HW7

March 15, 2020

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[129]: from sklearn.cluster import KMeans
       from sklearn.decomposition import PCA
       from sklearn.manifold import TSNE
       from sklearn.preprocessing import StandardScaler
       from scipy.spatial.distance import cdist
       from sklearn.metrics import silhouette_samples, silhouette_score
```

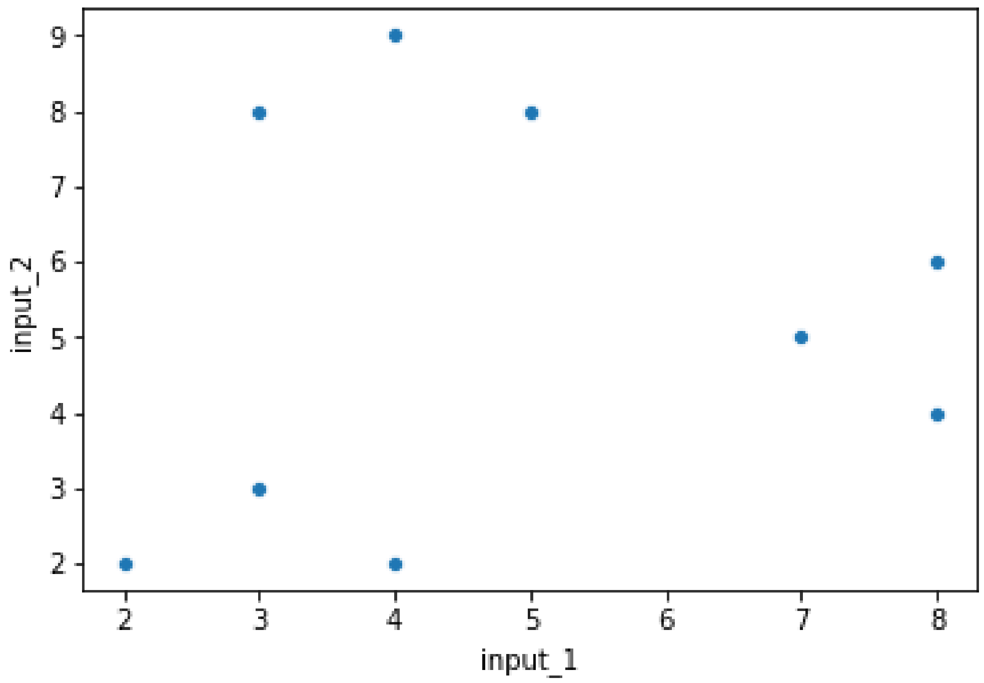
```
[2]: np.random.seed(2019)
```

1 k-Means Clustering "By Hand"

```
[60]: input_1=[5,8,7,8,3,4,2,3,4,5]
      input_2=[8,6,5,4,3,2,2,8,9,8]
      data_df=pd.DataFrame.from_dict({"input_1":input_1,"input_2":input_2})
```

```
[48]: sns.scatterplot(x ='input_1', y='input_2', data=data_df)
```

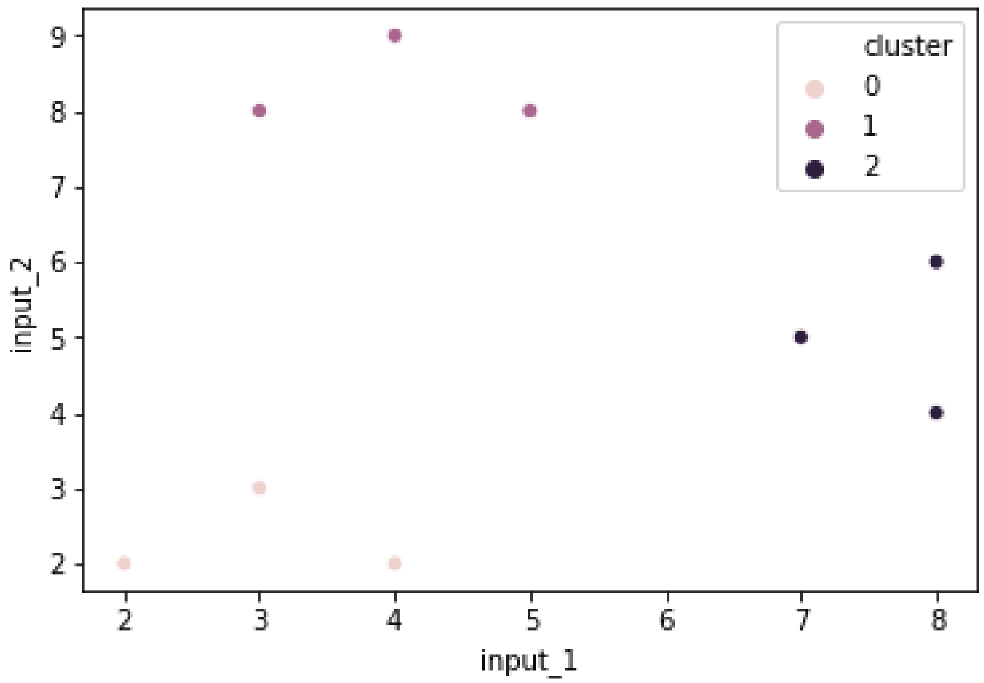
```
[48]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1ebbb790>
```



[49]:
```
def k_means(ncluster, data, max_iteration):

    # initialization
    cluster=np.random.choice(ncluster, size=data.shape[0])
    data['cluster']=cluster

    n_interation=0

    #Compute the cluster centroid and update cluster iteratively
    while(n_interation < max_iteration):

        is_done=True
        # centroid
        centroids = {}
        for k in range(ncluster):
            data_slice=data[data['cluster']==k]
            centroid=(data_slice.mean()['input_1'], data_slice.
 ↪mean()['input_2'])
            centroids[k]=centroid

        # distance
        labels=[]
        for index, row in data.iterrows():
            input_1=row['input_1']
            input_2=row['input_2']
            old_label=row['cluster']
            dist={}
            for k in centroids.keys():
                
↪dist[k]=((input_1-centroids[k][0])**2+(input_2-centroids[k][1])**2)**0.5
            key_min = min(dist.keys(), key=(lambda k: dist[k]))
            labels.append(key_min)
            if key_min != old_label:
                is_done = False

        #if converge
        if is_done == True:
            break

        else:
            # update
            data['cluster']=labels
            n_interation+=1

    return data
```

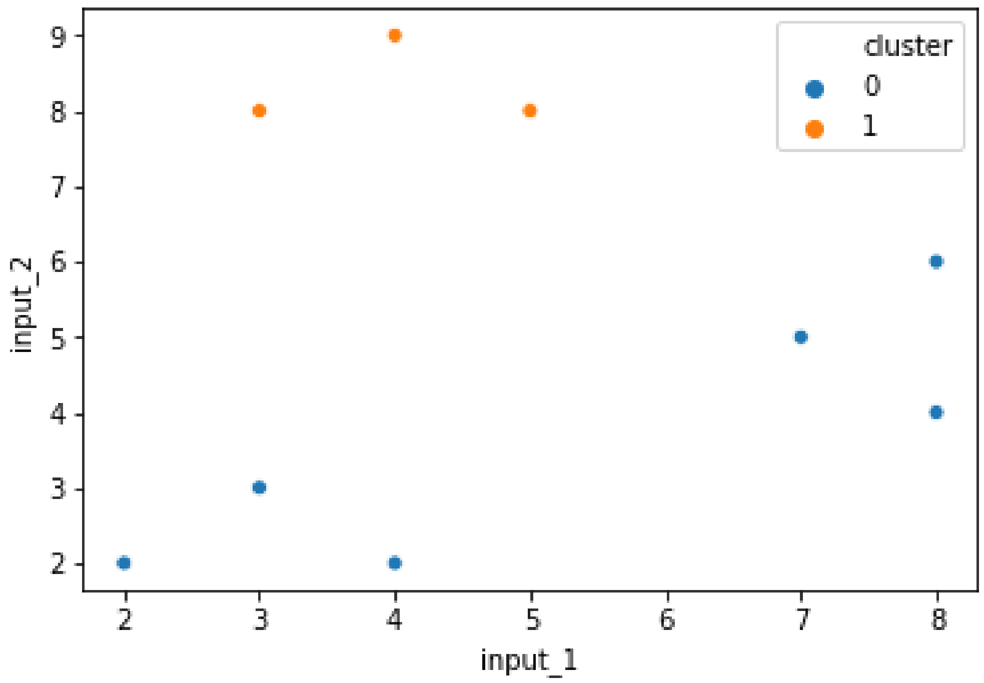
[65]:
```
# 1-3 kmeans and visualization
data_df_3= k_means(3, data_df, 50).copy()
sns.scatterplot(x ='input_1', y='input_2', hue="cluster", data=data_df_3)
```

[65]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a22356610>
```



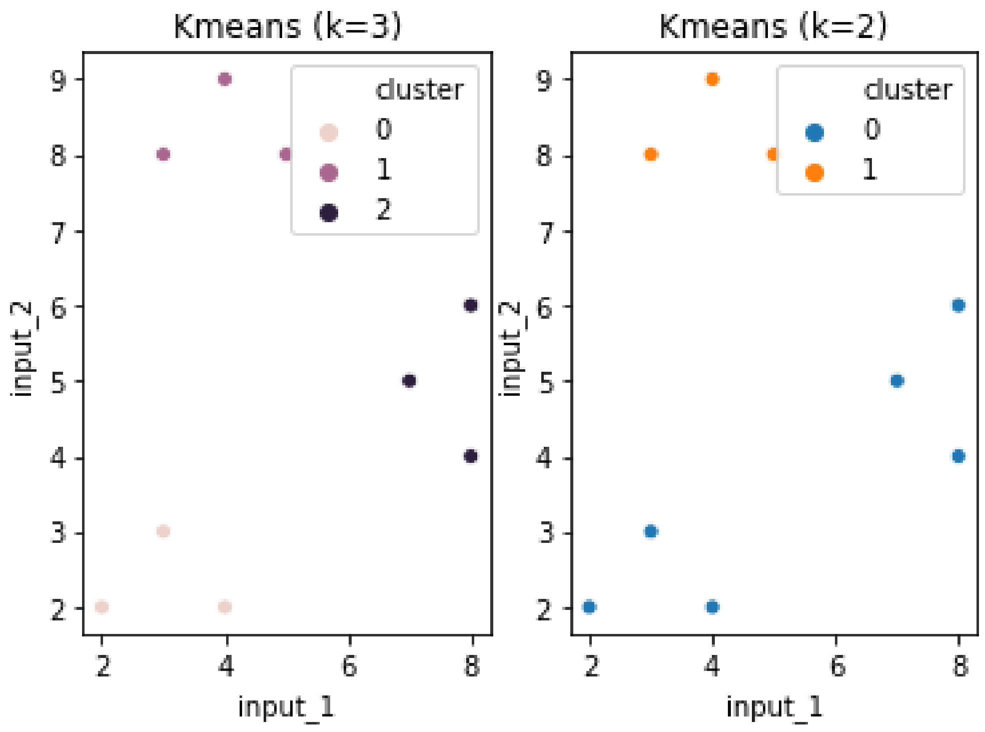
[66]:
```
# 4. repeat the process, but this time initialize at k = 2
data_df_2= k_means(2, data_df,50).copy()
sns.scatterplot(x ='input_1', y='input_2', hue="cluster", data=data_df_2)
```

[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a225be550>
```



```
[67]: fig, ax =plt.subplots(1,2)
      sns.scatterplot(x ='input_1', y='input_2', hue="cluster", data=data_df_3,␣
       ↪ax=ax[0])
      ax[0].set_title("Kmeans (k=3)")
      sns.scatterplot(x ='input_1', y='input_2', hue="cluster", data=data_df_2,␣
       ↪ax=ax[1])
      ax[1].set_title("Kmeans (k=2)")
```

```
[67]: Text(0.5,1,'Kmeans (k=2)')
```



From the two visualizations, we could find that k=3 fit the data better. This is because k=3 is more accurate in describing the inner patterns of our data. Other cluster numbers like k=2 however, does not fit our data very well.

2 Application

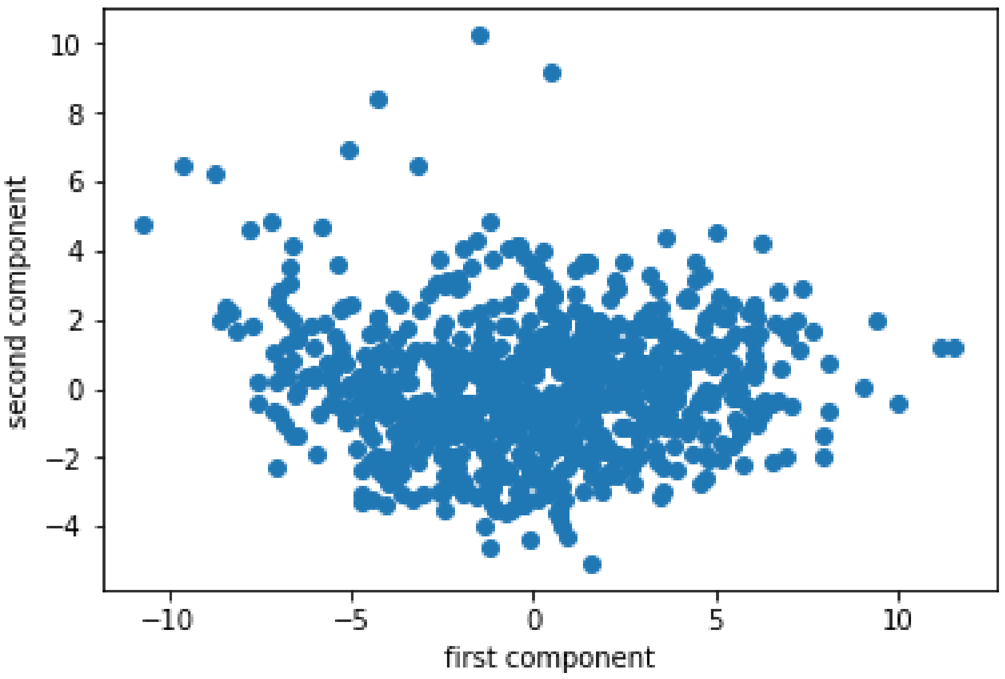
[70]:
```
wiki_df=pd.read_csv("./data/wiki.csv")
wiki_df=pd.DataFrame(StandardScaler().fit_transform(wiki_df), columns=wiki_df.
 ↪columns)
```

2.1 Dimension reduction

[73]:
```
# Perform PCA on the dataset
pca = PCA(n_components=2, random_state=200)
wiki_new=pca.fit_transform(wiki_df)
```

[76]:
```
# plot the components
plt.scatter(wiki_new[:,0], wiki_new[:,1])
plt.xlabel('first component')
plt.ylabel('second component')
```

[76]:
```
Text(0,0.5,'second component')
```



```
[81]: pca.components_.shape
```

```
[81]: (2, 57)
```

```
[90]: # feature correlation
      components=pd.DataFrame(pca.components_, columns=wiki_df.columns).T
      components.columns=['PC1', 'PC2']
```

```
[91]: components.nlargest(5, 'PC1')
```

```
[91]:             PC1       PC2
      bi2   0.230924  0.083441
      bi1   0.226193  0.056383
      use3  0.218809  0.155164
      use4  0.214558  0.160873
      pu3   0.210863  0.028799
```

```
[92]: components.nlargest(5, 'PC2')
```

```
[92]:                                  PC1       PC2
      exp4                        0.099873  0.228471
      use2                        0.147852  0.218622
      use1                        0.181477  0.197830
      vis3                        0.175351  0.197639
```

```
domain_Engineering_Architecture  0.051309  0.171497
```

The top 5 variables that are strongly correlated to the first principal component are bi2, bi1, use3, use4, pu3. The top 5 variables that are strongly correlated to the first principal component are exp4, use2, use1, domain_Engineering_Architecture, vis3. Hence, the first component seems to be related to the conception of wiki while the second component is about real experience.

[94]:
```
# individual PVE
pca.explained_variance_ratio_
```

[94]:
```
array([0.22810628, 0.06372474])
```

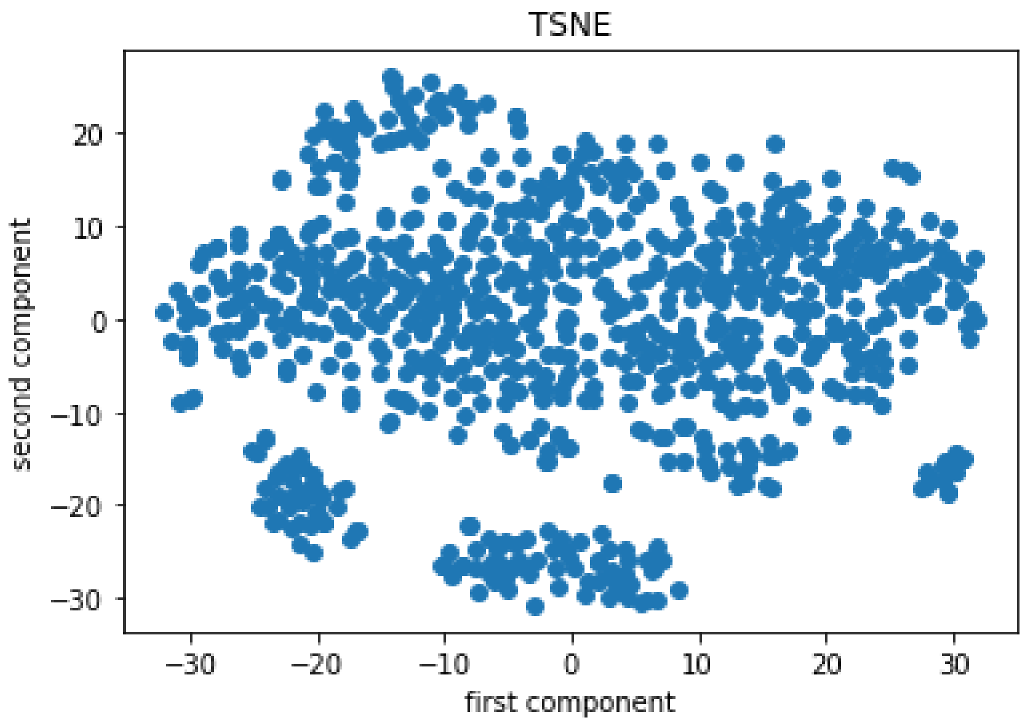
[96]:
```
# cumulative PVE
cum_pve=pca.explained_variance_ratio_[0]+pca.explained_variance_ratio_[1]
cum_pve
```

[96]:
```
0.29183102087597385
```

The proportion of variance explained (PVE) of the first component is 22.8%, the PVE of the second component is 6.4%. The cumulative PVE of the two componets is 29.2%

[97]:
```
# Perform t-SNE on the dataset and plot the observations
tsne=TSNE(n_components=2, random_state=200)
wiki_tsne=tsne.fit_transform(wiki_df)

# plot the components
plt.scatter(wiki_tsne[:,0], wiki_tsne[:,1])
plt.title("TSNE")
plt.xlabel('first component')
plt.ylabel('second component')
```

[97]:
```
Text(0,0.5,'second component')
```



From the above TSNE plot, we could see that in the TSNE plot, there are many small clusters around a central, large cluster. Such pattern is even evident when compared to the PCA plot which seems to be a single large cluster. Such difference is caused by the algorithm difference of PCA and TSNE. While PCA focuses more about linear/global pattern,TSNE tends to focus on non-linear, local patterns.

2.2 Clustering

[102]:
```
wiki_pca_kmeans=pd.DataFrame.from_dict({"PC1":wiki_new[:,0],"PC2":wiki_new[:
 ↪,1]})
```

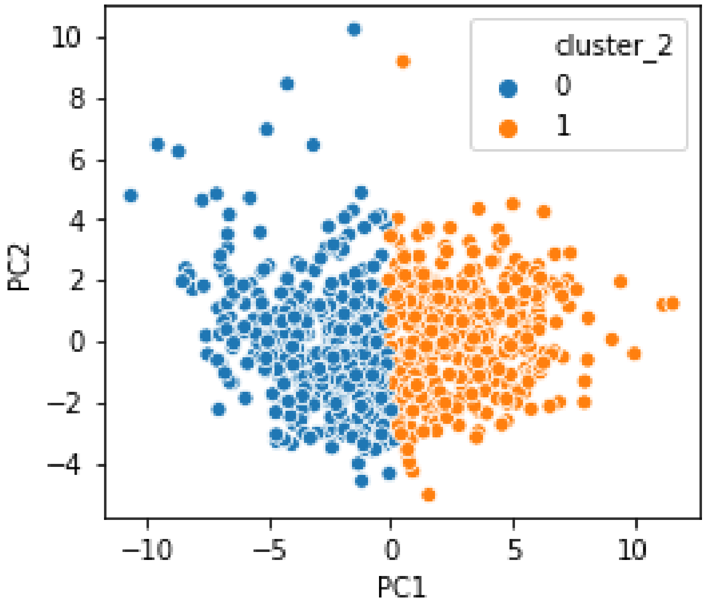
[104]:
```
# kmeans
kmeans_2 = KMeans(n_clusters=2, random_state=200).fit(wiki_df)
wiki_pca_kmeans['cluster_2']=kmeans_2.labels_
```

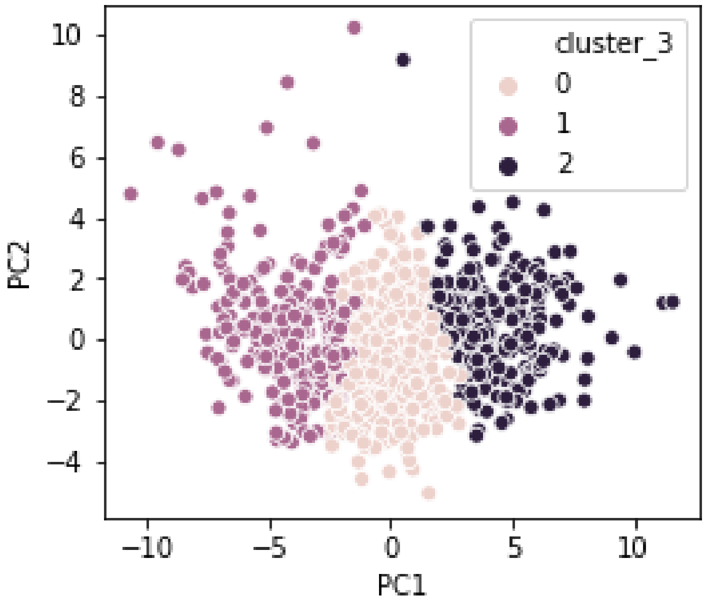
[107]:
```
# kmeans
kmeans_3 = KMeans(n_clusters=3, random_state=200).fit(wiki_df)
wiki_pca_kmeans['cluster_3']=kmeans_3.labels_
```

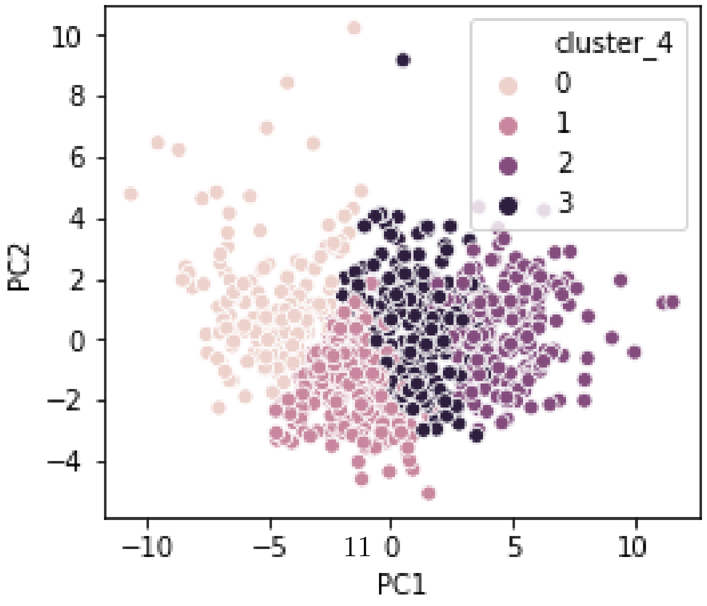
[106]:
```
# kmeans
kmeans_4 = KMeans(n_clusters=4, random_state=200).fit(wiki_df)
wiki_pca_kmeans['cluster_4']=kmeans_4.labels_
```

```
[110]: # visualization
       fig, ax=plt.subplots(3,1, figsize=(4, 12))
       sns.scatterplot(x='PC1', y='PC2', hue='cluster_2', data=wiki_pca_kmeans,␣
        ↪ax=ax[0])
       sns.scatterplot(x='PC1', y='PC2', hue='cluster_3', data=wiki_pca_kmeans,␣
        ↪ax=ax[1])
       sns.scatterplot(x='PC1', y='PC2', hue='cluster_4', data=wiki_pca_kmeans,␣
        ↪ax=ax[2])
```

```
[110]: <matplotlib.axes._subplots.AxesSubplot at 0x1a246a8190>
```


cluster_2
0
1
PC2
PC1
10
8
6
4
2
0
−2
−4
−10
−5
0
5
10


cluster_3
0
1
2
PC2
PC1
10
8
6
4
2
0
−2
−4
−10
−5
0
5
10

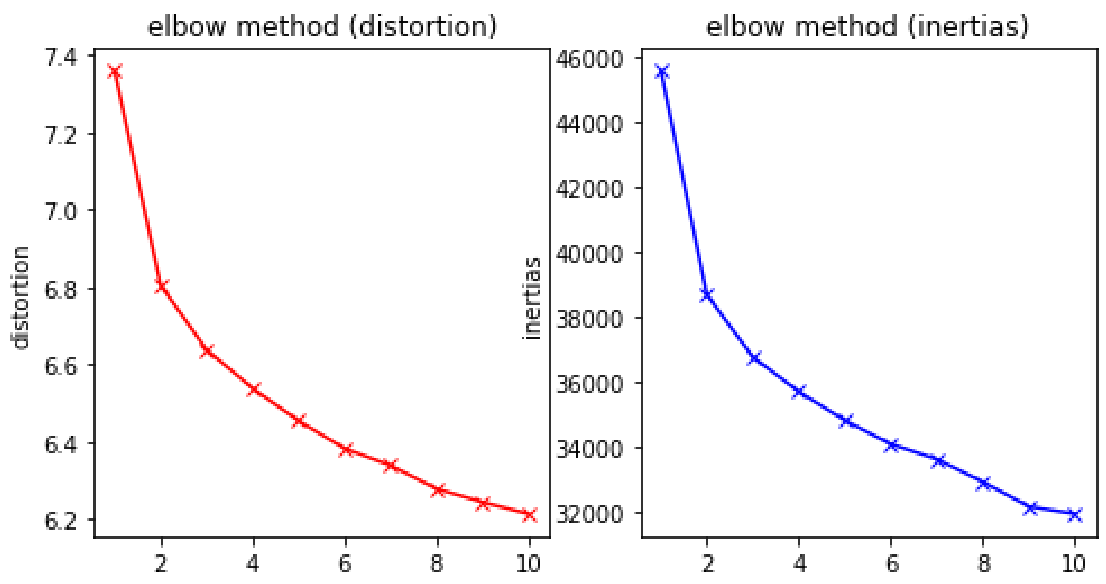

cluster_4
0
1
2
3
PC2
PC1
10
8
6
4
2
0
−2
−4
−10
−5
0
5
10

From the above graph, we could see that the decision of k cluster division is mainly based on the value of the first component of the PCA. Generally, when k=2, the cluster is decided by split the data into half (those with positive PC1 value and those with negative PC1 value). Similar pattern could also be observed when k=3 and k=4. As the number of cluster increase, the PC2 starts to make some impact as well.

[140]:
```
# optimal number of clusters
# elbow method
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
K = range(1,11)

for k in K:
    #Building and fitting the model
    kmeanModel = KMeans(n_clusters=k, random_state=200).fit(wiki_df)
    kmeanModel.fit(wiki_df)

    distortions.append(sum(np.min(cdist(wiki_df, kmeanModel.cluster_centers_,
                      'euclidean'),axis=1)) / wiki_df.shape[0])
    inertias.append(kmeanModel.inertia_)
```

[150]:
```
fig, ax=plt.subplots(1, 2, figsize=(8, 4), sharey=False)
ax[0].plot(K, distortions,'rx-')
ax[1].plot(K, inertias, 'bx-')
ax[0].set_ylabel("distortion")
ax[1].set_ylabel("inertias")
ax[0].set_title("elbow method (distortion)")
ax[1].set_title("elbow method (inertias)")
```

[150]:
```
Text(0.5,1,'elbow method (inertias)')
```

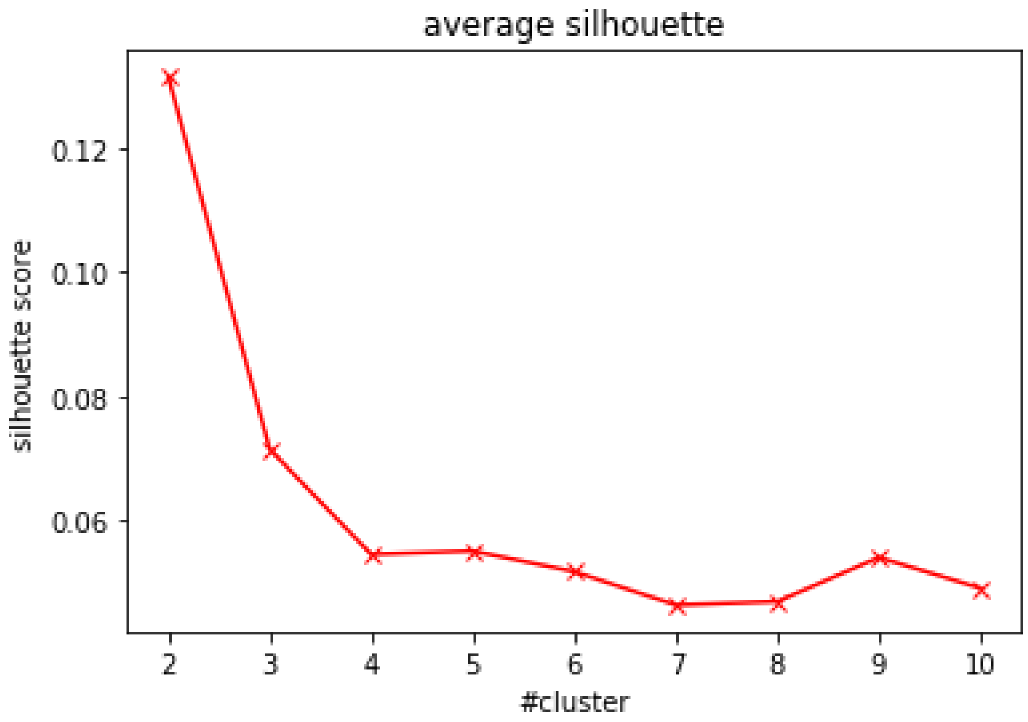


From the above graph, we could see that the elbow is at k=2.

[142]:
```
# average silhouette
avg_silk=[]

for k in range(2,11):
    clusterer = KMeans(n_clusters=k, random_state=200)
    cluster_labels = clusterer.fit_predict(wiki_df)
    silhouette_avg = silhouette_score(wiki_df, cluster_labels)
    avg_silk.append(silhouette_avg)
```

[148]:
```
plt.plot(np.arange(2,11), avg_silk, 'rx-')
plt.title("average silhouette")
plt.xlabel("#cluster")
plt.ylabel("silhouette score")
```

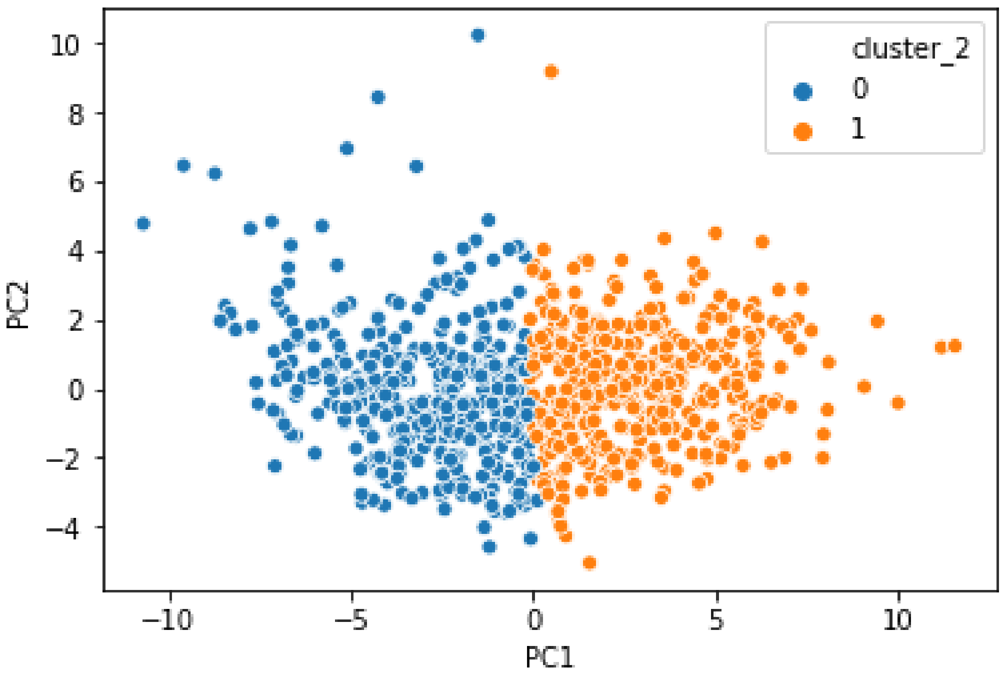
[148]:
```
Text(0,0.5,'silhouette score')
```



From the silhouette score, we could see that the optimal k value is 4. Since we got two different value from silhouette analysis and elbow method, we just decide to choose k=2 as the optimal value.

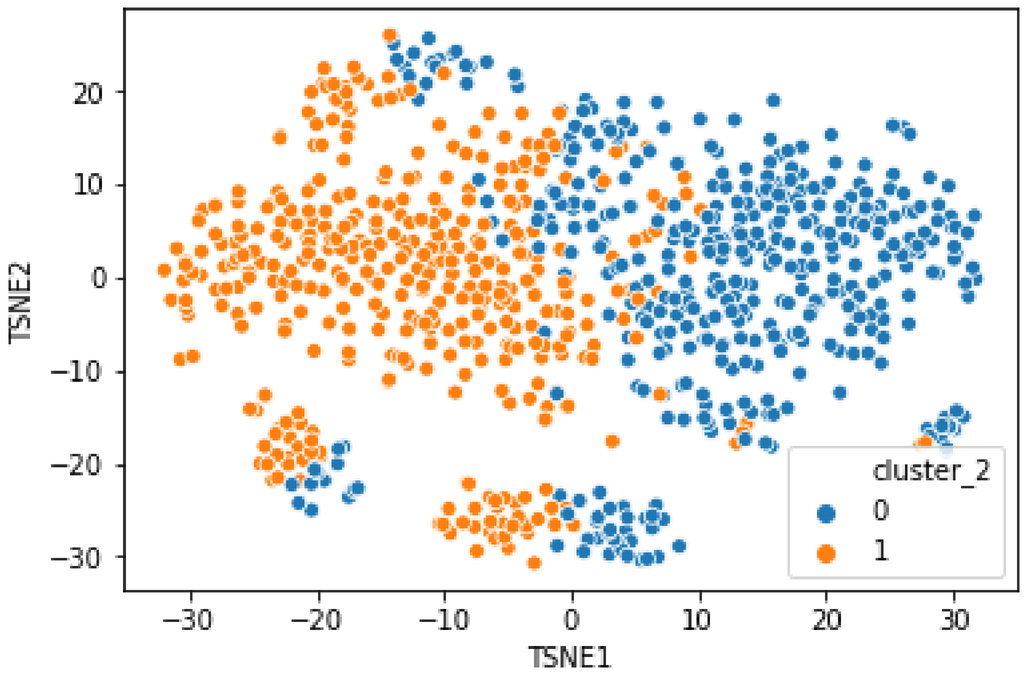
[144]:
```
# kmeans
kmeans_2 = KMeans(n_clusters=2, random_state=200).fit(wiki_df)
wiki_pca_kmeans['cluster_2']=kmeans_2.labels_
sns.scatterplot(x='PC1', y='PC2', hue='cluster_2', data=wiki_pca_kmeans)
```

[144]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a27c2c5d0>
```



```
[147]: # tsne
       wiki_tsne_kmeans=pd.DataFrame.from_dict({"TSNE1":wiki_tsne[:,0],"TSNE2":
        ↪wiki_tsne[:,1]})
       wiki_tsne_kmeans['cluster_2']=kmeans_2.labels_
       sns.scatterplot(x='TSNE1', y='TSNE2', hue='cluster_2', data=wiki_tsne_kmeans)
```

```
[147]: <matplotlib.axes._subplots.AxesSubplot at 0x1a260aa990>
```



From the above graph, we could see that PCA could clearly seperate the two clusters based on the value of its first component. Moreover, PCA seperation has no overlaps. As for TSNE, the boundar between the two clusters is not very decisive. The difference between PCA and TSNE is caused by their different emphasis on data patterns. While PCA cares more about the global structure, TSNE is focused on local proximity, hence lead to small clusters and fuzzy boundaries. In this specific case of K=2, PCA is a better fit than TSNE.