# Jiang_Luying_HW7

March 15, 2020

[1]:
```
import random
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from gap_statistic import optimalK
import warnings
warnings.filterwarnings("ignore")
```

## 1 k-Means Clustering "By Hand"

You fielded an experiment and collected observations for 10 respondents across two features. The data are:

input_1 = c(5,8,7,8,3,4,2,3,4,5)

input_2 = c(8,6,5,4,3,2,2,8,9,8)

After inspecting your data, you suspect 3 clusters likely characterize these data, but you'd like to check your intuition. Perform k-means clustering "by hand" on these data, initializing at k = 3. Be sure to set the seed for reproducibility. Specifically:

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

[2]:
```
np.random.seed(0)
```

[3]:
```
input1 = [5,8,7,8,3,4,2,3,4,5]
input2 = [8,6,5,4,3,2,2,8,9,8]
label = np.random.choice(3,10)
df_k3 = pd.DataFrame({'x_1': input1, 'x_2': input2, 'k_label': label})
df_k3
```

```
[3]:    x_1  x_2  k_label
     0    5    8        0
     1    8    6        1
     2    7    5        0
     3    8    4        1
     4    3    3        1
     5    4    2        2
     6    2    2        0
     7    3    8        2
     8    4    9        0
     9    5    8        0
```

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

```
[4]: def k_means_clustering(k, df):
         for i in range(50):
             centroids = {}
             for i in range(k):
                 center1 = df[df['k_label'] == i]['x_1'].mean()
                 center2 = df[df['k_label'] == i]['x_2'].mean()
                 centroids[i] = (center1, center2)
             new_labels = []
             for index, row in df.iterrows():
                 min_distance = 30
                 for key, value in centroids.items():
                     distance = ((row['x_1'] - value[0])**2 +
                                 (row['x_2'] - value[1])**2)**0.5
                     if distance < min_distance:
                         min_distance = distance
                         new_k = key
                 new_labels.append(new_k)
             df['k_label'] = new_labels
         return df
```

```
[5]: k_3 = k_means_clustering(3, df_k3)
     k_3
```

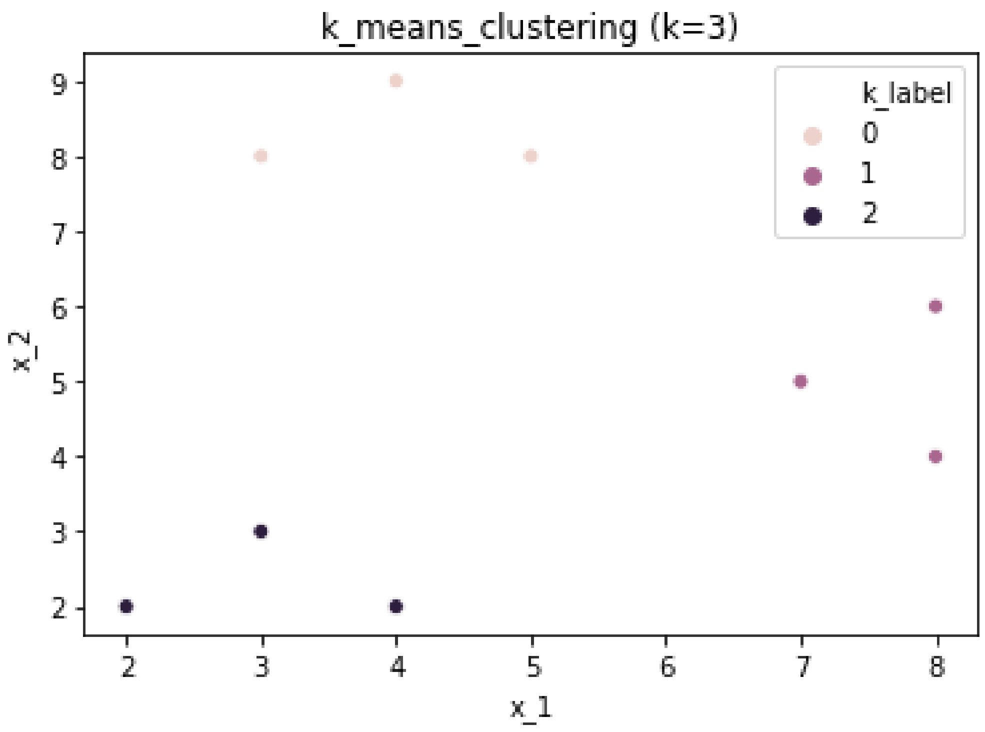
```
[5]:    x_1  x_2  k_label
     0    5    8        0
     1    8    6        1
     2    7    5        1
     3    8    4        1
     4    3    3        2
     5    4    2        2
     6    2    2        2
     7    3    8        0
```

```
8    4    9        0
9    5    8        0
```

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

```
[6]: sns.scatterplot(x='x_1', y='x_2', hue='k_label', data=k_3).
      ↪set_title('k_means_clustering (k=3)')
```

```
[6]: Text(0.5, 1.0, 'k_means_clustering (k=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.

```
[7]: input1 = [5,8,7,8,3,4,2,3,4,5]
     input2 = [8,6,5,4,3,2,2,8,9,8]
     label = np.random.choice(2,10)
     df_k2 = pd.DataFrame({'x_1': input1, 'x_2': input2, 'k_label': label})
     df_k2
```

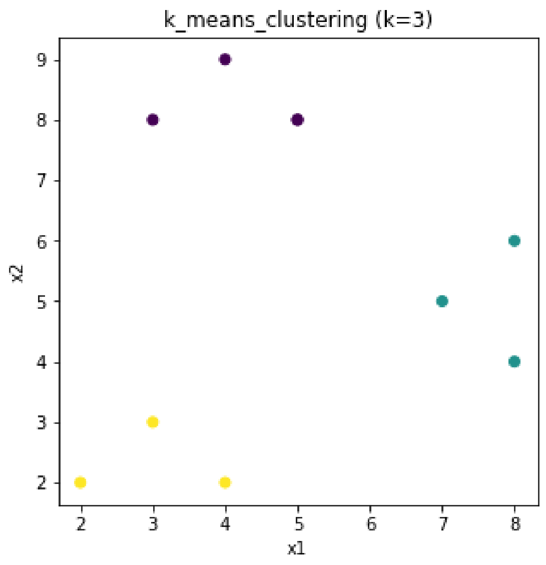
```
[7]:    x_1  x_2  k_label
     0    5    8        0
     1    8    6        0
     2    7    5        1
     3    8    4        0
```

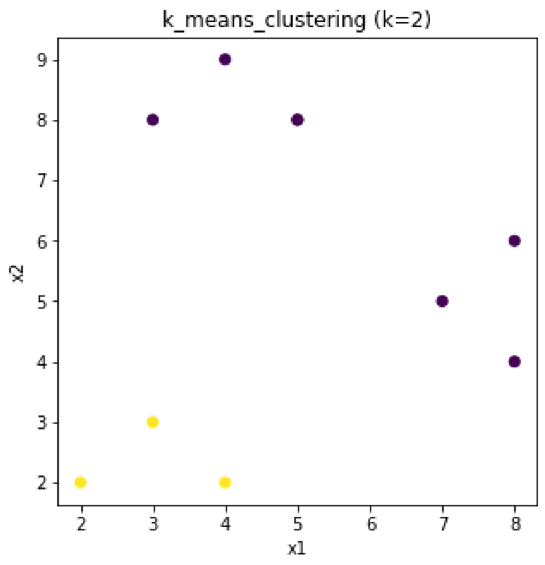
```
4    3    3         1
5    4    2         1
6    2    2         0
7    3    8         0
8    4    9         1
9    5    8         1
```

[8]:
```
k_2 = k_means_clustering(2, df_k2)
k_2
```

[8]:
```
   x_1  x_2  k_label
0    5    8        0
1    8    6        0
2    7    5        0
3    8    4        0
4    3    3        1
5    4    2        1
6    2    2        1
7    3    8        0
8    4    9        0
9    5    8        0
```

[9]:
```
fig, ax = plt.subplots(1, 2, figsize=(11,5))
ax[0].scatter(k_3['x_1'], k_3['x_2'], c=k_3['k_label'])
ax[0].set_xlabel('x1')
ax[0].set_ylabel('x2')
ax[0].set_title('k_means_clustering (k=3)')

ax[1].scatter(k_2['x_1'], k_2['x_2'], c=k_2['k_label'])
ax[1].set_xlabel('x1')
ax[1].set_ylabel('x2')
ax[1].set_title('k_means_clustering (k=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?

According to the above graph, the k-means with k=3 performs better than the k-means with k=2. In k=2, it clustered two groups into the same dark group. When k=3, we can see that groups are seperated clearly. Within-cluster distance is minimized and distance between different groups is maximized.

## 2 Application

wiki.csv contains a data set of survey responses from university faculty members related to their perceptions and practices of using Wikipedia as a teaching resource. Documentation for this dataset can be found at the UCI machine learning repository. The dataset has been pre-processed for you as follows:

- Include only employees of UOC and remove OTHER*, UNIVERSITY variables
- Impute missing values
- Convert domain and uoc_position to dummy variables

### 2.1 Dimension reduction

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?

[10]:
```
df_wiki = pd.read_csv('wiki.csv')
df_wiki.head()
```

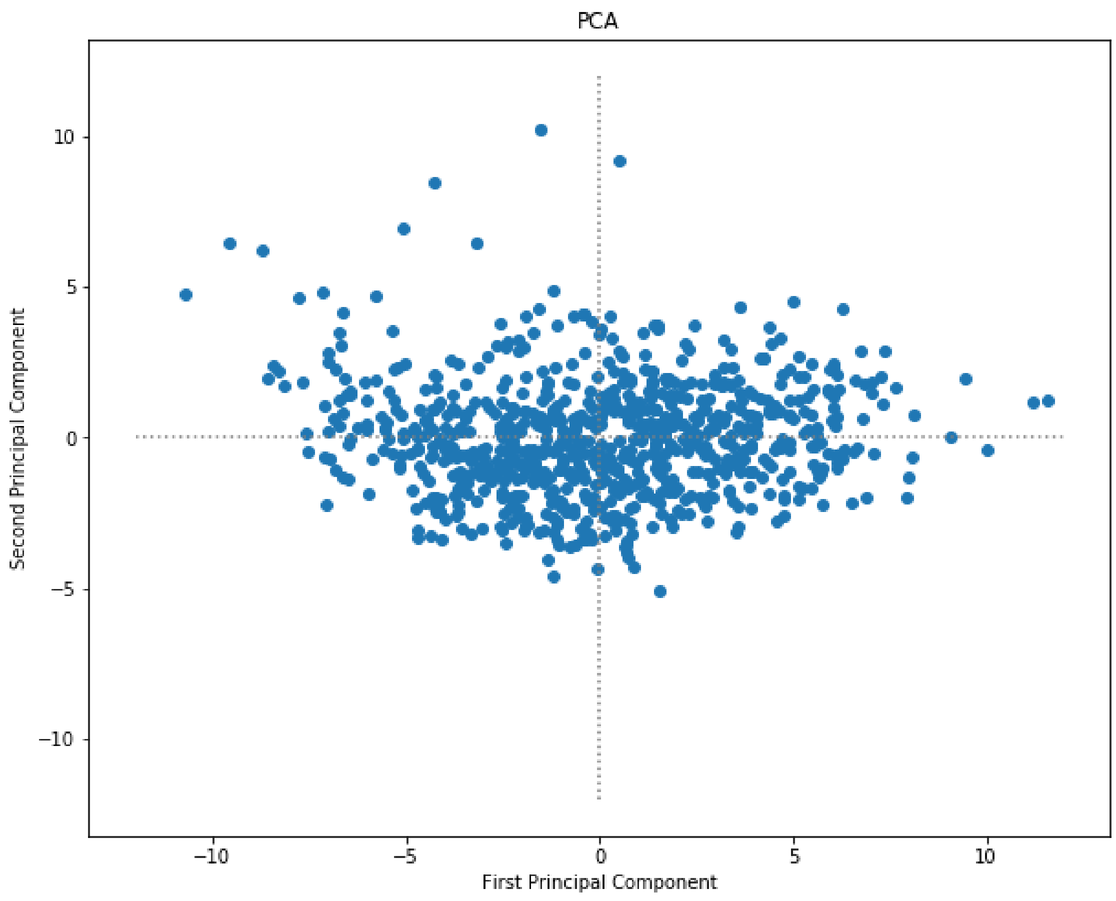
[10]:
```
   age  gender  phd  yearsexp  userwiki  pu1  pu2  pu3  peu1  peu2  …  exp5  \
0   40       0    1        14         0    4    4    3     5     5  …     2
1   42       0    1        18         0    2    3    3     4     4  …     4
2   37       0    1        13         0    2    2    2     4     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]
```

[11]:
```python
x = StandardScaler().fit_transform(df_wiki)
```

[12]:
```python
pca = PCA(n_components = 2, random_state = 0)
x_fit = pca.fit_transform(x)
df_pca = pd.DataFrame(data = x_fit, columns = ['PC1', 'PC2'])
```

[13]:
```python
plt.figure(figsize=(10, 8))
plt.scatter(df_pca['PC1'], df_pca['PC2'])
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.title('PCA')
plt.hlines(0,-12,12, linestyles='dotted', colors='grey')
plt.vlines(0,-12,12, linestyles='dotted', colors='grey')
```

[13]:
```
<matplotlib.collections.LineCollection at 0x1a20acfba8>
```



```
[14]: component = pd.DataFrame(pca.components_,columns = df_wiki.columns).T
      component[0].sort_values(ascending = False)[:5]
```

```
[14]: bi2     0.230924
      bi1     0.226193
      use3    0.218809
      use4    0.214558
      pu3     0.210863
      Name: 0, dtype: float64
```

```
[15]: component[1].sort_values(ascending = False)[:5]
```

```
[15]: exp4                               0.228504
      use2                               0.218631
      use1                               0.197829
      vis3                               0.197628
      domain_Engineering_Architecture    0.171486
      Name: 1, dtype: float64
```

According to the above analysis, the top 5 variables that are strongly correlated with the first components are:

‘bi1’, ‘bi2’: behavior intentions to use and recommend wiki

‘use3’, ‘use4’: user behavior of recommending others to use wiki

‘pu3’: the usefulness for teaching.

The top 5 variables that are strongly correlated with the second component are:

‘exp4’: experience of contribute to wiki

‘use2’: develop teaching with wiki

‘use1’: develop educational activities

‘vis3’: cite wiki for academics

‘domain_engineering_architecture’: from the domain of engineer and architectures

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?

[16]:
```
print('variance explained by the first principal conponent:', pca.
 ↪explained_variance_ratio_[0])
print('variance explained by the second principal conponent:', pca.
 ↪explained_variance_ratio_[1])
print('variance explained by the first two principal conponent:', sum(pca.
 ↪explained_variance_ratio_))
```

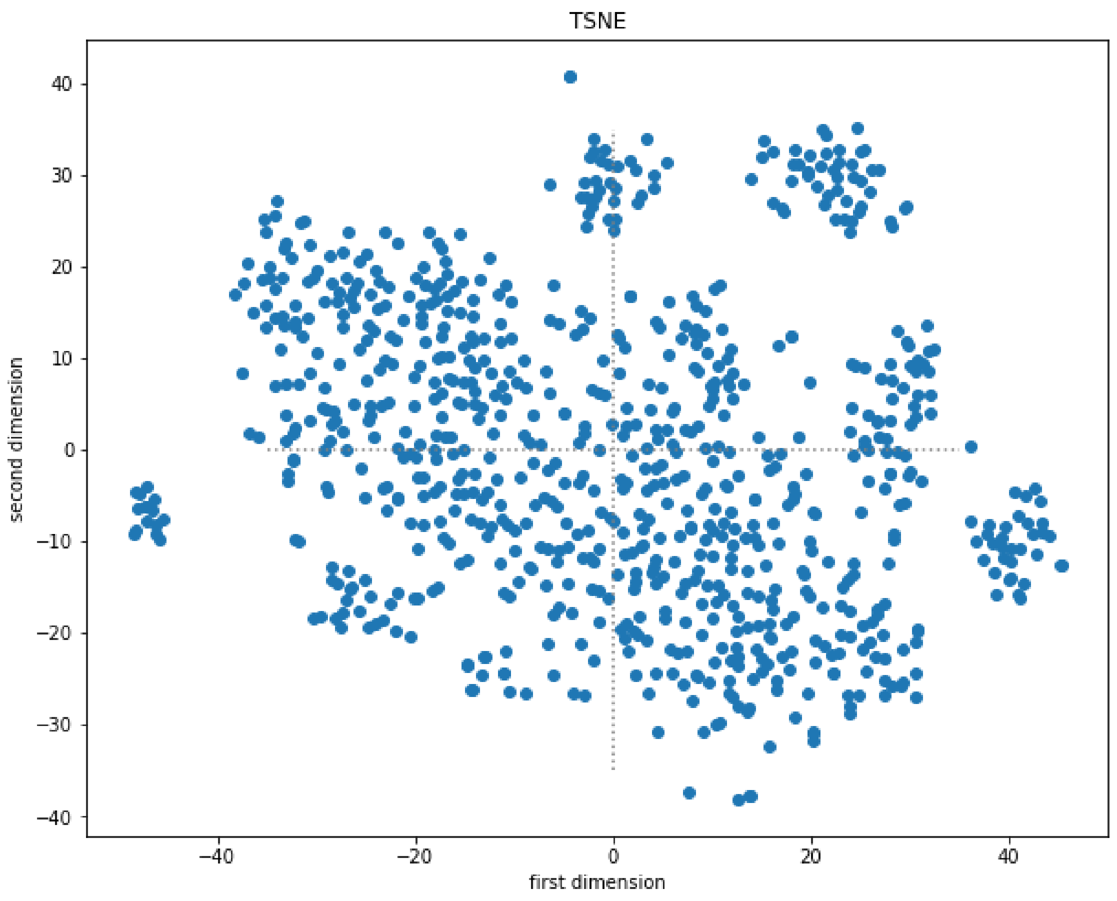
```
variance explained by the first principal conponent: 0.22810627785663395
variance explained by the second principal conponent: 0.06372474481018335
variance explained by the first two principal conponent: 0.2918310226668173
```

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

[17]:
```
tsne = TSNE(n_components=2,random_state=0,perplexity=20).fit_transform(x)
```

[18]:
```
plt.figure(figsize=(10,8))
plt.scatter(tsne[:,0],tsne[:,1])
plt.title('TSNE')
plt.xlabel('first dimension')
plt.ylabel('second dimension')
plt.hlines(0,-35,35, linestyles='dotted', colors='grey')
plt.vlines(0,-35,35, linestyles='dotted', colors='grey')
```

[18]:
```
<matplotlib.collections.LineCollection at 0x1a20be8c18>
```



From the above plot, t-SNE plot is composed of a large group with several smaller groups around. This plot shows one advantage of t-SNE over PCA: t-SNE deals better with the problem of overlapping and with non-linear data compared to PCA.

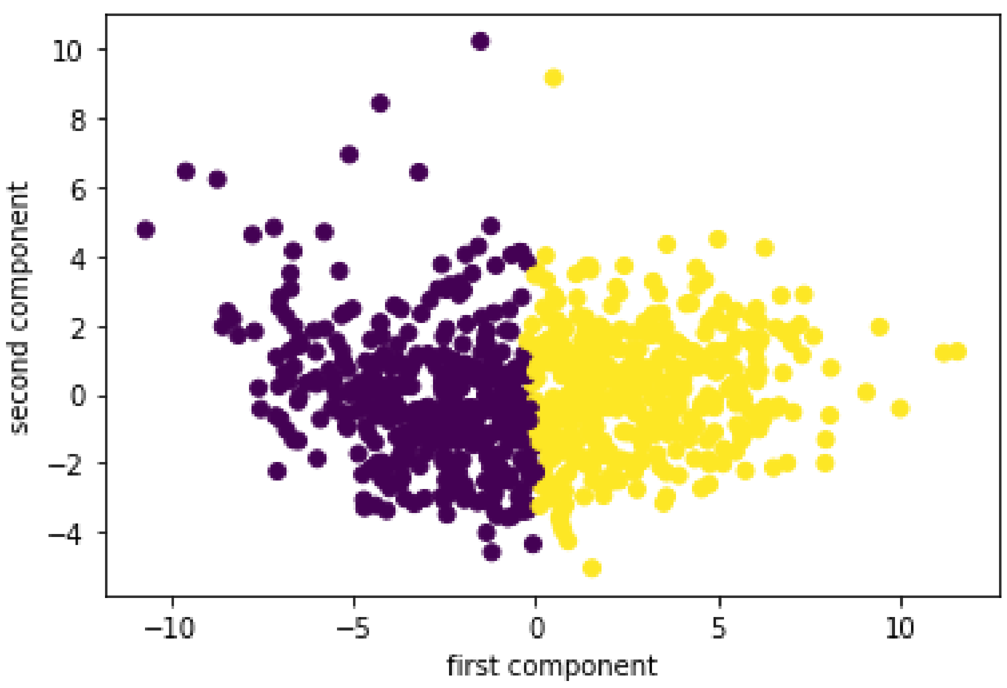
## 2.2 Clustering

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.

[19]:
```
kmeans2 = KMeans(n_clusters = 2, random_state = 0).fit_predict(x)
```

[20]:
```
plt.scatter(x_fit[:,0],x_fit[:,1],c = kmeans2)
plt.xlabel('first component')
plt.ylabel('second component')
```

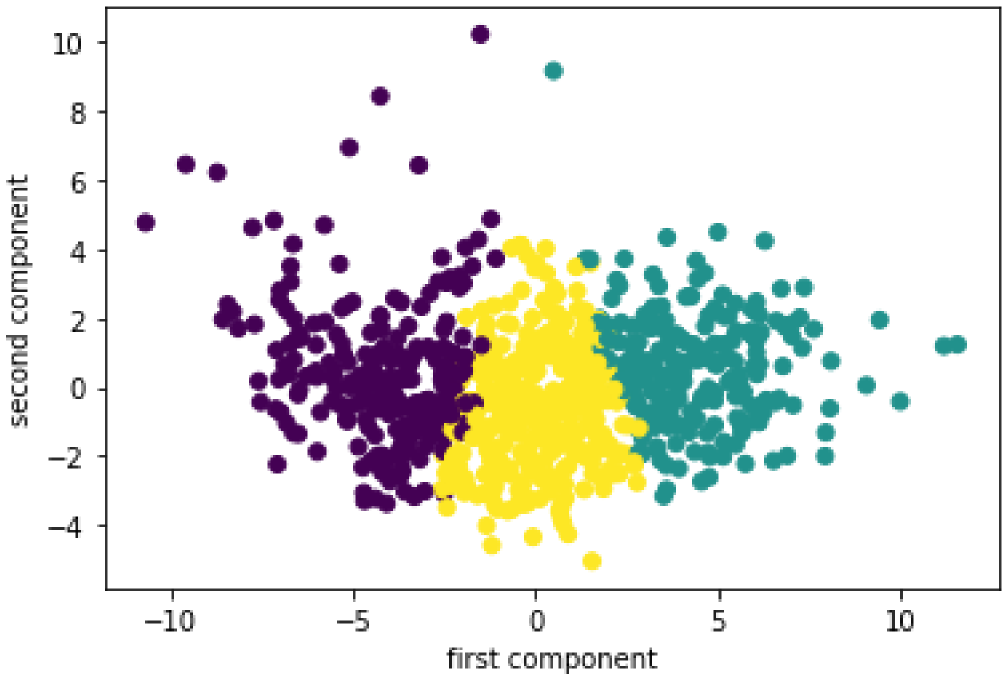
[20]:
```
Text(0, 0.5, 'second component')
```



```
[21]: kmeans3 = KMeans(n_clusters = 3, random_state = 0).fit_predict(x)
```

```
[22]: plt.scatter(x_fit[:,0],x_fit[:,1],c = kmeans3)
      plt.xlabel('first component')
      plt.ylabel('second component')
```

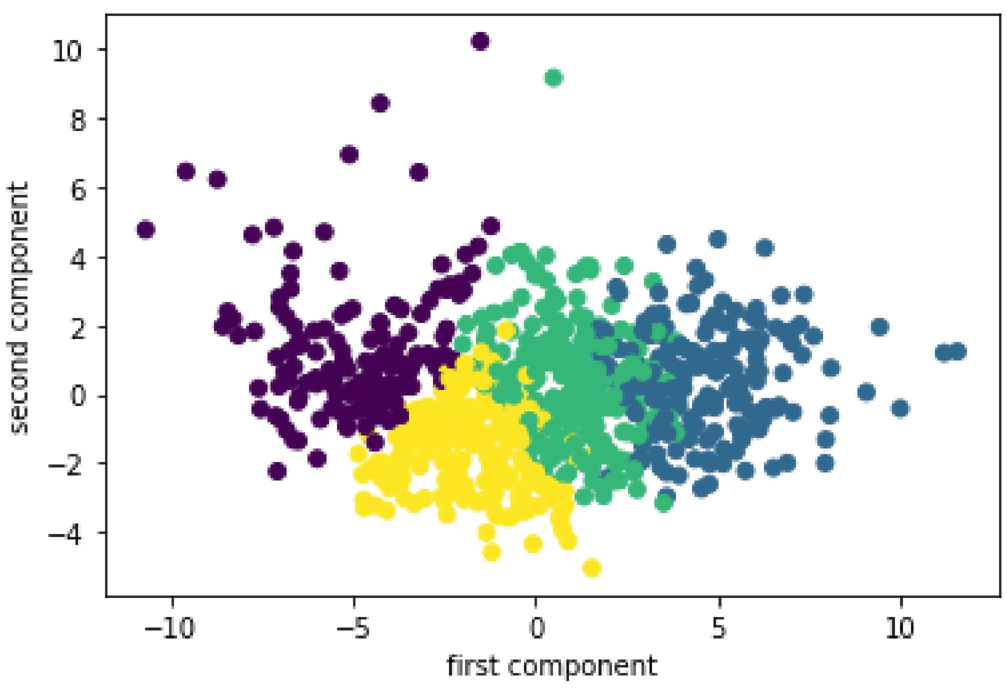
```
[22]: Text(0, 0.5, 'second component')
```



```
[23]: kmeans4 = KMeans(n_clusters=4, random_state = 0).fit_predict(x)
```

```
[24]: plt.scatter(x_fit[:,0],x_fit[:,1],c = kmeans4)
      plt.xlabel('first component')
      plt.ylabel('second component')
```

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



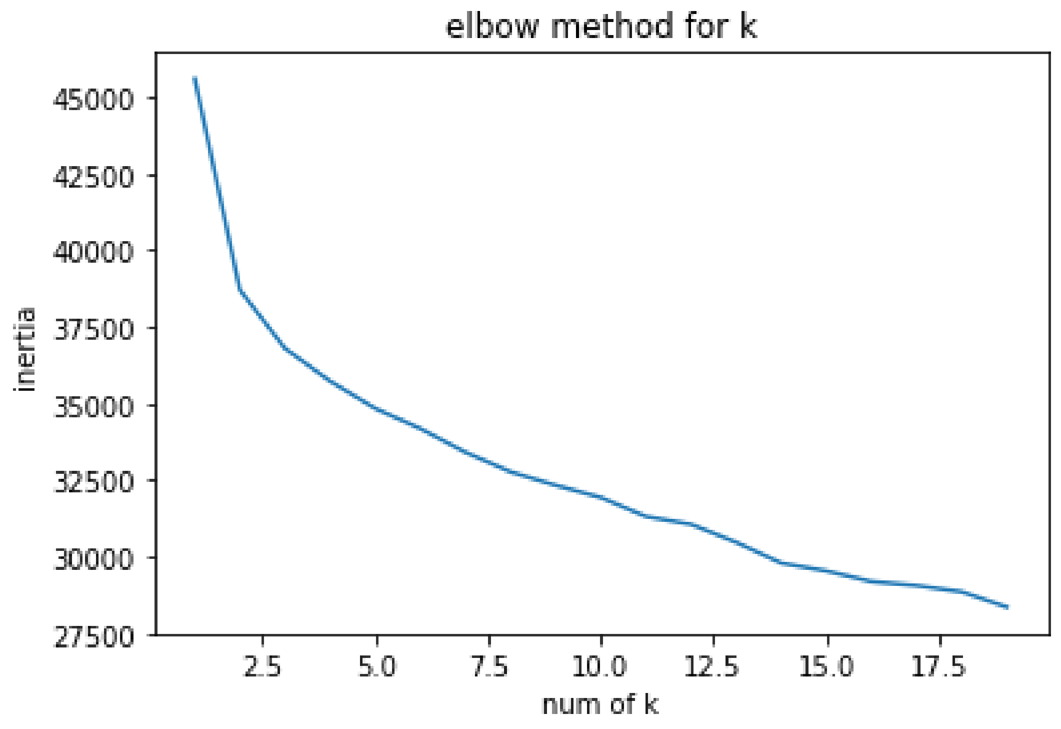
From the above plots, we can see that when k=2,3 method seperate the data well. When k=4, the clustered groups are very closed to one another, and in some regions they overlap suggesting that other factors besides PC1 and PC2 have an impact on our data.

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.

[25]:
```
inertia = []
for i in range(1,20):
    kmeans = KMeans(n_clusters=i, random_state = 0).fit(x)
    inertia.append(kmeans.inertia_)
```

[26]:
```
plt.plot(range(1,20), inertia)
plt.title('elbow method for k')
plt.xlabel('num of k')
plt.ylabel('inertia')
```

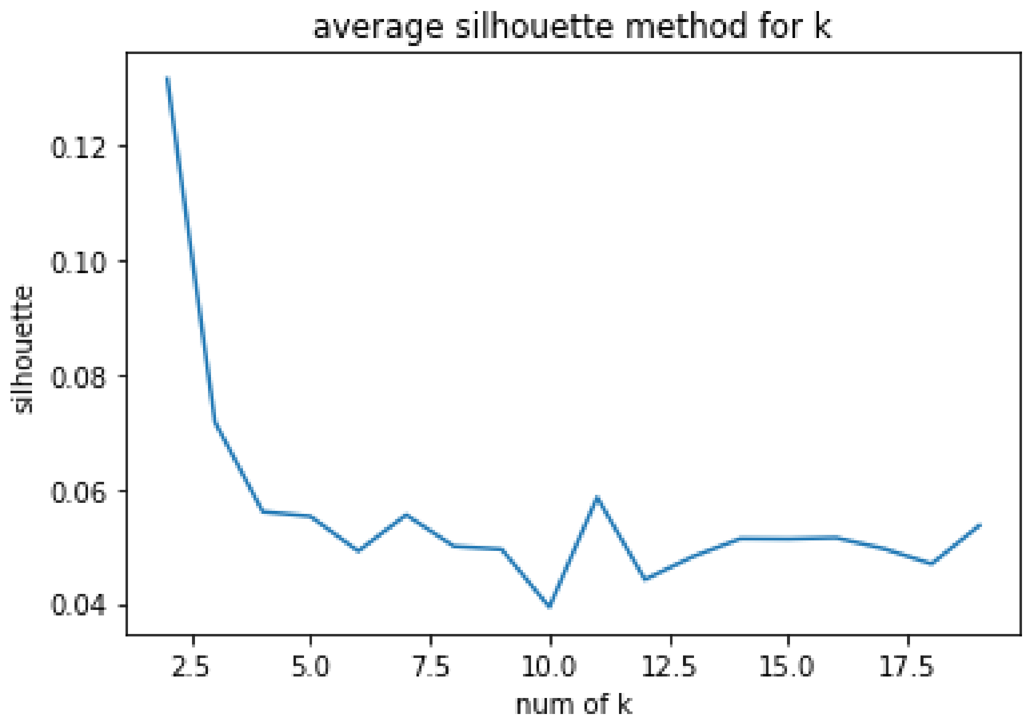
[26]:
```
Text(0, 0.5, 'inertia')
```



```
[27]: silhouette = []
      for i in range(2,20):
          kmeans = KMeans(n_clusters=i, random_state = 0).fit(x)
          silhouette.append(silhouette_score(x,kmeans.labels_))
```

```
[28]: plt.plot(range(2,20),silhouette)
      plt.title('average silhouette method for k')
      plt.xlabel('num of k')
      plt.ylabel('silhouette')
```

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



[29]:
```
opt = optimalK.OptimalK()
n_clusters = opt(x, cluster_array=np.arange(2, 20))
print('The optimal number of clusters is: ', n_clusters)
```

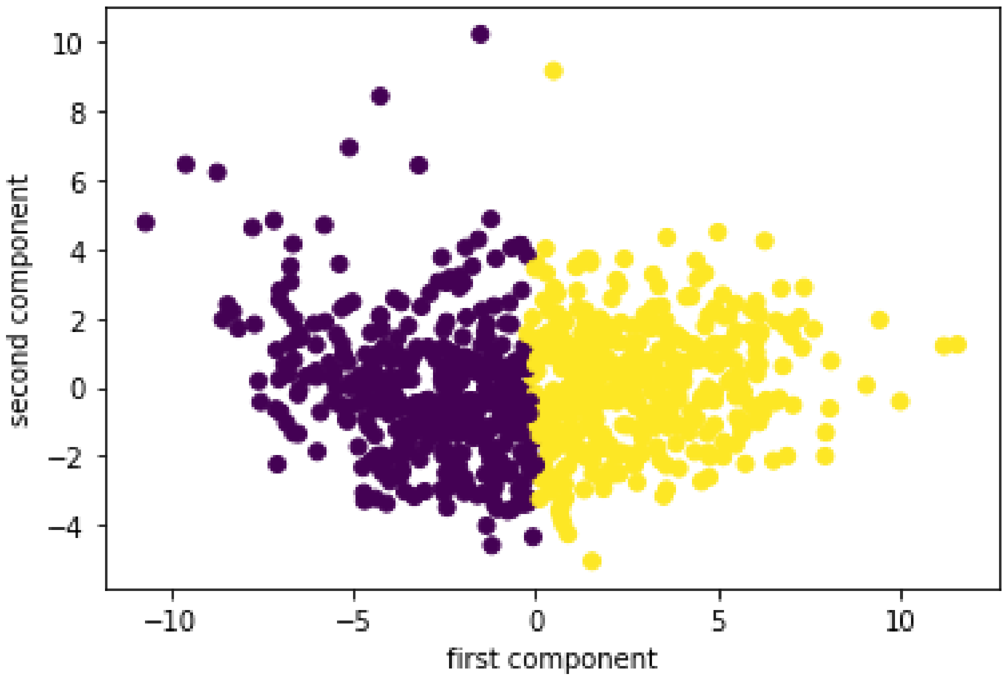
```
The optimal number of clusters is:  19
```

In gap statistics, the optimal number is 19. Both elbow method and silhouette score suggest that the optimal number of clusters is 2, and so we will choose k=2.

11.Visualize the results of the optimal ˆk-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?

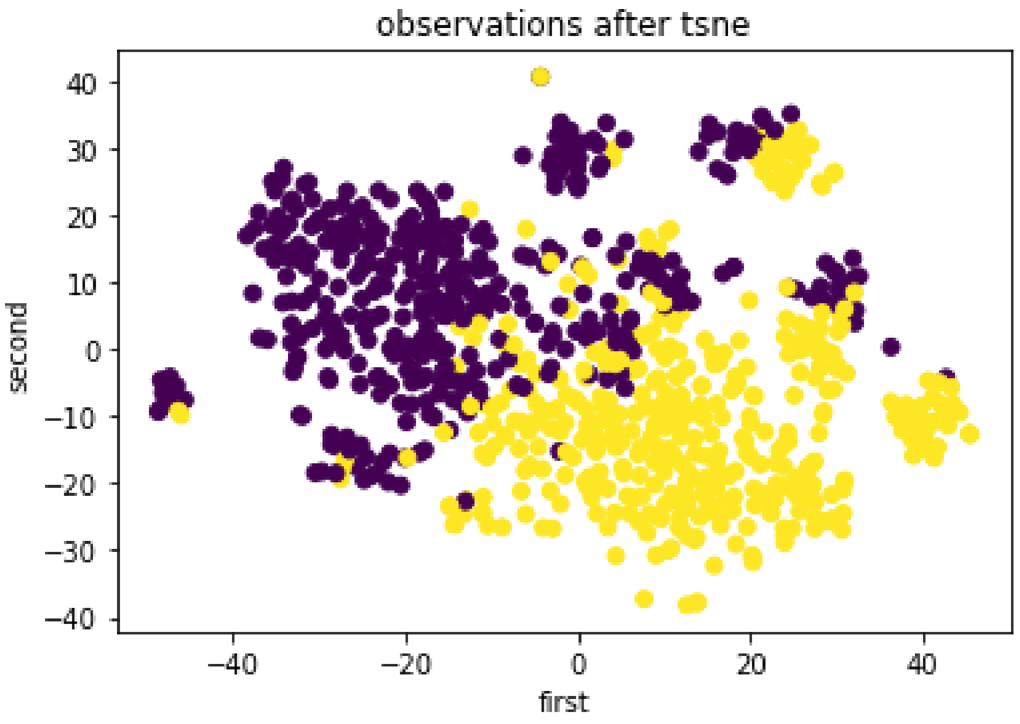
[30]:
```
plt.scatter(x_fit[:,0],x_fit[:,1],c = kmeans2)
plt.title('k-means clustering (k=2)')
plt.xlabel('first component')
plt.ylabel('second component')
```

[30]:
```
Text(0, 0.5, 'second component')
```



[31]:
```
plt.scatter(tsne[:,0],tsne[:,1],c=kmeans2)
plt.title('TSNE (dim=2)')
plt.xlabel('first dimension')
plt.ylabel('second dimension')
```

[31]:
```
Text(0, 0.5, 'second')
```



From the plots above, we can see that PCA seperates data well and there are merely overlapping between two clusters. While for the t-SNE, the boundary is not as clear as in the PCA. Also, PCA preserves large distances between points while t-SNE will preserve points which are close to each other.