Xiong_Yinjiang_hw7

March 15, 2020

0.1 Unsupervised Learning

0.1.1 k-Means Clustering "By Hand"

[88]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
```

1.Imitate the k-means random initialization part of the algorithm by assigning each observation to a cluster at random.

[22]:
```
np.random.seed(24)
input_1 = [5,8,7,8,3,4,2,3,4,5]
input_2 = [8,6,5,4,3,2,2,8,9,8]
clusters = np.random.choice(3, 10)
df = pd.DataFrame({'input_1':input_1, 'input_2':input_2})
df['cluster'] = clusters
df
```

[22]:

	input_1	input_2	cluster
0	5	8	2
1	8	6	0
2	7	5	1
3	8	4	1
4	3	3	1
5	4	2	0
6	2	2	0
7	3	8	2
8	4	9	1
9	5	8	2

2.Compute the cluster centroid and update cluster assignments for each observation iteratively based on spatial similarity.

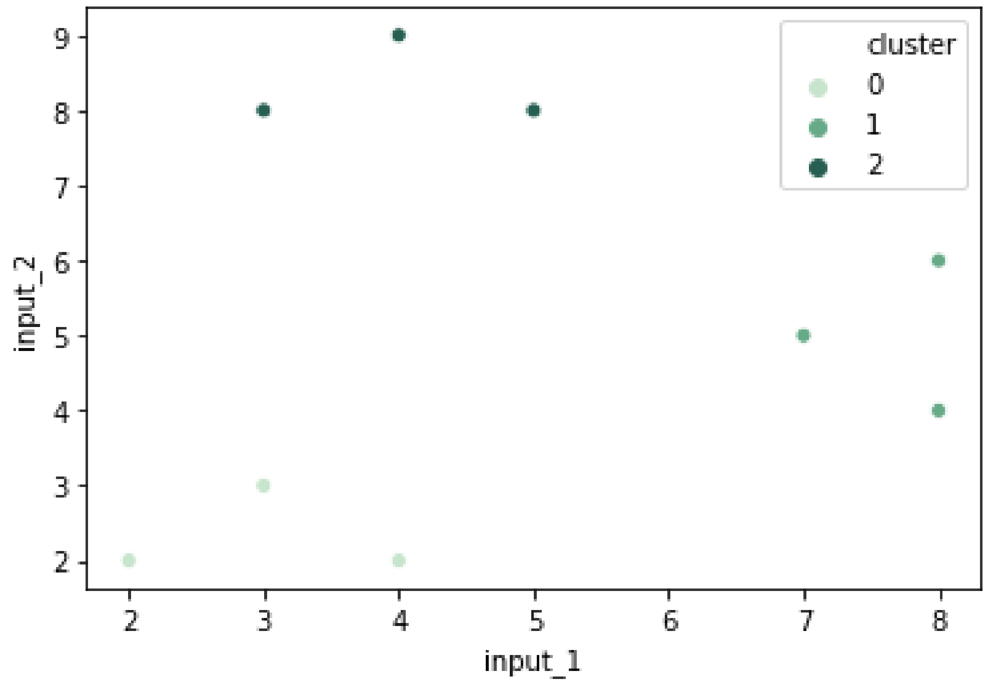
```
[42]: # 10 iterations
def fit(k_num, df):
    fit_df = df.copy()
    for _ in range(10):
        centroid = {}
        new_labels = []

        for k in range(k_num):
            cent1 = fit_df[fit_df['cluster'] == k]['input_1'].mean()
            cent2 = fit_df[fit_df['cluster'] == k]['input_2'].mean()
            centroid[k] = (cent1, cent2)


        for idx, row in fit_df.iterrows():
            min_distance = 100
            for k, v in centroid.items():
                eu_distance = np.sqrt((row['input_1'] - v[0]) ** 2 +␣
 ↪(row['input_2'] - v[1]) ** 2)
                if eu_distance < min_distance:
                    min_distance = eu_distance
                    new_k = k
            new_labels.append(new_k)
        fit_df['cluster'] = new_labels
    return fit_df

fit_3 = fit(3, df)
fit_3
```

[42]:

	input_1	input_2	cluster
0	5	8	2
1	8	6	1
2	7	5	1
3	8	4	1
4	3	3	0
5	4	2	0
6	2	2	0
7	3	8	2
8	4	9	2
9	5	8	2

3.Present a visual description of the final, converged (stopped) cluster assignments.

```
[52]: sns.scatterplot(x='input_1', y='input_2', hue='cluster', data=fit_3, palette='ch:
 ↪2.5,-.2,dark=.3');
```



4.Now, repeat the process, but this time initialize at k = 2 and present a final cluster assignment visually next to the previous search at k = 3

[53]:
```
def fit(k_num, df):
    fit_df = df.copy()
    for _ in range(10):
        centroid = {}
        new_labels = []

        for k in range(k_num):
            cent1 = fit_df[fit_df['cluster'] == k]['input_1'].mean()
            cent2 = fit_df[fit_df['cluster'] == k]['input_2'].mean()
            centroid[k] = (cent1, cent2)


        for idx, row in fit_df.iterrows():
            min_distance = 100
            for k, v in centroid.items():
                eu_distance = np.sqrt((row['input_1'] - v[0]) ** 2 +␣
↪(row['input_2'] - v[1]) ** 2)
                if eu_distance < min_distance:
                    min_distance = eu_distance
                    new_k = k
            new_labels.append(new_k)
        fit_df['cluster'] = new_labels
```

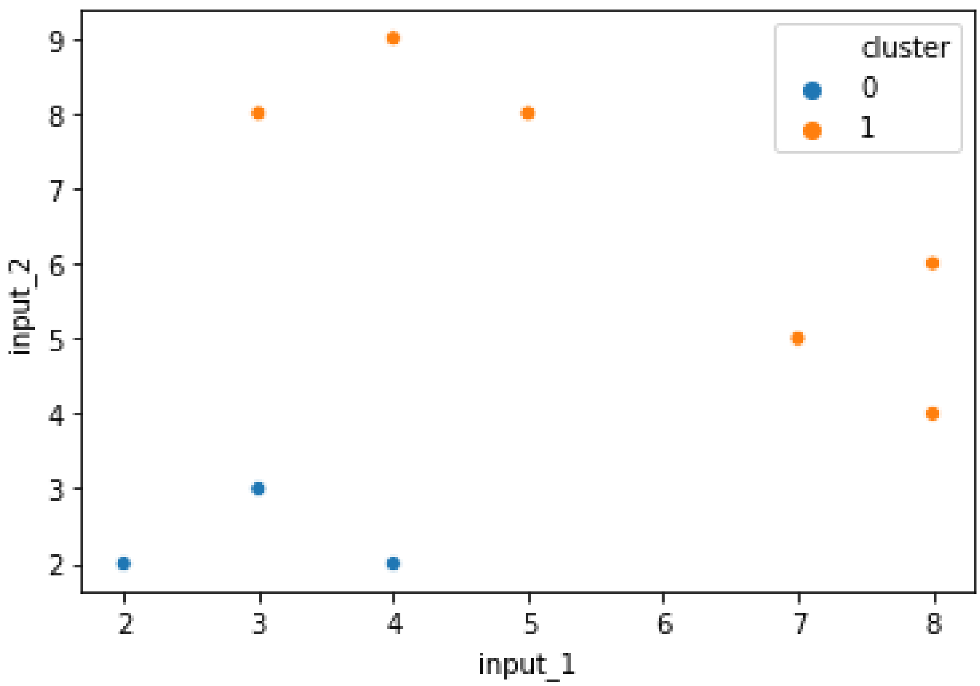
```
    return fit_df

fit_2 = fit(2, df)
fit_2
```

[53]:

	input_1	input_2	cluster
0	5	8	1
1	8	6	1
2	7	5	1
3	8	4	1
4	3	3	0
5	4	2	0
6	2	2	0
7	3	8	1
8	4	9	1
9	5	8	1

[54]:

```
sns.scatterplot(x='input_1', y='input_2', hue='cluster', data=fit_2);
```



5.Did your initial hunch of 3 clusters pan out, or would other values of k, like 2, fit these data better? Why or why not?

3 clusters apparently work better than 2, as visually there are two subgroups in cluster 1.

0.1.2 Application

[55]:
```
wiki = pd.read_csv('wiki.csv')
```

6.Perform PCA on the dataset and plot the observations on the first and second principal components. Describe your results, e.g.,

What variables appear strongly correlated on the first principal component? What about the second principal component?

[57]:
```
wiki.head(5)
```

[57]:
```
   age  gender  phd  yearsexp  userwiki  pu1  pu2  pu3  peu1  peu2  ...  exp5  \
0   40       0    1        14         0    4    4    3     5     5  ...     2
1   42       0    1        18         0    2    3    3     4     4  ...     4
2   37       0    1        13         0    2    2    2     4     4  ...     3
3   40       0    0        13         0    3    3    4     3     3  ...     4
4   51       0    0         8         1    4    3    5     5     4  ...     4

   domain_Sciences  domain_Health.Sciences  domain_Engineering_Architecture  \
0                1                       0                                0
1                0                       0                                0
2                0                       0                                1
3                0                       0                                1
4                0                       0                                1

   domain_Law_Politics  uoc_position_Associate  uoc_position_Assistant  \
0                    0                       1                       0
1                    1                       1                       0
2                    0                       0                       1
3                    0                       0                       1
4                    0                       0                       1

   uoc_position_Lecturer  uoc_position_Instructor  uoc_position_Adjunct
0                      0                        0                     0
1                      0                        0                     0
2                      0                        0                     0
3                      0                        0                     0
4                      0                        0                     0

[5 rows x 57 columns]
```

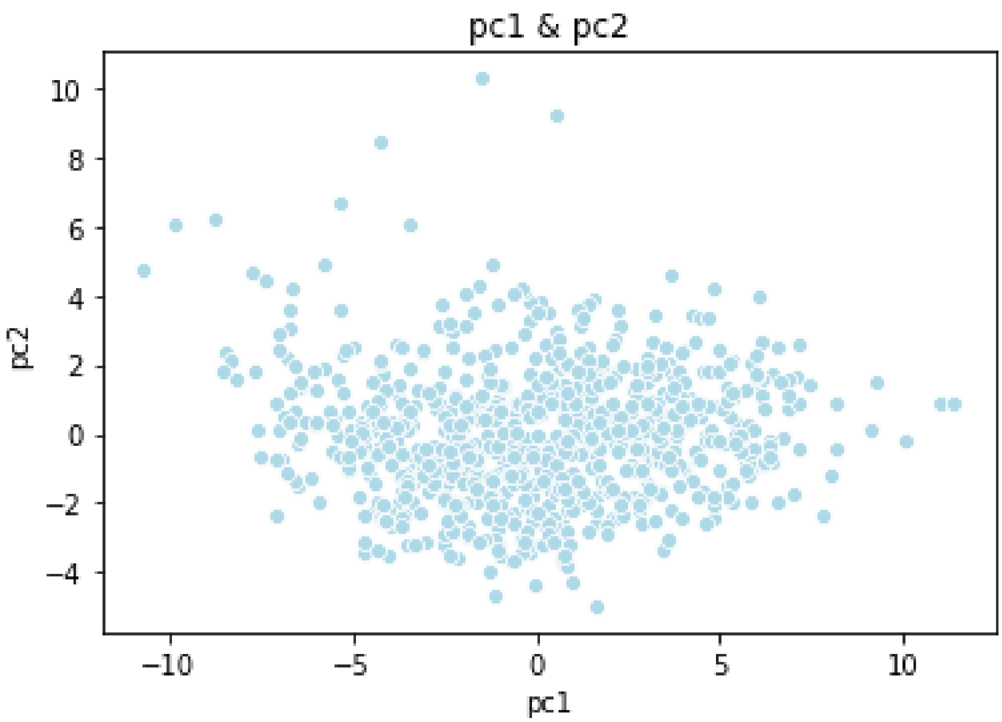
[60]:
```
# normalize
X = wiki.drop('userwiki', axis=1)
X = scale(X)
```

[62]:
```
pca = PCA(n_components=2).fit(X)
```

[63]:
```
pc = pca.transform(X)
```

[65]:
```
df_pc = pd.DataFrame(pc, columns=['pc1','pc2'])
```

[68]:
```
sns.scatterplot(df_pc['pc1'],df_pc['pc2'], color='lightblue')
plt.title('pc1 & pc2');
```



[72]:
```
names = wiki.drop('userwiki', axis=1).columns
```

[73]:
```
df_pca = pd.DataFrame({'variable': names,'pc1': pca.components_[0], 'pc2':  pca.
↪components_[1]})
df_pca['pc1_abs'] = df_pca['pc1'].abs()
df_pca['pc2_abs'] = df_pca['pc2'].abs()
df_pca
```

[73]:

	variable	pc1	pc2	pc1_abs	pc2_abs
0	age	-0.022077	0.092033	0.022077	0.092033
1	gender	-0.034886	-0.155387	0.034886	0.155387
2	phd	-0.030691	0.034753	0.030691	0.034753
3	yearsexp	-0.034243	0.070067	0.034243	0.070067
4	pu1	0.193727	0.015663	0.193727	0.015663
5	pu2	0.191626	0.027574	0.191626	0.027574
6	pu3	0.211832	0.038821	0.211832	0.038821

7	peu1	0.062337	-0.271289	0.062337	0.271289
8	peu2	0.114558	-0.221201	0.114558	0.221201
9	peu3	0.100383	-0.075439	0.100383	0.075439
10	enj1	0.146710	-0.147221	0.146710	0.147221
11	enj2	0.132156	-0.228262	0.132156	0.228262
12	qu1	0.178945	-0.028277	0.178945	0.028277
13	qu2	0.164528	-0.059215	0.164528	0.059215
14	qu3	0.159126	-0.022451	0.159126	0.022451
15	qu4	-0.061584	-0.116239	0.061584	0.116239
16	qu5	0.184129	0.018107	0.184129	0.018107
17	vis1	0.171639	-0.025255	0.171639	0.025255
18	vis2	0.114412	-0.064030	0.114412	0.064030
19	vis3	0.175155	0.198986	0.175155	0.198986
20	im1	0.160947	0.119045	0.160947	0.119045
21	im2	0.078226	-0.059122	0.078226	0.059122
22	im3	0.161703	0.052045	0.161703	0.052045
23	sa1	0.122289	-0.236572	0.122289	0.236572
24	sa2	0.118397	-0.233422	0.118397	0.233422
25	sa3	0.121358	-0.246815	0.121358	0.246815
26	use1	0.181365	0.200756	0.181365	0.200756
27	use2	0.147354	0.216021	0.147354	0.216021
28	use3	0.219028	0.160997	0.219028	0.160997
29	use4	0.214833	0.166242	0.214833	0.166242
30	use5	0.207593	0.040924	0.207593	0.040924
31	pf1	0.101548	0.100047	0.101548	0.100047
32	pf2	0.103210	0.005079	0.103210	0.005079
33	pf3	0.109020	0.081363	0.109020	0.081363
34	jr1	0.081736	-0.140190	0.081736	0.140190
35	jr2	0.062673	-0.112953	0.062673	0.112953
36	bi1	0.226856	0.060833	0.226856	0.060833
37	bi2	0.231545	0.088570	0.231545	0.088570
38	inc1	0.105760	-0.248794	0.105760	0.248794
39	inc2	0.096978	-0.199878	0.096978	0.199878
40	inc3	0.082408	-0.222931	0.082408	0.222931
41	inc4	0.090387	-0.206284	0.090387	0.206284
42	exp1	0.209351	0.077729	0.209351	0.077729
43	exp2	0.195861	-0.025129	0.195861	0.025129
44	exp3	0.144484	-0.127559	0.144484	0.127559
45	exp4	0.097582	0.210692	0.097582	0.210692
46	exp5	0.110325	0.069560	0.110325	0.069560
47	domain_Sciences	0.022783	-0.004298	0.022783	0.004298
48	domain_Health.Sciences	-0.017386	-0.017971	0.017386	0.017971
49	domain_Engineering_Architecture	0.051131	0.176930	0.051131	0.176930
50	domain_Law_Politics	-0.095381	-0.020834	0.095381	0.020834
51	uoc_position_Associate	0.010699	0.014058	0.010699	0.014058
52	uoc_position_Assistant	0.007174	0.003580	0.007174	0.003580
53	uoc_position_Lecturer	-0.017970	-0.024496	0.017970	0.024496

```
54          uoc_position_Instructor  0.004428 -0.003063  0.004428  0.003063
55             uoc_position_Adjunct -0.007748 -0.006650  0.007748  0.006650
```

[74]:
```python
df_pca.sort_values(by=['pc1_abs'], ascending=False).head(5)
```

[74]:

	variable	pc1	pc2	pc1_abs	pc2_abs
37	bi2	0.231545	0.088570	0.231545	0.088570
36	bi1	0.226856	0.060833	0.226856	0.060833
28	use3	0.219028	0.160997	0.219028	0.160997
29	use4	0.214833	0.166242	0.214833	0.166242
6	pu3	0.211832	0.038821	0.211832	0.038821

[75]:
```python
df_pca.sort_values(by=['pc2_abs'], ascending=False).head(5)
```

[75]:

	variable	pc1	pc2	pc1_abs	pc2_abs
7	peu1	0.062337	-0.271289	0.062337	0.271289
38	inc1	0.105760	-0.248794	0.105760	0.248794
25	sa3	0.121358	-0.246815	0.121358	0.246815
23	sa1	0.122289	-0.236572	0.122289	0.236572
24	sa2	0.118397	-0.233422	0.118397	0.233422

Most strongly correlated with PC1: bi2, bi1, use3, use4, pu3
Most strongly correlated with PC2: peu1, inc1, sa3, sa1, sa2

7.Calculate the proportion of variance explained (PVE) and the cumulative PVE for all the principal components. Approximately how much of the variance is explained by the first two principal components?

[76]:
```python
print(pca.explained_variance_ratio_[0]+pca.explained_variance_ratio_[1], 'of the variance is explained by the first two principal components.')
```

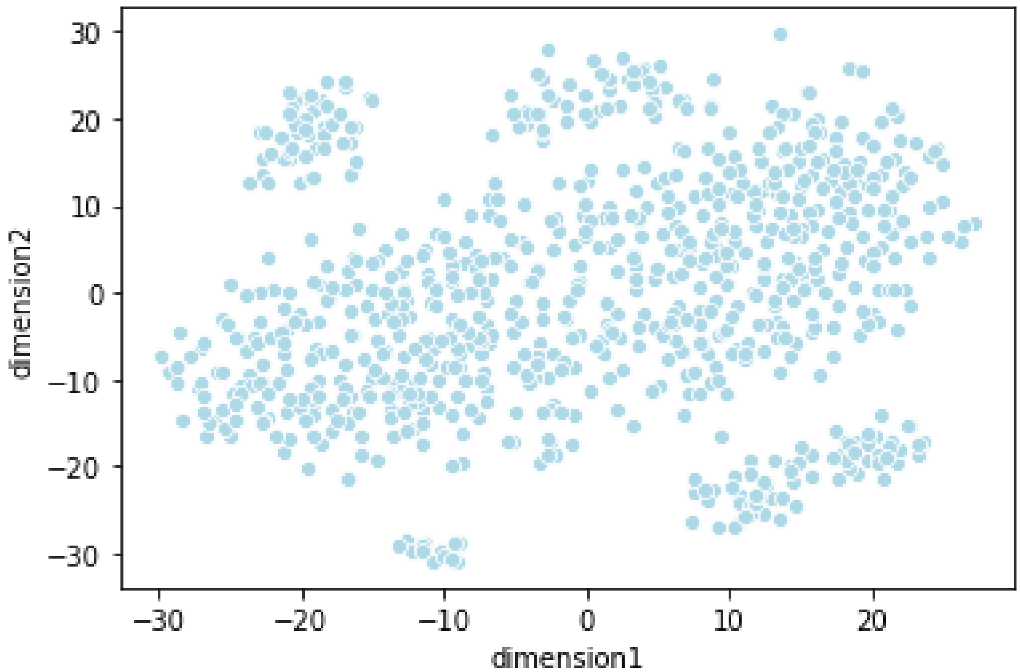
```
0.2947644036820181 of the variance is explained by the first two principal
components.
```

8.Perform *t*-SNE on the dataset and plot the observations on the first and second dimensions. Describe your results.

[78]:
```python
X_embedded = TSNE(n_components=2, random_state=2).fit_transform(X)
```

[79]:
```python
X_embedded = pd.DataFrame(X_embedded, columns = ['dimension1','dimension2'])
```

[80]:
```python
sns.scatterplot(X_embedded['dimension1'],X_embedded['dimension2'], color='lightblue');
```

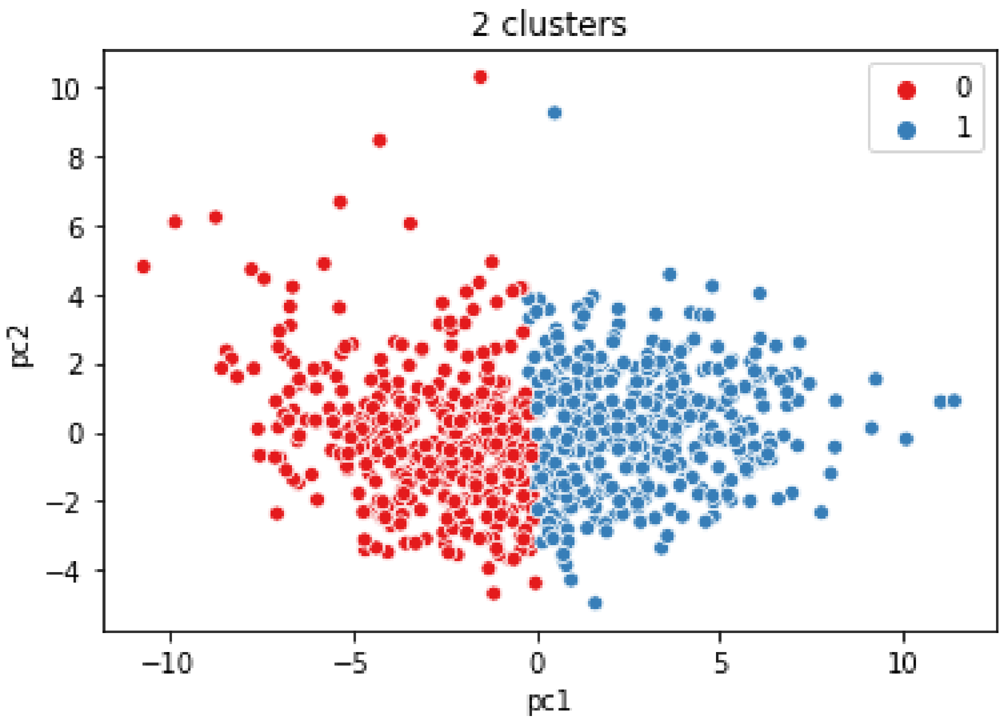


0.1.3 Clustering

9.Perform k-means clustering with $k = 2, 3, 4$. Be sure to scale each feature (i.e.,mean zero and standard deviation one). Plot the observations on the first and second principal components from PCA and color-code each observation based on their cluster membership. Discuss your results.

[82]:
```
km2 = KMeans(n_clusters=2).fit(X)
km3 = KMeans(n_clusters=3).fit(X)
km4 = KMeans(n_clusters=4).fit(X)
```

[85]:
```
sns.scatterplot(df_pc['pc1'],df_pc['pc2'], hue=km2.labels_, palette='Set1')
plt.title('2 clusters');
```

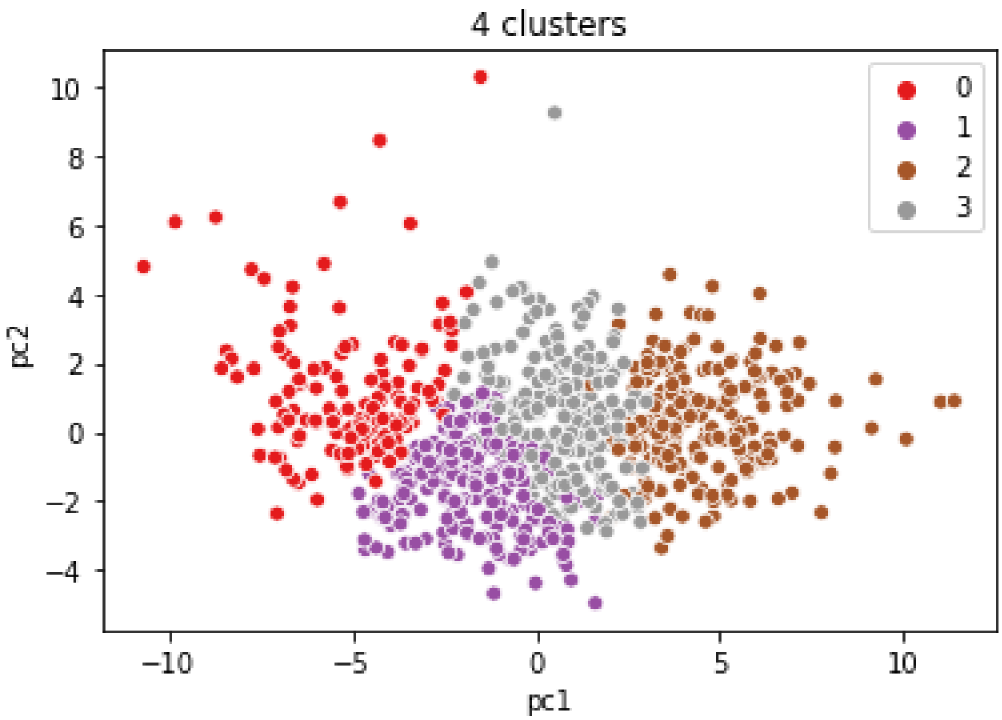


2 clusters seem to arbitrarilly divde the points up.

[86]:
```
sns.scatterplot(df_pc['pc1'],df_pc['pc2'], hue=km3.labels_, palette='Set1')
plt.title('3 clusters');
```

3 clusters don't seem to have clear boundaries either.

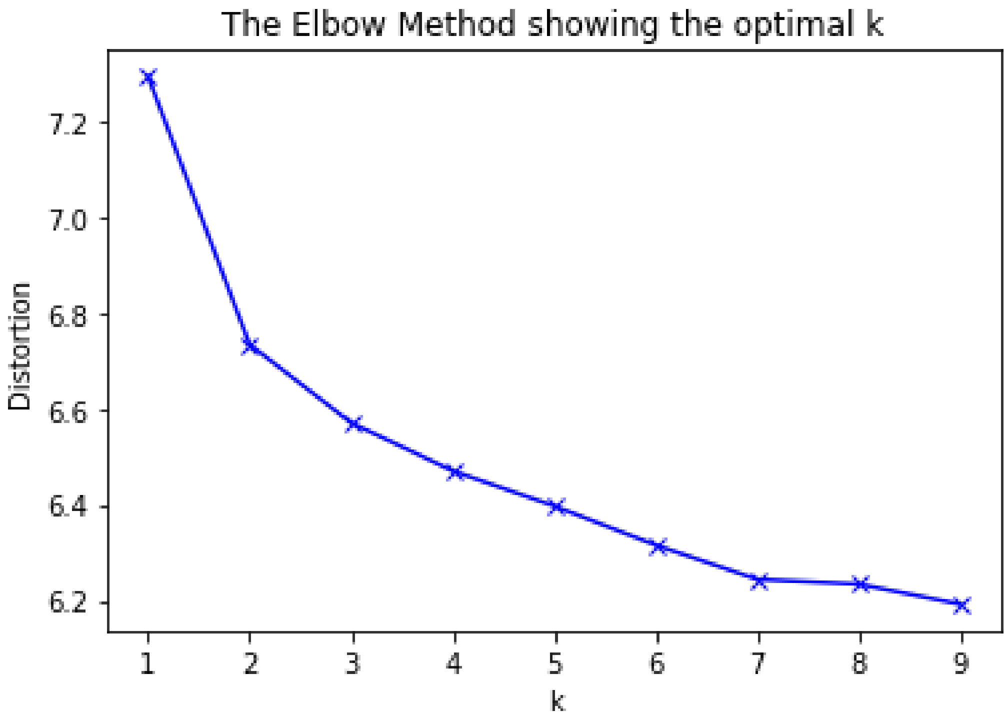
[87]:
```
sns.scatterplot(df_pc['pc1'],df_pc['pc2'], hue = km4.labels_, palette='Set1')
plt.title('4 clusters');
```



4 clusters are by far the best.

10.Use the elbow method, average silhouette, and/or gap statistic to identify the optimal number of clusters based on *k*-means clustering with scaled features.

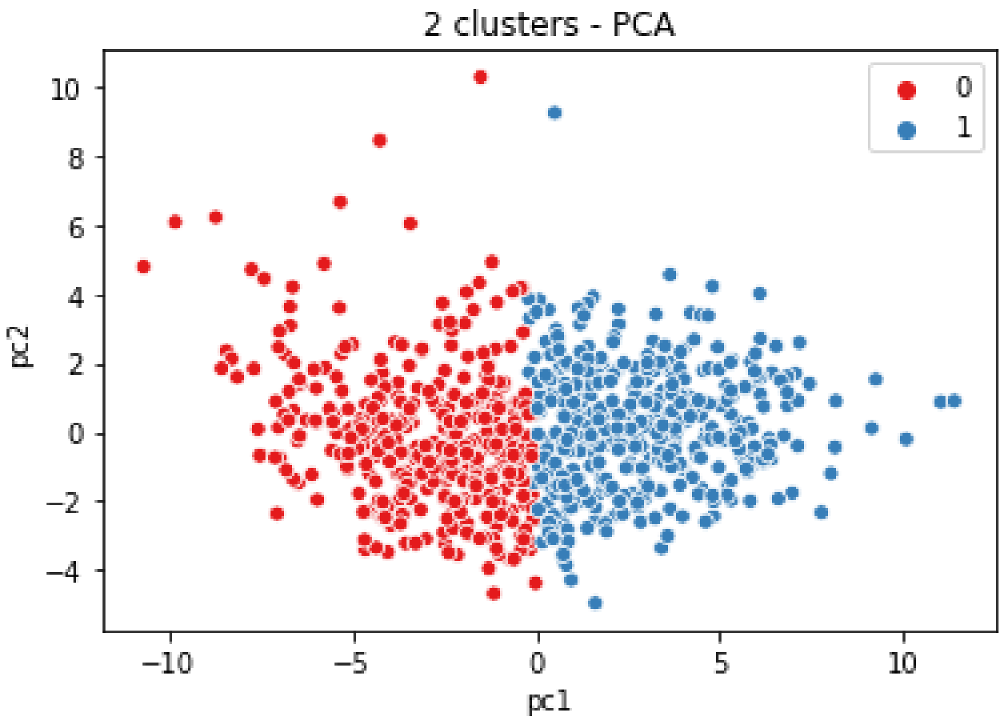
[89]:
```
# elbow method
# k means determine k
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(X)
    kmeanModel.fit(X)
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_,
 ↪'euclidean'), axis=1)) / X.shape[0])

# Plot the elbow
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



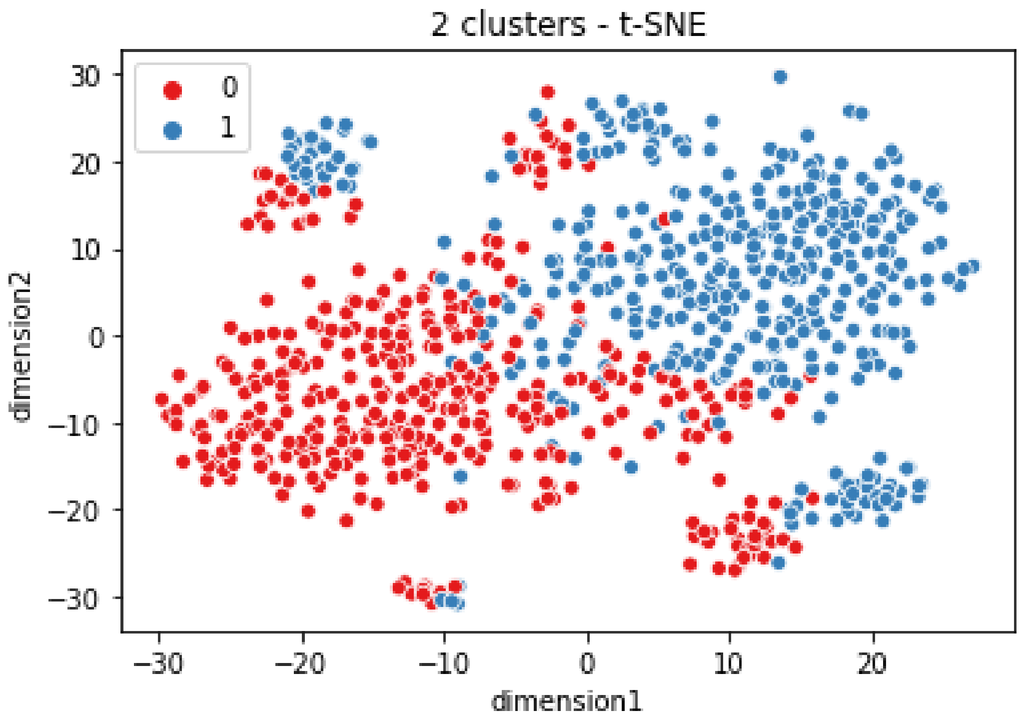
The elbow happened at 2.

11.Visualize the results of the optimal $\hat{k}$-means clustering model. First use the first and second principal components from PCA, and color-code each observation based on their cluster membership. Next use the first and second dimensions from t-SNE, and color-code each observation based on their cluster membership. Describe your results. How do your interpretations differ between PCA and t-SNE?

[91]:
```
sns.scatterplot(df_pc['pc1'],df_pc['pc2'], hue=km2.labels_, palette='Set1')
plt.title('2 clusters - PCA');
```



[93]:
```
sns.scatterplot(X_embedded['dimension1'],X_embedded['dimension2'], hue=km2.
 ↪labels_, palette='Set1')
plt.title('2 clusters - t-SNE');
```



It seems that PCA did a better job in splitting data into cluster membership.