Untitled

May 18, 2021

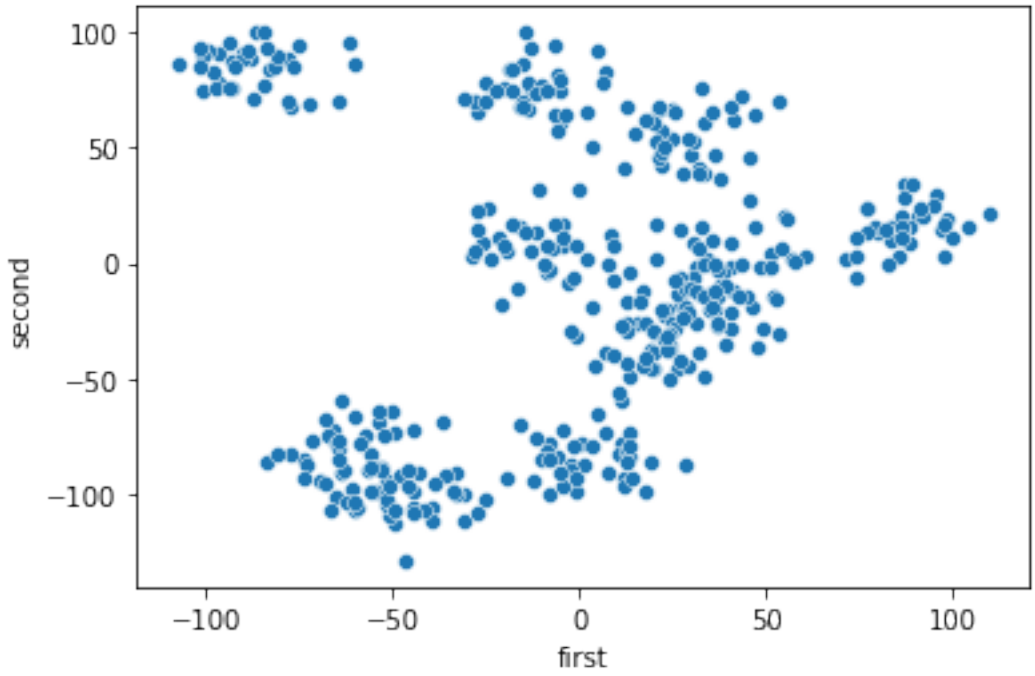
[1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
```

[2]:
```
data = pd.read_csv('https://raw.githubusercontent.com/mini-pw/2021L-WUM/main/
↪Prace_domowe/Praca_domowa5/clustering.csv')
data.columns = ['first', 'second']
data.head()
```

[2]:

	first	second
0	-96.586516	90.957033
1	-54.143591	-99.153377
2	19.929231	-45.859779
3	-82.941076	84.099186
4	13.389996	-4.016202

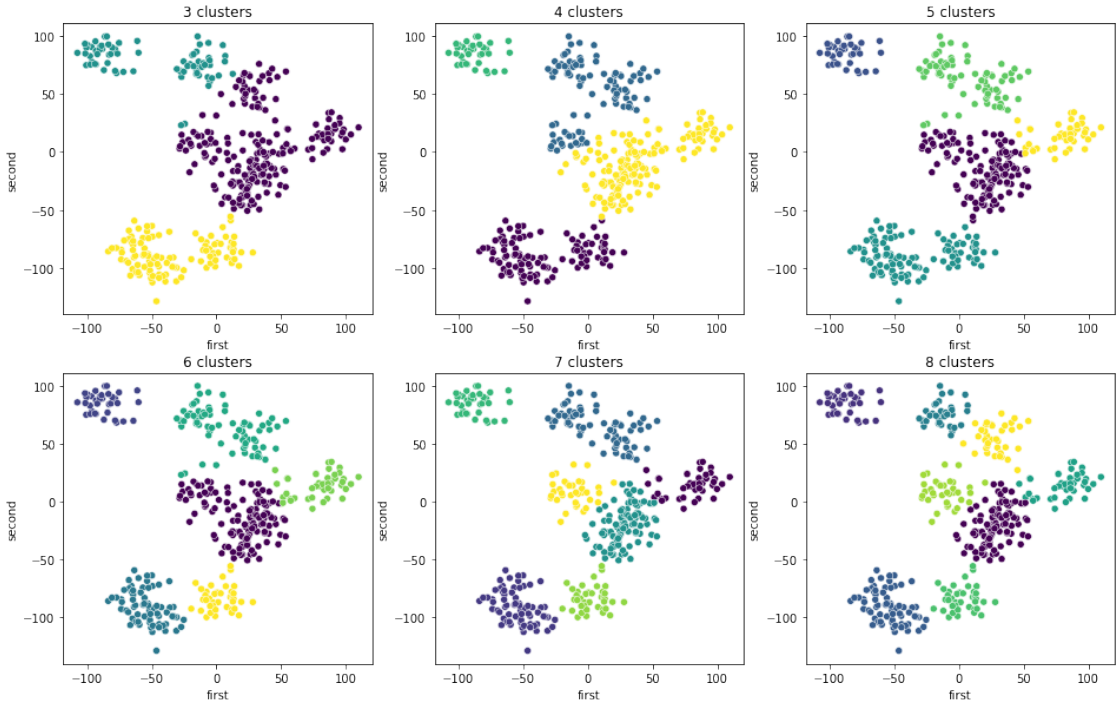
[3]:
```
sns.scatterplot(data = data, x = 'first', y = 'second')
plt.show()
```



0.1 Metoda k-średnich i metoda łokcia

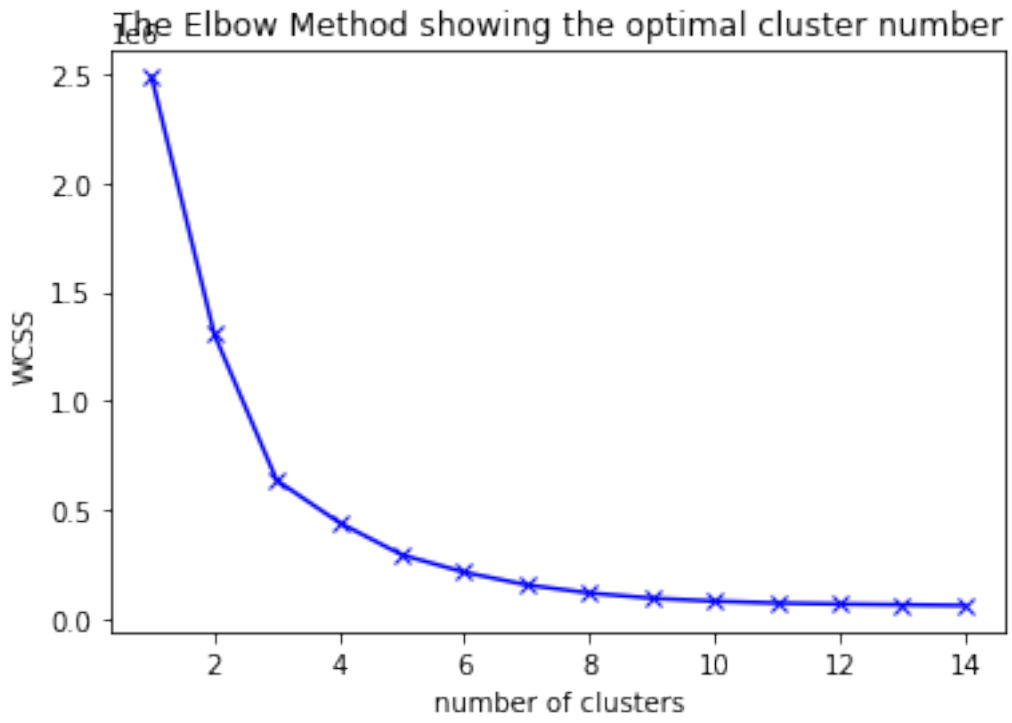
Na oko punkty na powyższym wykresie można sklasteryzować na kilka sposobów, użyjmy metody k-means żeby podzielić punkty na 3, 4, 5, 6, 7 lub 8 klastrów.

[4]:
```
fig, axs = plt.subplots(2, 3, figsize = (16, 10))

for i in range(3, 9):
    kmeans = KMeans(n_clusters = i, random_state = 0)
    kmeans.fit(data)
    y_kmeans = kmeans.predict(data)
    sns.scatterplot(data = data, x = 'first', y = 'second',
                    hue = y_kmeans, palette = 'viridis', legend = False,
                    ax = axs[i // 3 -1, i % 3])
    axs[i // 3 -1, i % 3].set_title(f'{i} clusters')

plt.show()
```



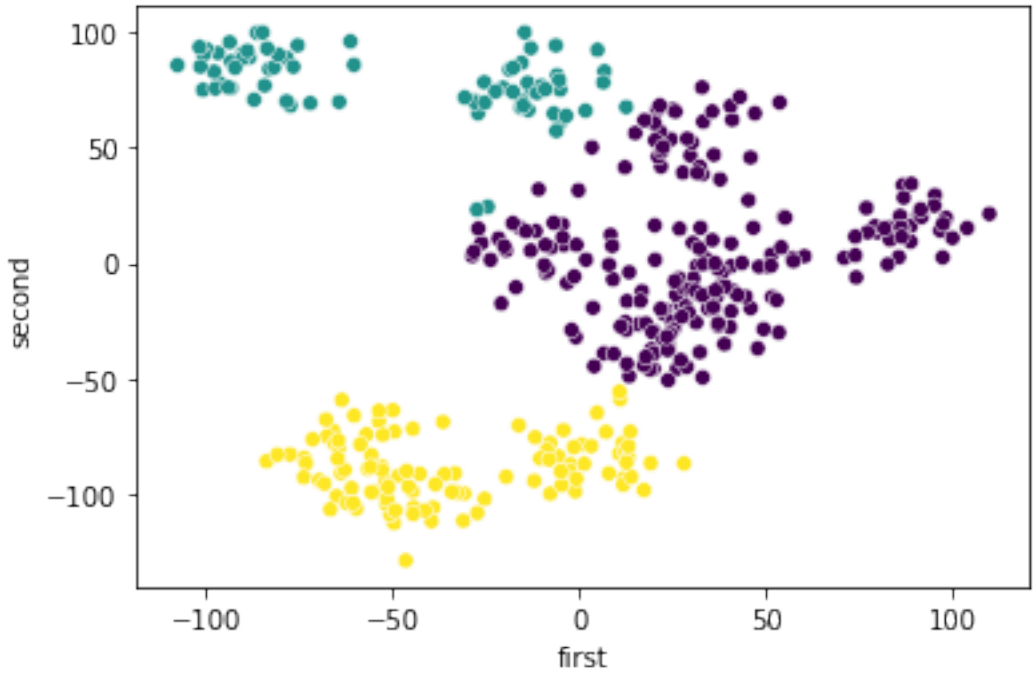
7 i 8 klastrów wygląda chyba na za dużo, użyjmy metody łokcia, żeby znaleźć odpowiednią liczbę, zastosujemy metrykę L_2, któa jest najbardziej naturalna dla dwuwymiarowego problemu.

[5]:
```
scores = []
clusters = []
for i in range(1, 15):
    kmeans = KMeans(n_clusters = i, random_state = 0)
    kmeans.fit(data)
    wcss = kmeans.score(data) * -1
    scores.append(wcss)
    clusters.append(i)

plt.plot(clusters, scores, 'bx-')
plt.xlabel('number of clusters')
plt.ylabel('WCSS')
plt.title('The Elbow Method showing the optimal cluster number')
plt.show()
```



Punlt największego przegięcia to wartość WCSS dla 3 klastrów, więc według metody łokcia to jest odpowiednia liczba, narysujmy wkres jeszcze raz.

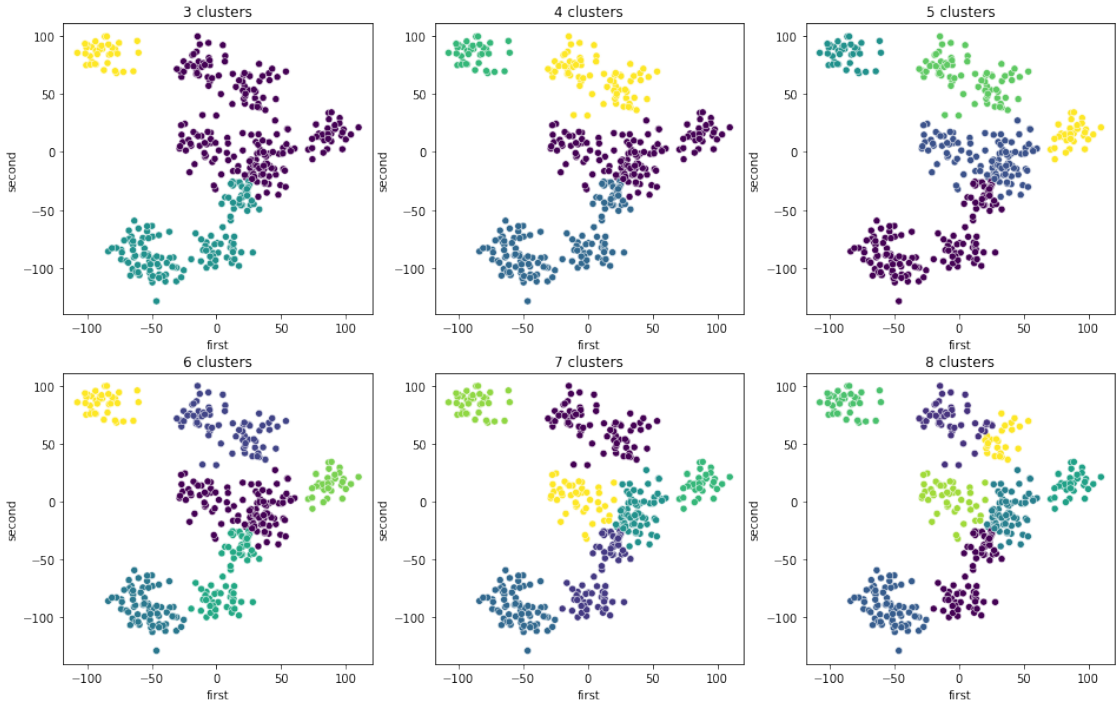
[6]:
```
kmeans = KMeans(n_clusters = 3, random_state = 0)
kmeans.fit(data)
y_kmeans = kmeans.predict(data)
sns.scatterplot(data = data, x = 'first', y = 'second',
                hue = y_kmeans, palette = 'viridis', legend = False)

plt.show()
```



0.2 Klastrowaanie aglomeracyjne i metoda Silhouette

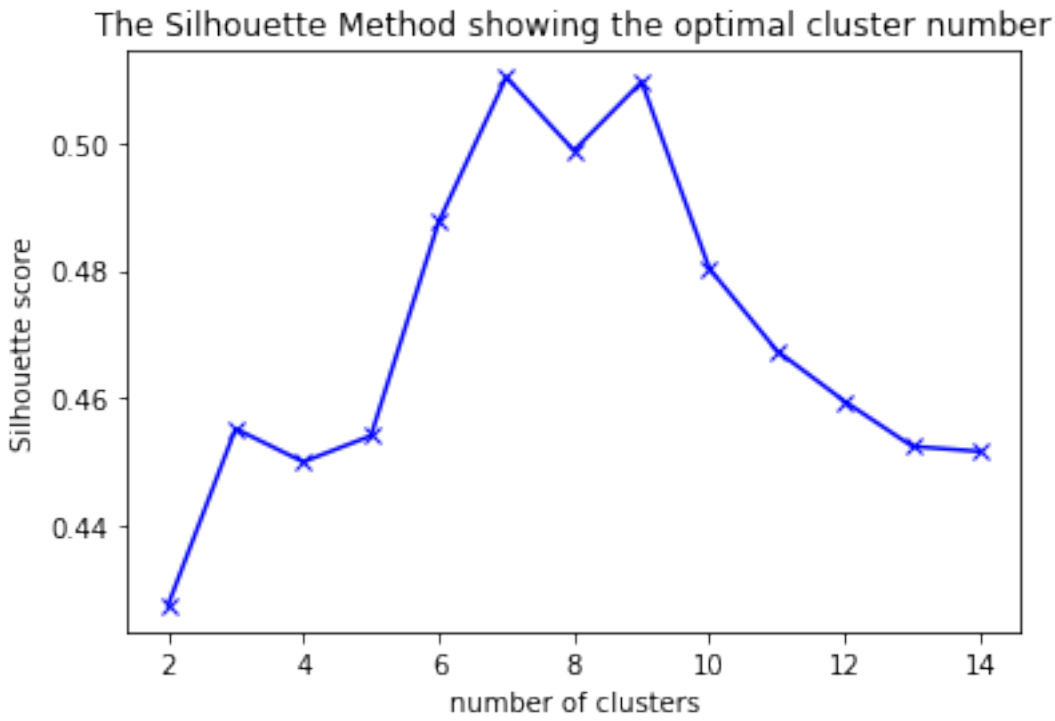
Spróbujmy użyć teraz metody Silhouette do wyznaczenia optymalnej liczby klastrów, w środku zastosujemy model klastrowania aglomeracyjnego (AgglomerativeClustering). Najpierw znowu narysujmy podział zbioru na 3 - 8 klastrów.

[7]:
```
fig, axs = plt.subplots(2, 3, figsize = (16, 10))

for i in range(3, 9):
    aggClus = AgglomerativeClustering(n_clusters = i, linkage = 'complete')
    y_aggClus = aggClus.fit_predict(data)
    sns.scatterplot(data = data, x = 'first', y = 'second',
                    hue = y_aggClus, palette = 'viridis', legend = False,
                    ax = axs[i // 3 -1, i % 3])
    axs[i // 3 -1, i % 3].set_title(f'{i} clusters')

plt.show()
```



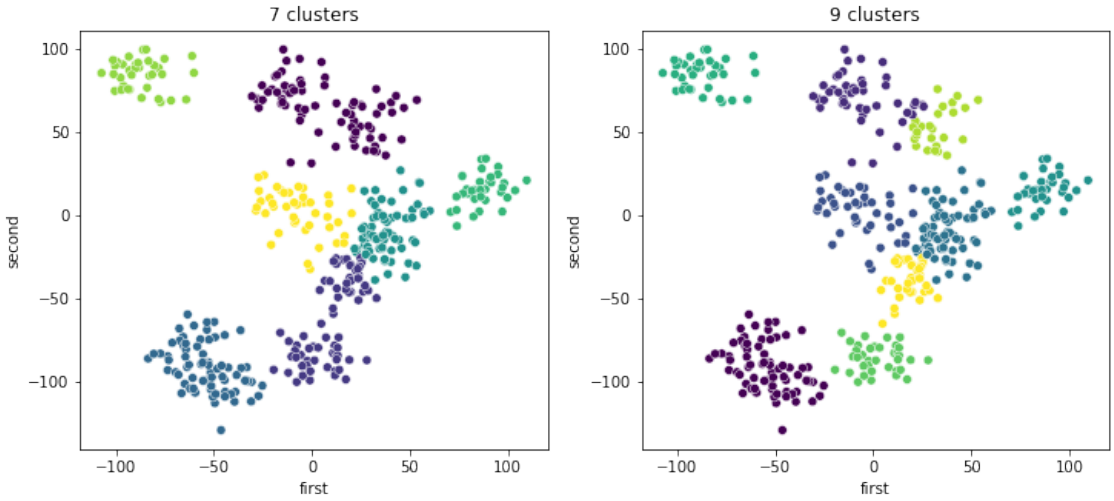
Podziały wyglądają inczaej niż dla k-means, dla większej liczby klastrów sensoweniej niż wcześcniej. Użyjmy metody Silhouette do znalezienia optymalnej liczby.

[8]:
```
scores = []
clusters = []

for i in range(2, 15):
    aggClus = AgglomerativeClustering(n_clusters = i, linkage = 'complete')
    y_aggClus = aggClus.fit_predict(data)
    scores.append(silhouette_score(data, y_aggClus))
    clusters.append(i)

plt.plot(clusters, scores, 'bx-')
plt.xlabel('number of clusters')
plt.ylabel('Silhouette score')
plt.title('The Silhouette Method showing the optimal cluster number')
plt.show()
```



Metoda Silhouette wskazuje, że najlepsza liczba klastów to 7 lub 9, sprawdźmy jak wyglądają wykresy.

[9]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (12, 5))

aggClus = AgglomerativeClustering(n_clusters = 7, linkage = 'complete')
y_aggClus = aggClus.fit_predict(data)

sns.scatterplot(data = data, x = 'first', y = 'second',
                hue = y_aggClus, palette = 'viridis', legend = False,
                ax = ax1)
ax1.set_title('7 clusters')

aggClus = AgglomerativeClustering(n_clusters = 9, linkage = 'complete')
y_aggClus = aggClus.fit_predict(data)

sns.scatterplot(data = data, x = 'first', y = 'second',
                hue = y_aggClus, palette = 'viridis', legend = False,
                ax = ax2)
ax2.set_title('9 clusters')

plt.show()
```


7 clusters
9 clusters
second
first
100
50
0
−50
−100
−100
−50
0
50
100