Wang_Miaohan_HW7

March 14, 2020

[1]:
```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import random
from itertools import combinations
import seaborn as sns
import matplotlib.pyplot as plt
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_samples, silhouette_score
```

[2]:
```
seed = 7227
random.seed(seed)
```

k-Means Clustering "By Hand"

[3]:
```
df = pd.DataFrame()
df['input_1'] = [5,8,7,8,3,4,2,3,4,5]
df['input_2'] = [8,6,5,4,3,2,2,8,9,8]
```

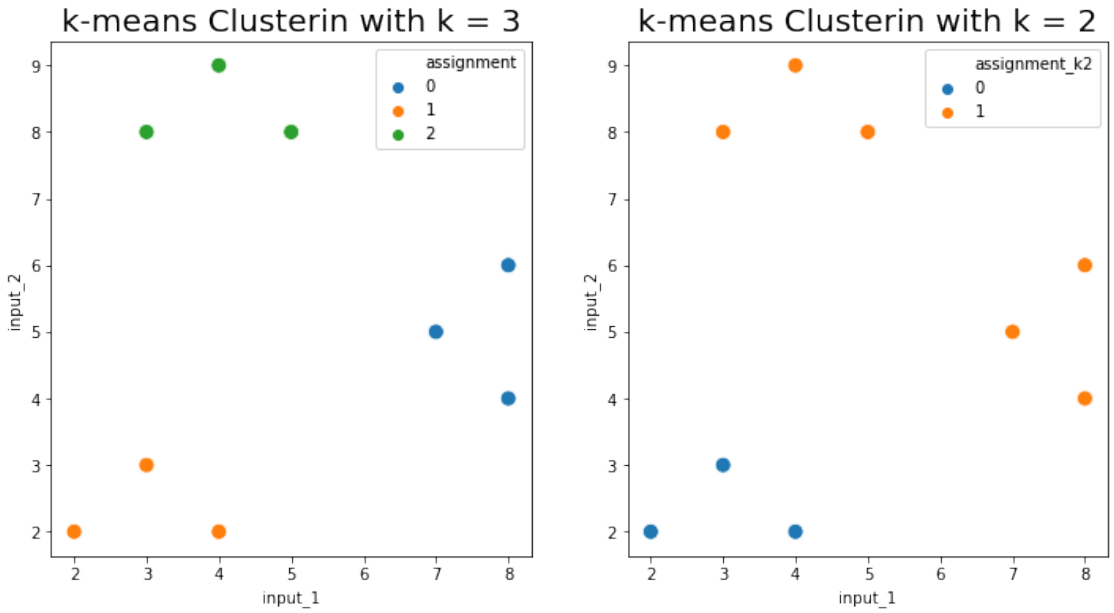
[4]:
```
def Euclidean(x1, x2, y1, y2):
    return np.sqrt((y1-x1)**2 + (y2-x2)**2)

def find_assignment(x1, x2, centroids):
    min_distance = 10000
    best_centroid = None

    for i in range(centroids.shape[0]):
        distance = Euclidean(x1, x2, centroids.iloc[i][0], centroids.iloc[i][1])
        if distance < min_distance:
            min_distance = distance
```

```
            best_centroid = i

    return best_centroid
```

[5]:
```
# k = 3
df['assignment'] = [np.random.randint(0,3) for i in range(10)]
df['new_assignment'] = [np.random.randint(0,3) for i in range(10)]

# Iteration for the best centroids
while not df.assignment.equals(df.new_assignment):
    df['assignment'] = df['new_assignment']
    df['new_assignment'] = df.apply(lambda row: find_assignment(row['input_1'],␣
 ↪row['input_2'],
                                                                df.
 ↪groupby('assignment').mean()) ,axis=1)
```

[6]:
```
# k = 2
df['assignment_k2'] = [np.random.randint(0,2) for i in range(10)]
df['new_assignment_k2'] = [np.random.randint(0,2) for i in range(10)]

# Iteration for the best centroids
while not df.assignment_k2.equals(df.new_assignment_k2):
    df['assignment_k2'] = df['new_assignment_k2']
    df['new_assignment_k2'] = df.apply(lambda row:␣
 ↪find_assignment(row['input_1'], row['input_2'],
                                                                df.
 ↪groupby('assignment_k2').mean()) ,axis=1)
```

[7]:
```
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(12,6))

ax1.set_title('k-means Clusterin with k = 3', fontsize=20)
sns.scatterplot(x='input_1', y='input_2', hue='assignment', data=df,
                      palette=sns.color_palette(n_colors=3), s=100, ax=ax1)

ax2.set_title('k-means Clusterin with k = 2', fontsize=20)
sns.scatterplot(x='input_1', y='input_2', hue='assignment_k2', data=df,
                      palette=sns.color_palette(n_colors=2), s=100, ax=ax2);
```



In this case, 3 clusters turn out two be a better fit than 2 clusters. With $k = 3$, points in each cluster share similar distance to each other. However, in $k = 2$ case, the span of the blue cluster is too large. There obviously exists two blobs of points within the blue cluster.

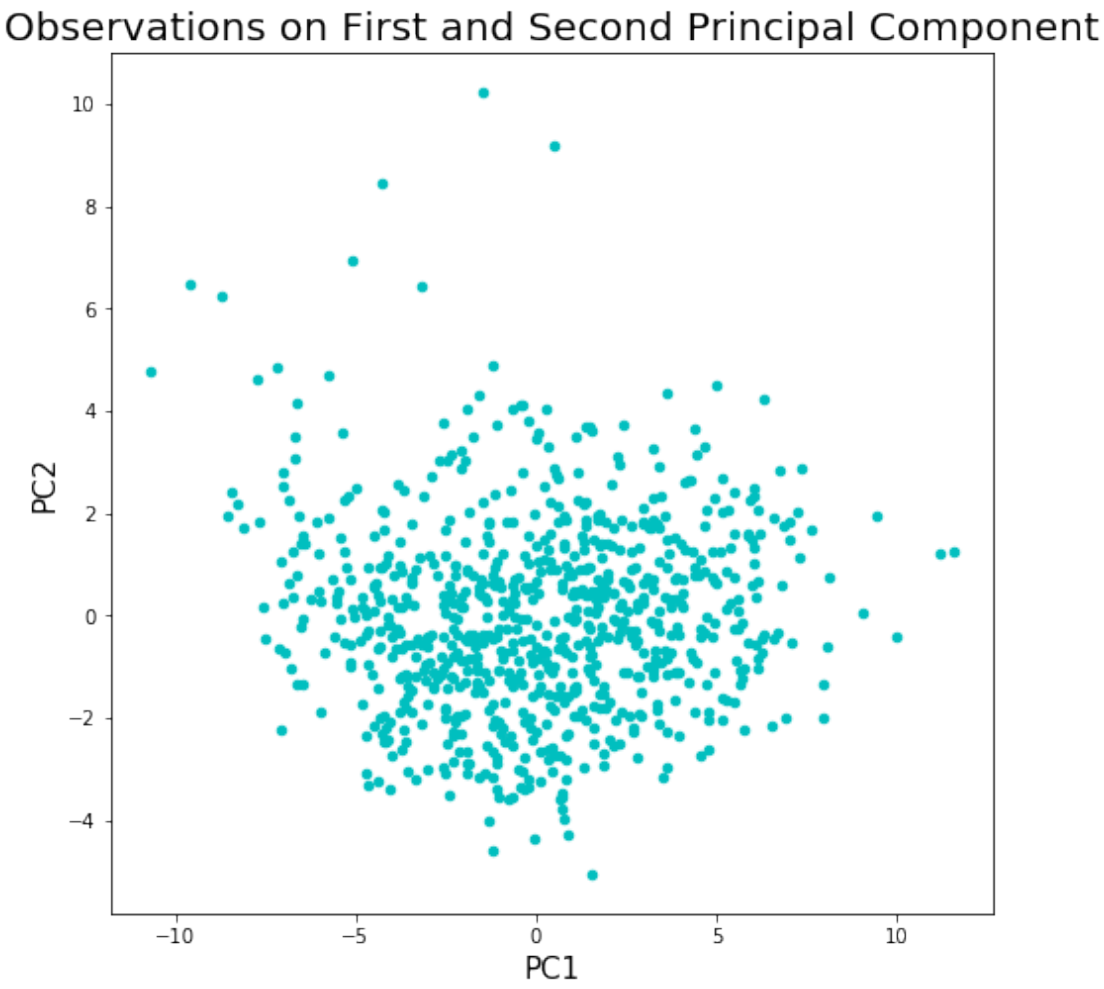
Application

[8]:
```
wiki_df = pd.read_csv('data/wiki.csv')
```

Dimension reduction

[9]:
```
X = StandardScaler().fit_transform(wiki_df)
pca = PCA(random_state=seed)
X_pc = pca.fit_transform(X)
```

[10]:
```
plt.figure(figsize=(8,8))
plt.title('Observations on First and Second Principal Component', fontsize=20)
plt.scatter(X_pc[:,0], X_pc[:,1], s=20, c='c')
plt.xlabel('PC1', fontsize=15)
plt.ylabel('PC2', fontsize=15);
```



[11]:
```
# obtaining loading vectors for all PCs
components = pd.DataFrame(pca.components_, columns=wiki_df.columns).T

# top loading vector entries of the first component
components[0].sort_values(ascending=False).head()
```

[11]:
```
bi2     0.230924
bi1     0.226193
use3    0.218809
use4    0.214558
pu3     0.210863
Name: 0, dtype: float64
```

Behavior intentions to use and reccommend Wikipedia (bi1, bi2) seem to be most correlated with the first component. Then, the user behavior of recommending students and colleagues to use Wikipedia (use3, use4) follows in the level of correlation. The fifth most correlated variable is whether one perceives Wikipedia to be useful for teaching (pu3). All top-correlated variables in

the first component seems to be about usefulness of wikipedia.

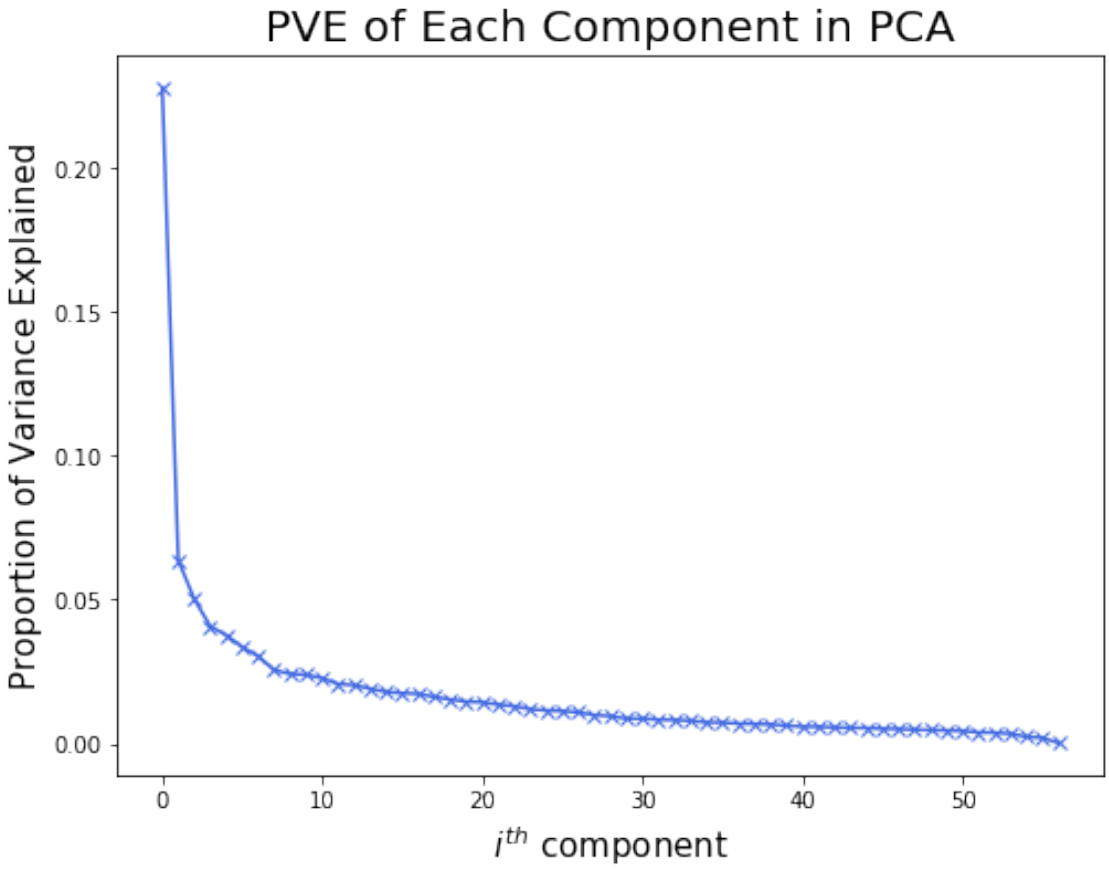
[12]:
```
# loading vector of the second component
components[1].sort_values(ascending=False).head()
```

[12]:
```
exp4                                0.228494
use2                                0.218629
use1                                0.197827
vis3                                0.197635
domain_Engineering_Architecture     0.171484
Name: 1, dtype: float64
```

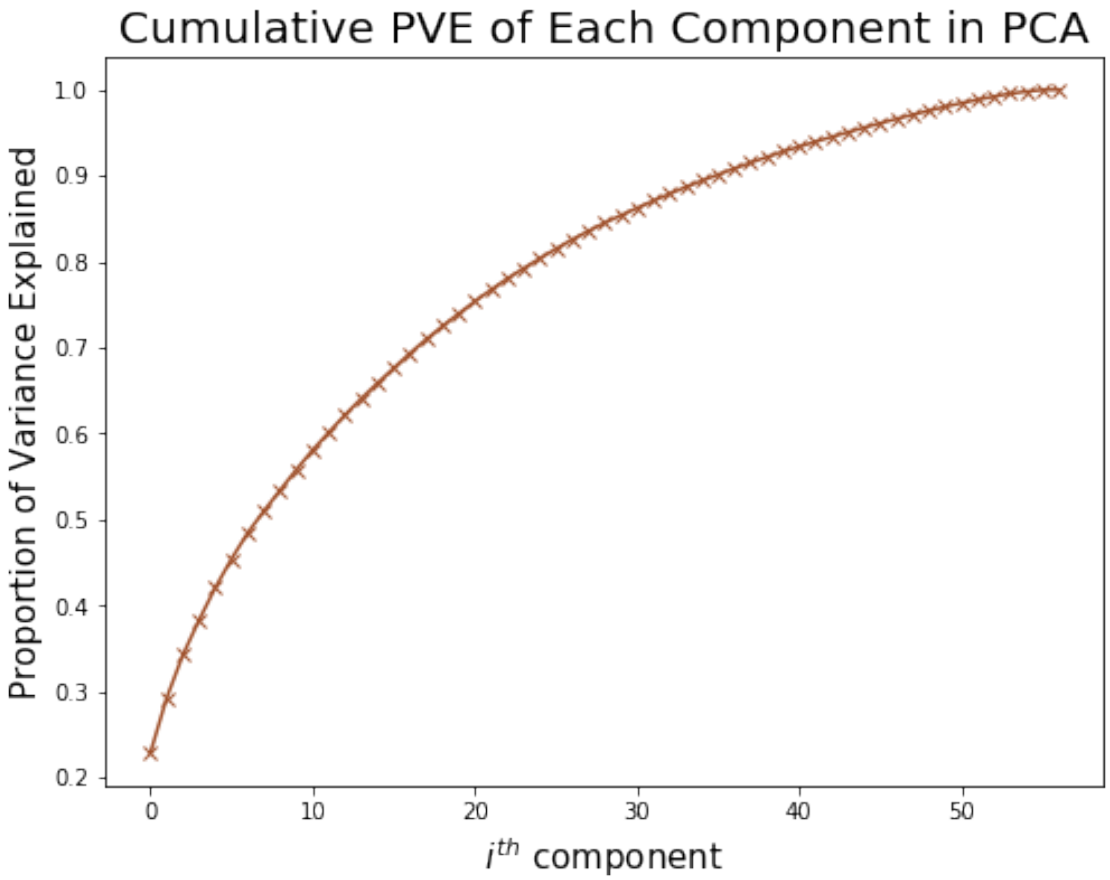
The most correlated variable in the second component is whether the person has the experience of contributing to Wikipedia (exp4). Then the variables of whether the person has the user behavior to develop his/her teaching materials with Wikipedia and whether he/she uses Wikipedia as a platform to develop educational activities with students (use2, use1). The fourth correlated variable here is whether the person cite wikipedia in his/her academic papers. The fifth correlated variable is whether the person come from the field of enginerring and architecture. The second component seems to be about how the person uses Wikipedia.

[13]:
```
plt.figure(figsize=(8,6))
plt.plot(np.arange(57), pca.explained_variance_ratio_, marker='x',␣
 ↪color='royalblue')
plt.title('PVE of Each Component in PCA', fontsize=20)
plt.xlabel('$i^{th}$ component', fontsize=15)
plt.ylabel('Proportion of Variance Explained', fontsize=15);
print(f'The first and second component explained about {round(pca.
 ↪explained_variance_ratio_[:2].sum(), 4)*100}% of variance')
```

```
The first and second component explained about 29.18% of variance
```

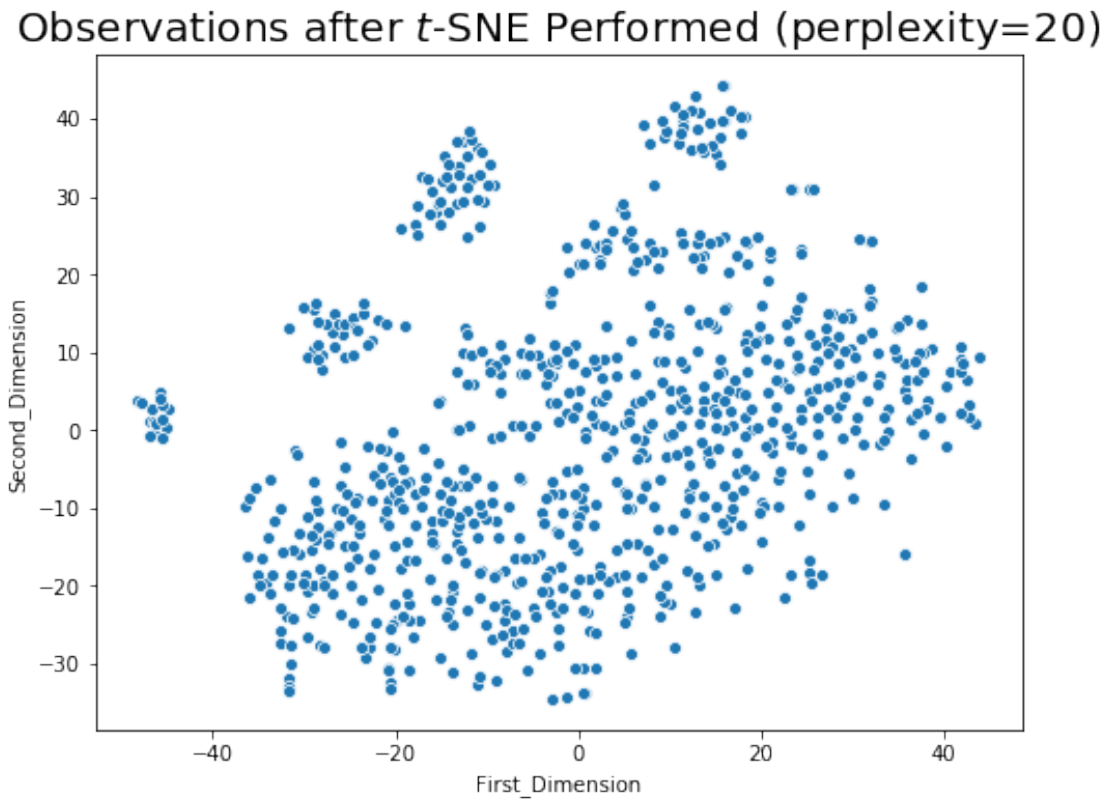


[14]:
```
cumulative_variance_explained = np.cumsum(pca.explained_variance_ratio_)

plt.figure(figsize=(8,6))
plt.plot(np.arange(57), cumulative_variance_explained, marker='x',␣
↪color='sienna')
plt.title('Cumulative PVE of Each Component in PCA', fontsize=20)
plt.xlabel('$i^{th}$ component', fontsize=15)
plt.ylabel('Proportion of Variance Explained', fontsize=15);
```



[15]:
```
# t-SNE
tsne = TSNE(n_components=2, perplexity=20, random_state=seed)
X_embedded = tsne.fit_transform(X)
```

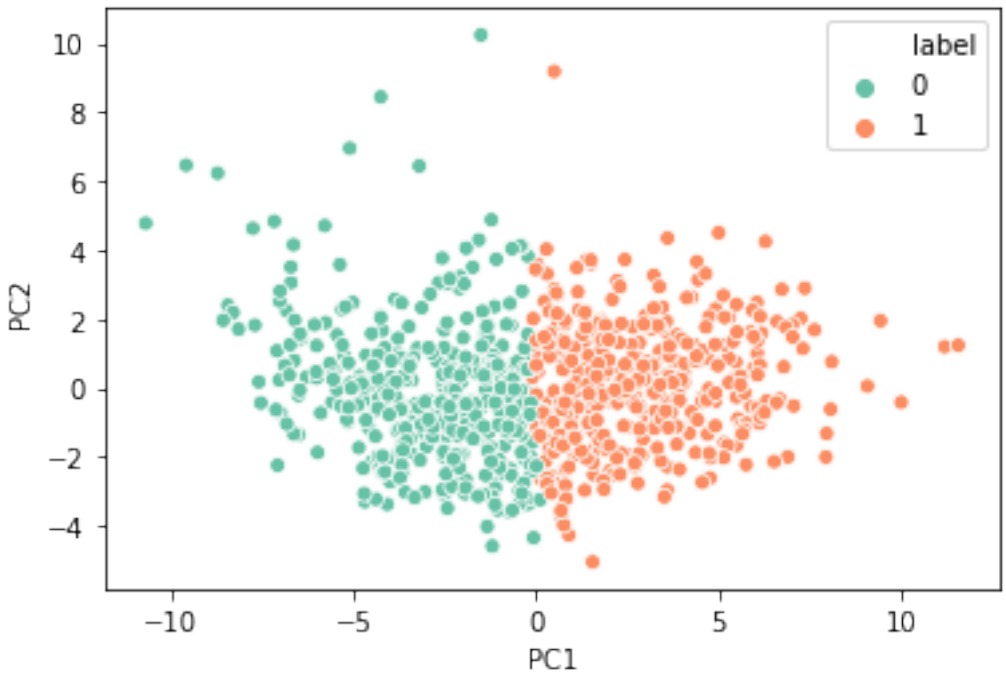
[16]:
```
tsne_df = pd.DataFrame(X_embedded, columns=['First_Dimension',␣
↪'Second_Dimension'])

plt.figure(figsize=(8,6))
plt.title('Observations after $t$-SNE Performed (perplexity=20)', fontsize=20)
sns.scatterplot(x='First_Dimension', y='Second_Dimension', data=tsne_df);
```



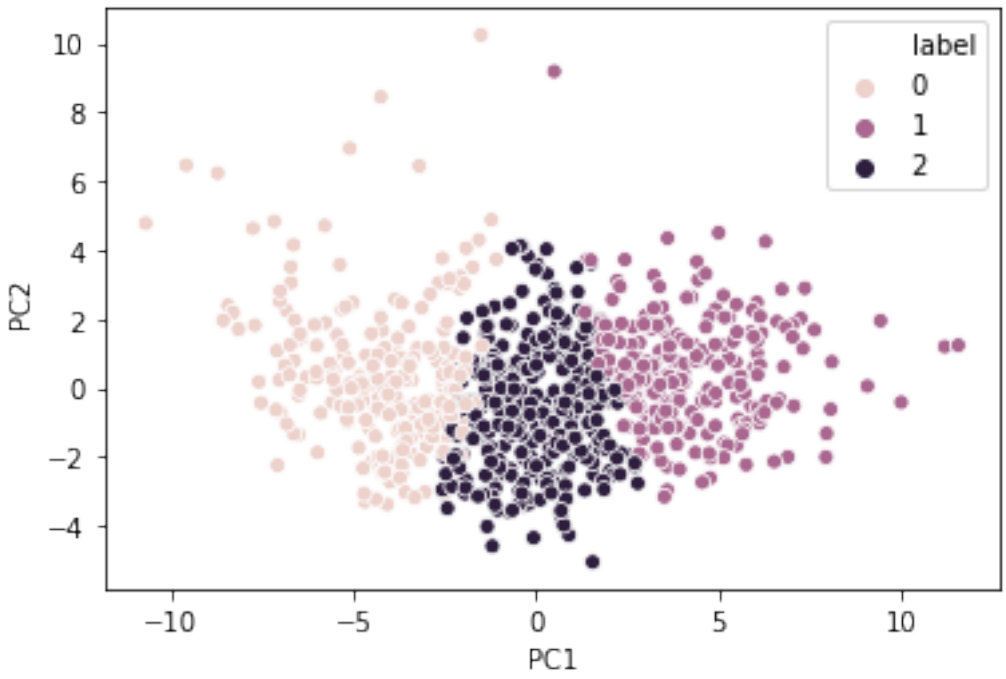
t-SNE yielded some small clusters on the upper left but did not do a good job separating the lower right blob of data. Boundaries between the lower blob are unclear, hence t-SNE might not be a good method for evaluating this dataset.

Clustering

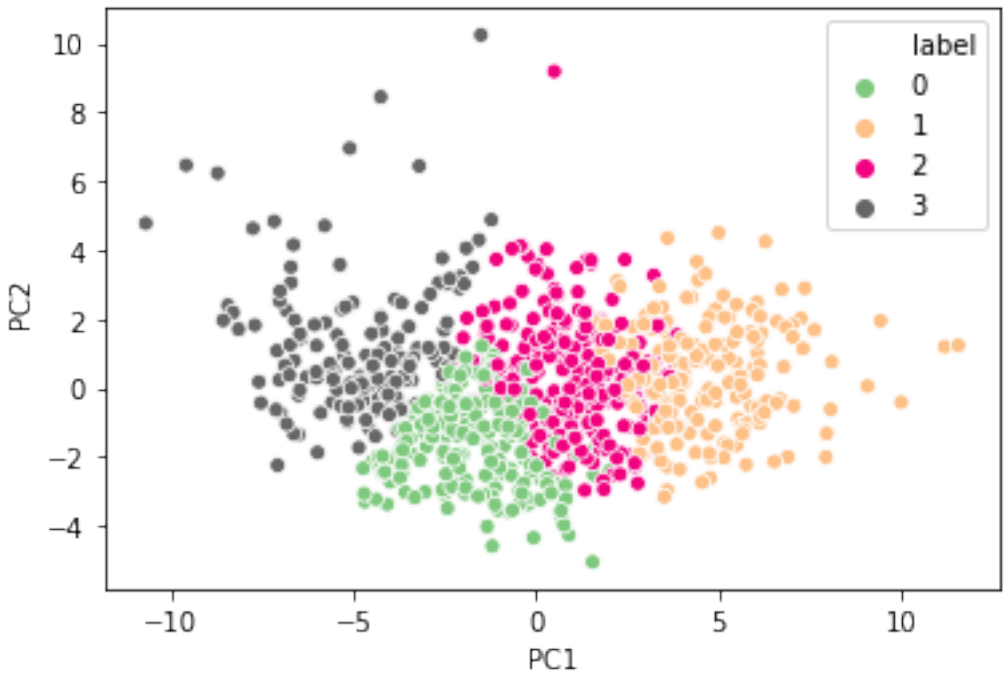
[17]:
```
# k = 2
k2 = KMeans(n_clusters=2, random_state=seed)

k2_df = pd.DataFrame(X_pc[:,0:2], columns=['PC1', 'PC2'])
k2_df['label'] = k2.fit_predict(X)

sns.scatterplot(x='PC1', y='PC2', hue='label', data=k2_df, palette='Set2');
```



[18]:
```
# k = 3
k3 = KMeans(n_clusters=3, random_state=seed)

k3_df = pd.DataFrame(X_pc[:,0:2], columns=['PC1', 'PC2'])
k3_df['label'] = k3.fit_predict(X)

sns.scatterplot(x='PC1', y='PC2', hue='label', data=k3_df);
```



[19]:
```
# k = 4
k4 = KMeans(n_clusters=4, random_state=seed)

k4_df = pd.DataFrame(X_pc[:,0:2], columns=['PC1', 'PC2'])
k4_df['label'] = k4.fit_predict(X)

sns.scatterplot(x='PC1', y='PC2', hue='label', data=k4_df, palette='Accent');
```

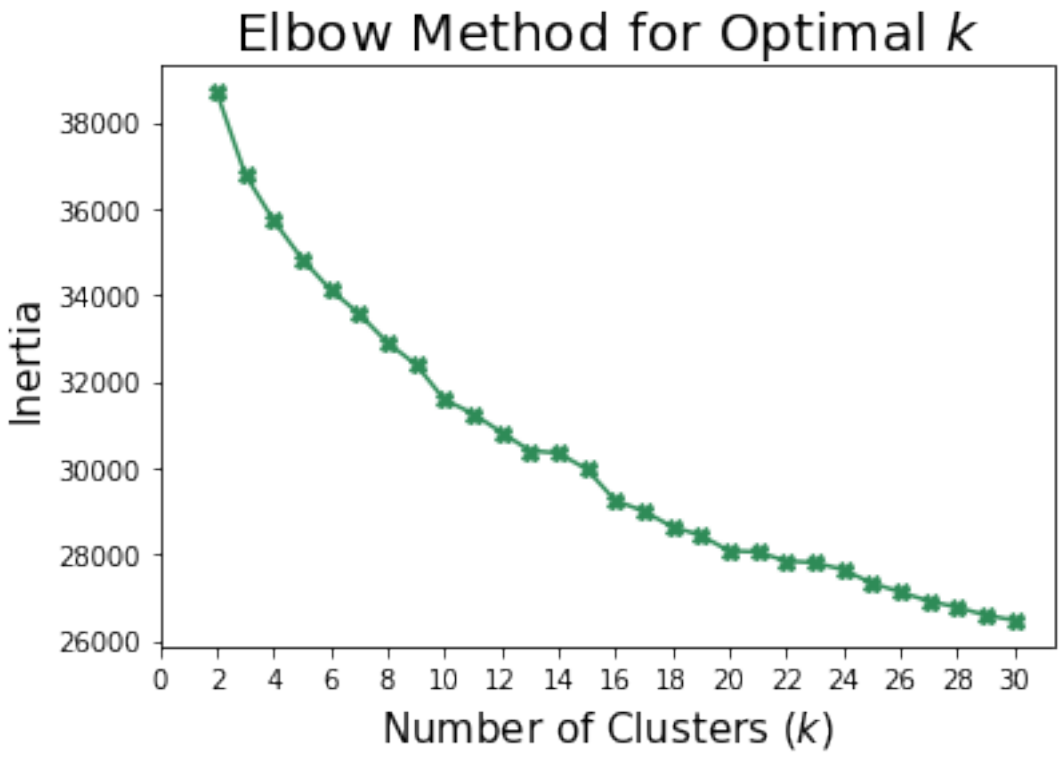


Through k-means clustering with $k = 2, 3, 4$, we observethat the clusters are mostly separated on the value of the first principal component. The second principal component does not contribute much to the splitting of the clusters.

Elbow Method & Average Silhouette

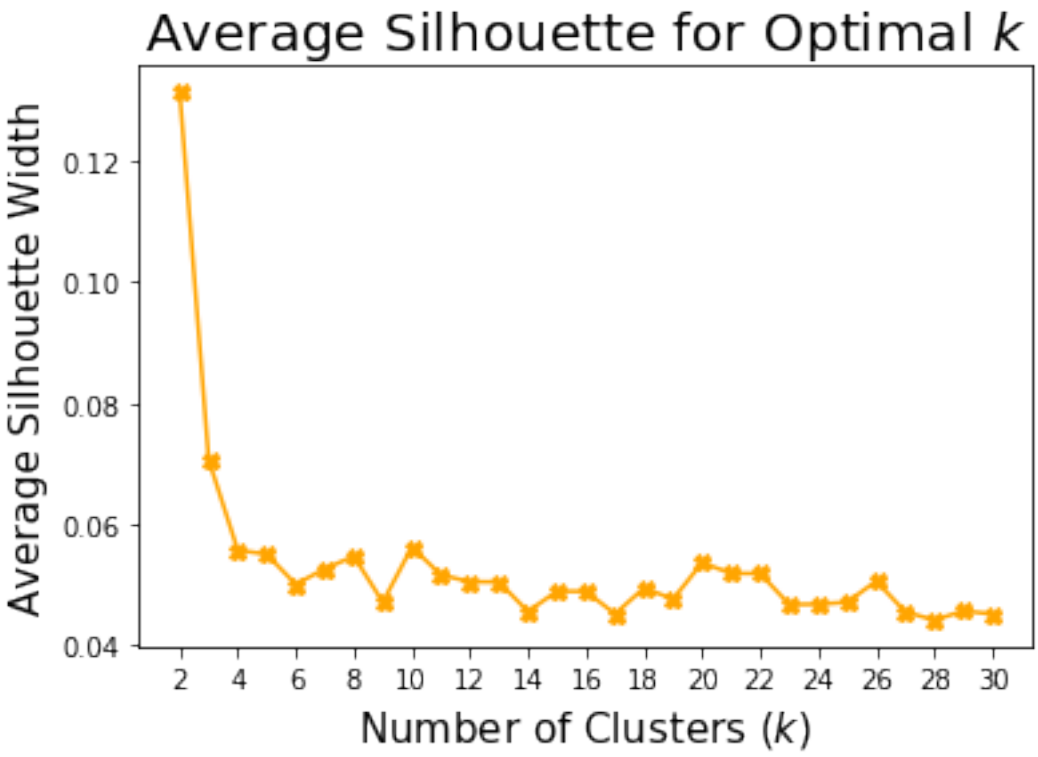
[20]:
```
inertia_score = []
avg_sil_score =[]

test_range = range(2,31)

for i in test_range:
    mod = KMeans(n_clusters=i, random_state=seed).fit(X)
    inertia_score.append(mod.inertia_)
    avg_sil_score.append(silhouette_score(X, mod.labels_))
```

[21]:
```
plt.plot(test_range, inertia_score, marker='X', color='seagreen')
plt.title('Elbow Method for Optimal $k$', fontsize=20)
plt.xlabel('Number of Clusters ($k$)', fontsize=15)
plt.xticks(np.arange(0,max(test_range)+2,2))
plt.ylabel('Inertia', fontsize=15);
```



The above graph does not show a clear "elbow," therefore the optimal number of clusters need to be further determined by other methods. For investigation purposes, we will say that the optimal k is 13, where a tiny "elbow" is reached.

[22]:
```
plt.plot(test_range, avg_sil_score, marker='X', color='orange')
plt.title('Average Silhouette for Optimal $k$', fontsize=20)
plt.xlabel('Number of Clusters ($k$)', fontsize=15)
plt.xticks(np.arange(2,max(test_range)+2,2))
plt.ylabel('Average Silhouette Width', fontsize=15);
```



According to the average silhouette graph, the optimal number of clusters is 2, since the average silhouette width peaks at $k = 2$.

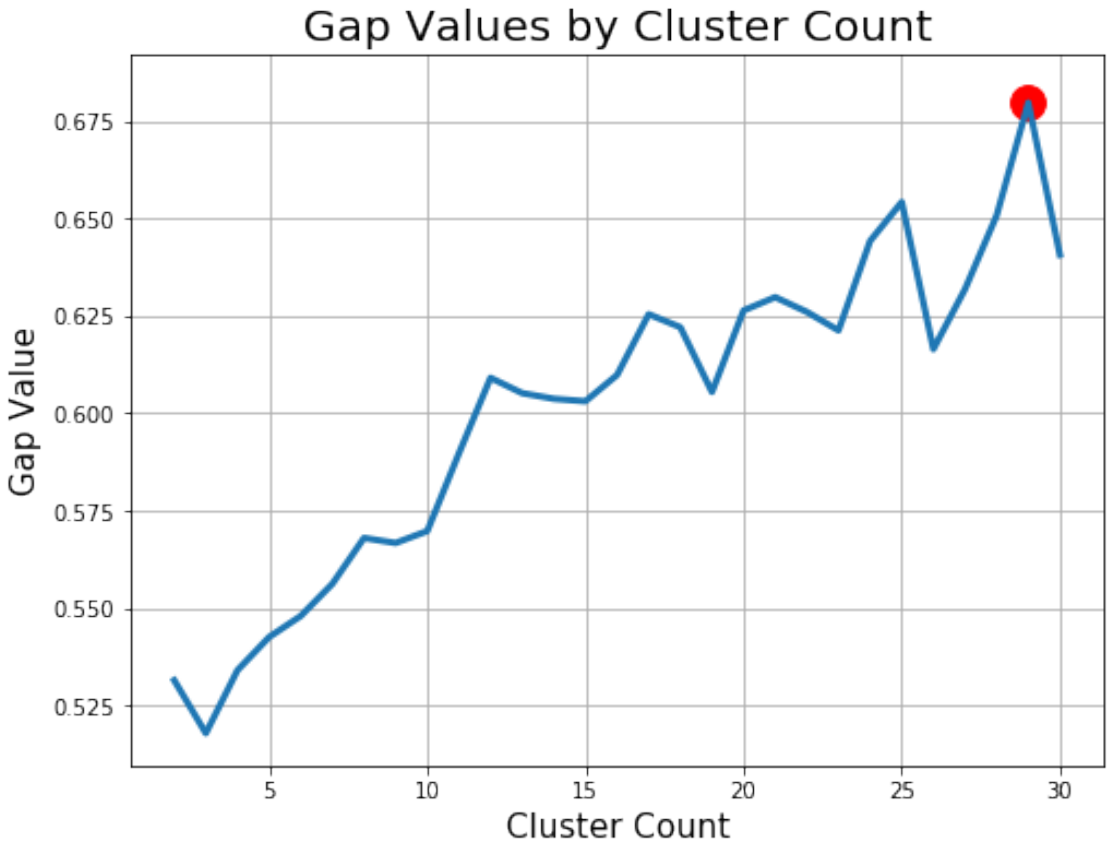
Gap Statistics

[23]:
```
from gap_statistic import OptimalK

optimalK = OptimalK()
n_clusters = optimalK(X, cluster_array=np.arange(2, 31))
print(f'With gap statistics, the optimal number of clusters is {n_clusters}.')
```

```
With gap statistics, the optimal number of clusters is 29.
```

[24]:
```
plt.figure(figsize=(8,6))
plt.plot(optimalK.gap_df.n_clusters, optimalK.gap_df.gap_value, linewidth=3)
plt.scatter(optimalK.gap_df[optimalK.gap_df.n_clusters == n_clusters].
 ↪n_clusters,
            optimalK.gap_df[optimalK.gap_df.n_clusters == n_clusters].
 ↪gap_value, s=250, c='r')
plt.grid(True)
plt.xlabel('Cluster Count', fontsize=15)
```

```
plt.ylabel('Gap Value', fontsize=15)
plt.title('Gap Values by Cluster Count', fontsize=20)
plt.show()
```

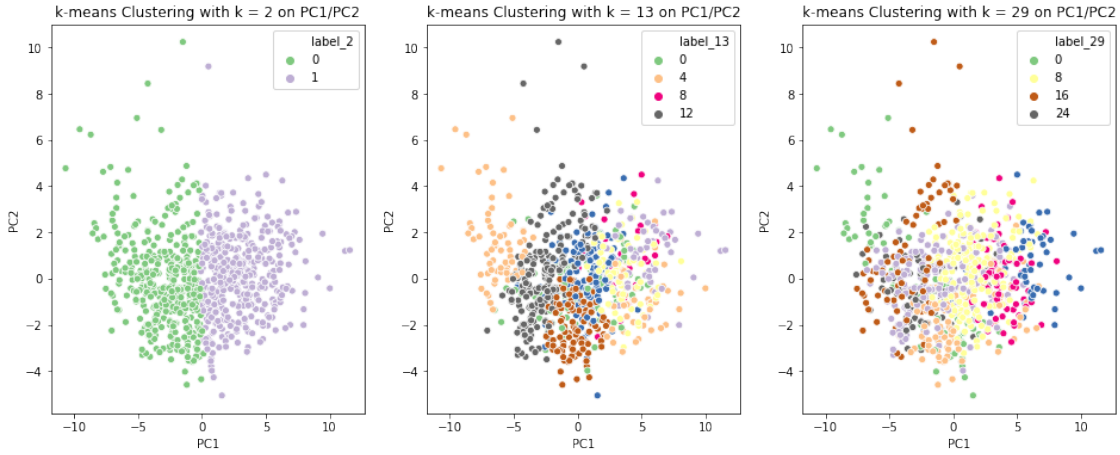


Each of three optimal k evaluation method yields different answers, so the optimal k for this dataset is unclear.

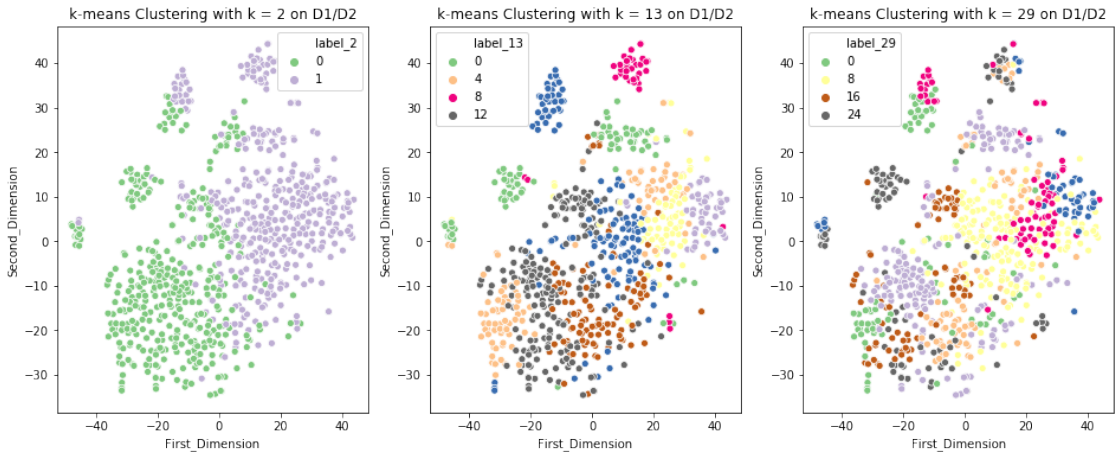
We are plot out the three answers from above:

[28]:
```
mod_2 = KMeans(n_clusters=2, random_state=seed)
mod_13 = KMeans(n_clusters=13, random_state=seed)
mod_29 = KMeans(n_clusters=22, random_state=seed)

best_df = pd.DataFrame(X_pc[:,0:2], columns=['PC1', 'PC2'])
best_df['label_2'] = mod_2.fit_predict(X)
best_df['label_13'] = mod_13.fit_predict(X)
best_df['label_29'] = mod_22.fit_predict(X)
tsne_df['label_2'] = mod_2.fit_predict(X)
tsne_df['label_13'] = mod_13.fit_predict(X)
tsne_df['label_29'] = mod_22.fit_predict(X)
```

```
# Clustering using
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(16,6))

ax1.set_title('k-means Clustering with k = 2 on PC1/PC2')
sns.scatterplot(x='PC1', y='PC2', hue='label_2', data=best_df,␣
 ↪palette='Accent', ax=ax1)

ax2.set_title('k-means Clustering with k = 13 on PC1/PC2')
sns.scatterplot(x='PC1', y='PC2', hue='label_13', data=best_df,␣
 ↪palette='Accent', ax=ax2)

ax3.set_title('k-means Clustering with k = 29 on PC1/PC2')
sns.scatterplot(x='PC1', y='PC2', hue='label_29', data=best_df,␣
 ↪palette='Accent', ax=ax3);
```

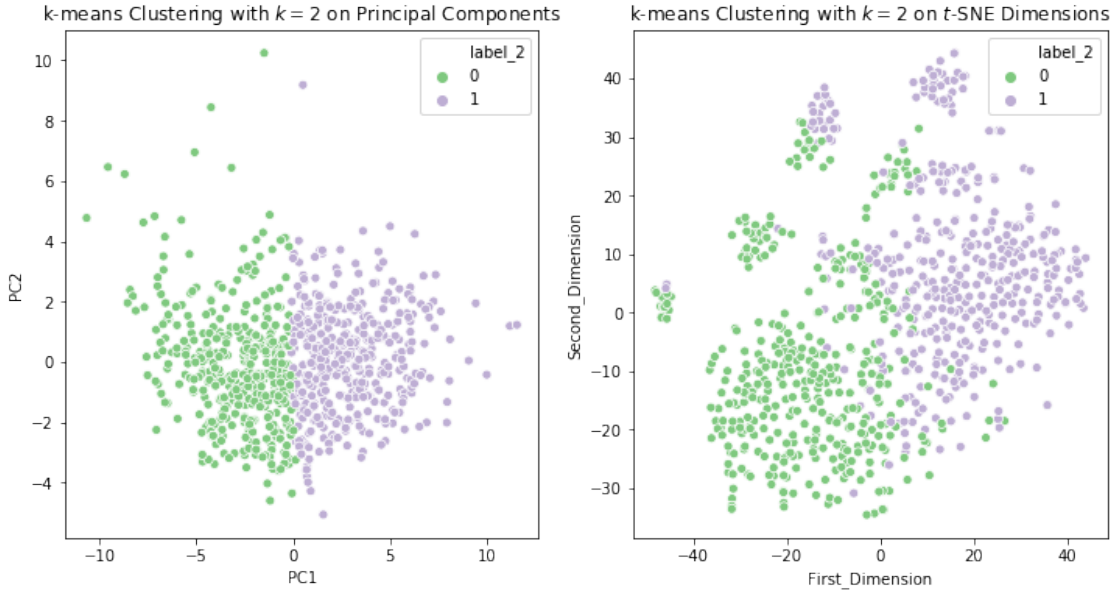


[29]:
```
# Clustering using the first and second dimensions from t-SNE
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(16,6))

ax1.set_title('k-means Clustering with k = 2 on D1/D2')
sns.scatterplot(x='First_Dimension', y='Second_Dimension', hue='label_2',␣
 ↪data=tsne_df, palette='Accent', ax=ax1)

ax2.set_title('k-means Clustering with k = 13 on D1/D2')
sns.scatterplot(x='First_Dimension', y='Second_Dimension', hue='label_13',␣
 ↪data=tsne_df, palette='Accent', ax=ax2)

ax3.set_title('k-means Clustering with k = 29 on D1/D2')
sns.scatterplot(x='First_Dimension', y='Second_Dimension', hue='label_29',␣
 ↪data=tsne_df, palette='Accent', ax=ax3);
```



Visually, $k = 2$ does the best work in clustering the dataset in both principal components graph and t-SNE dimensions graph. Therefore we will say that $k = 2$ is the optimal k for clustering in this scenario.

[27]:
```
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(12,6))

ax1.set_title('k-means Clustering with $k = 2$ on Principal Components')
sns.scatterplot(x='PC1', y='PC2', hue='label_2', data=best_df,␣
↪palette='Accent', ax=ax1)

ax2.set_title('k-means Clustering with $k = 2$ on $t$-SNE Dimensions')
sns.scatterplot(x='First_Dimension', y='Second_Dimension', hue='label_2',␣
↪data=tsne_df, palette='Accent', ax=ax2);
```



With principal components as axis, the data is mostly separated on the value of the first principal components, a vertical boundary. When the data plotted onto the first and second dimension of t-SNE, the data is separated by an oblique boundary, not horizontal or vertical, meaning that the boundary takes a linear functional form that requires both the value of first and second dimension. So in t-SNE, the first and second dimension is almost equally important in separating the two clusters, but in principal components, it's mostly the first principal components working.

[ ]: