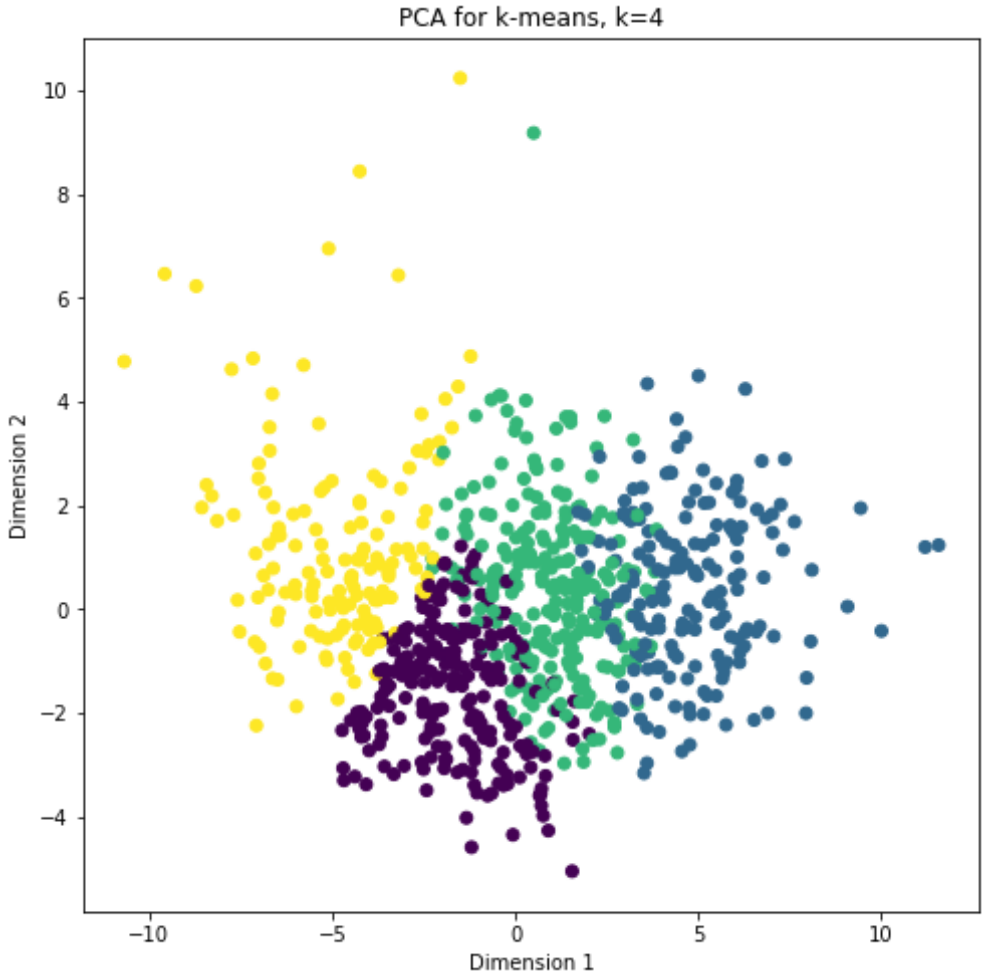
In [202]:
```
kmeans_3 = KMeans(n_clusters=3).fit(df_wiki)
plt.figure(figsize=(8, 8))
plt.scatter(df_pca['PC1'], df_pca['PC2'], c=kmeans_3.labels_)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('PCA for k-means, k=3')
```

Out[202]:
```
Text(0.5, 1.0, 'PCA for k-means, k=3')
```



In [203]:
```
kmeans_4 = KMeans(n_clusters=4).fit(df_wiki)
plt.figure(figsize=(8, 8))
plt.scatter(df_pca['PC1'], df_pca['PC2'], c=kmeans_4.labels_)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('PCA for k-means, k=4')
```

Out[203]:
```
Text(0.5, 1.0, 'PCA for k-means, k=4')
```

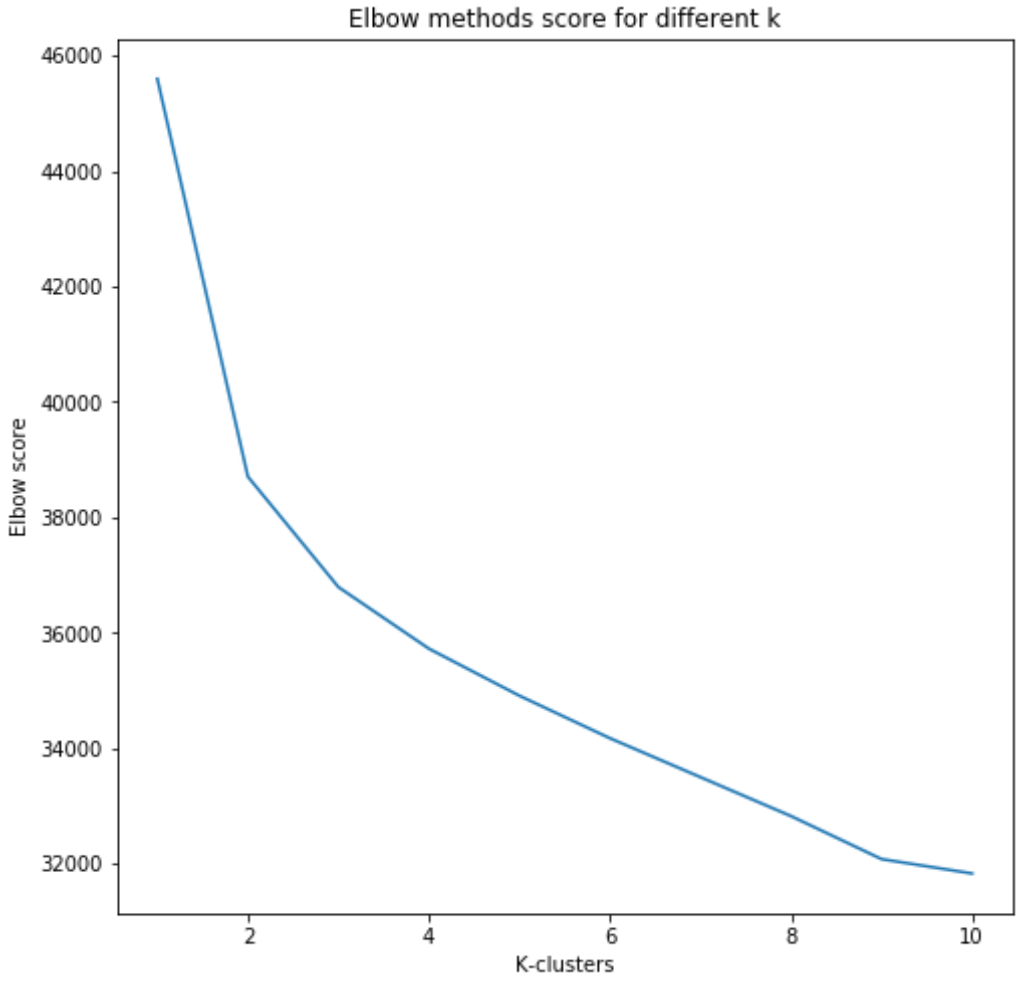


As is shown in the three graphs above, when k=2, we would see the clusters are well explained by the first two principal components. When k increases, the overlap increases. When k=3, the boundary is still clear while k=4, it seems they overlap a lot, meaning there are other factors influencing our data.

10.

In [204]:
```
ls_elbow = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k).fit(df_wiki)
    ls_elbow.append(kmeans.inertia_)
plt.figure(figsize=(8, 8))
plt.plot(np.arange(1,11), ls_elbow)
plt.xlabel('K-clusters')
plt.ylabel('Elbow score')
plt.title('Elbow methods score for different k')
```

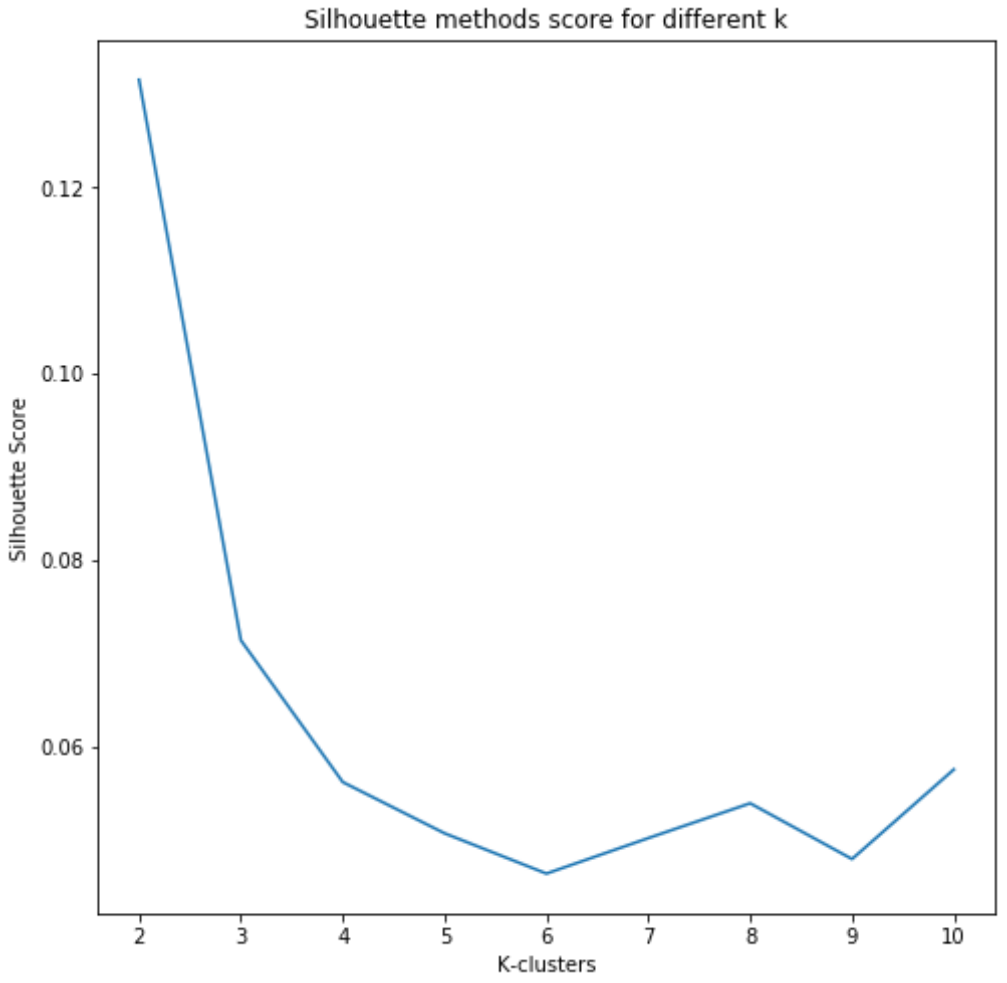
Out[204]:
```
Text(0.5, 1.0, 'Elbow methods score for different k')
```



In [205]:
```
ls_ave = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k)
    ls_ave.append(silhouette_score(df_wiki, kmeans.fit_predict(df_wiki
)))
plt.figure(figsize=(8, 8))
plt.plot(np.arange(2,11), ls_ave)
plt.xlabel('K-clusters')
```

```
plt.ylabel('Silhouette Score')
plt.title('Silhouette methods score for different k')
```

Out[205]: `Text(0.5, 1.0, 'Silhouette methods score for different k')`

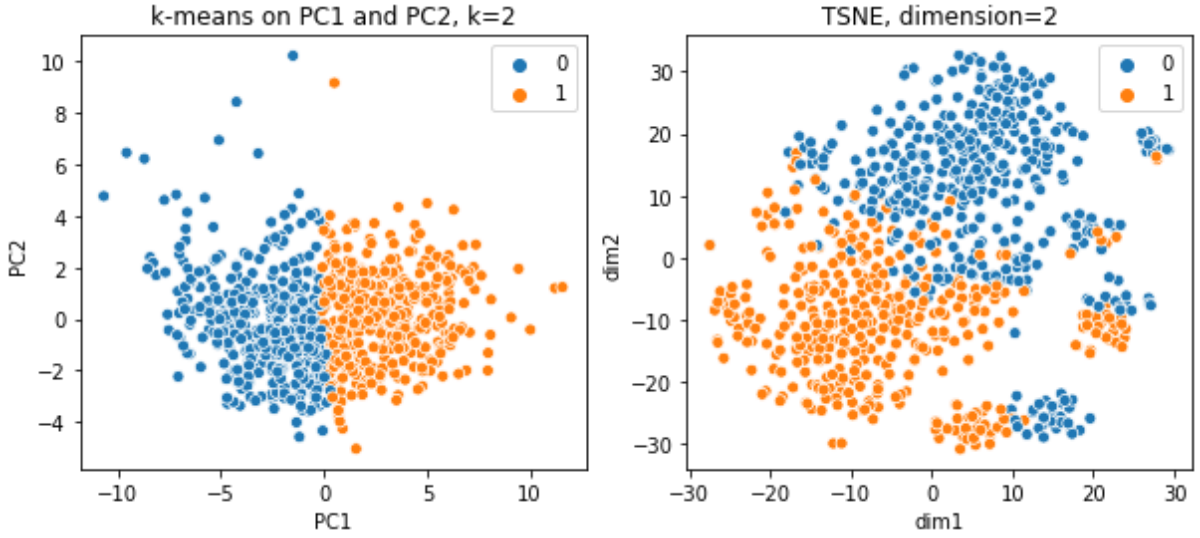


Based on the two graphs above, we would argue that when k=2, the average silhouette is the largest, thus the optimal number would be 2.

11.

In [207]:
```
plt.figure(figsize=(10, 4))
ax = plt.subplot(121)
ax.set_title('k-means on PC1 and PC2, k=2')
sns.scatterplot(x='PC1', y='PC2', data=df_pca, hue=kmeans_2.labels_)
ax = plt.subplot(122)
ax.set_title('TSNE, dimension=2')
sns.scatterplot(x='dim1', y='dim2', data=df_tsne, hue=kmeans_2.labels_)
```

Out[207]: <matplotlib.axes._subplots.AxesSubplot at 0x179adedd848>



These two graphs show that PCA(k=2) has a better explanation and seperation than TSNE, since in TSNE's clustering, some boundaries are overlapping with each other. This may be because that PCA aims to make clear classifications while TSNE maintain the individual observations between neighboring observations.

In [ ]: