Praca Domowa 5

Bartosz Siński

In [2]:

```python
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
np.set_seed = 42
```

Wczytanie danych

In [3]:

```python
X = pd.read_csv("./src/clustering.csv")
```

In [4]:

```python
X
```

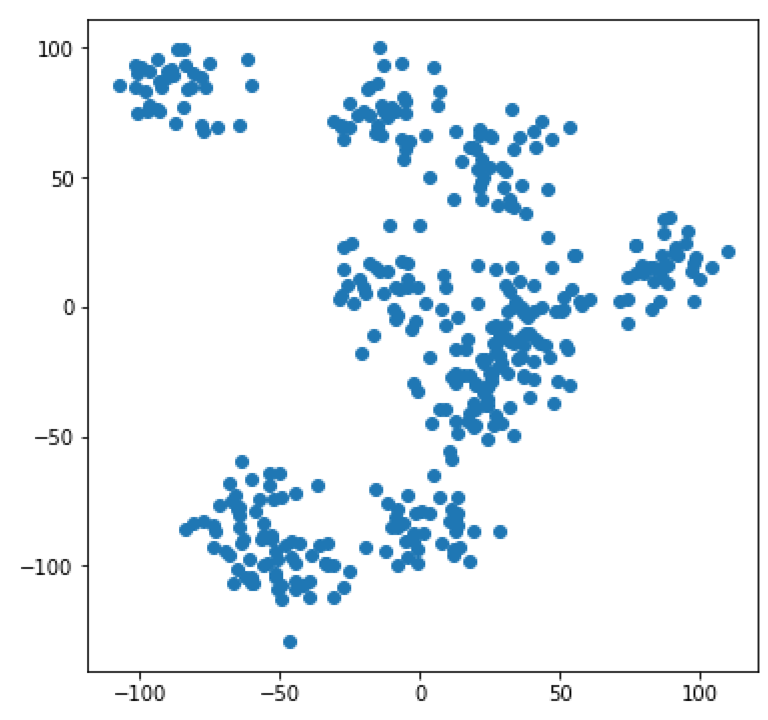
Out[4]:

	4.178890744399839718e+01	5.222018158503714602e+01
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
...	...	...
394	22.423142	50.252807
395	-58.534367	-78.679387
396	36.446549	-11.841887
397	-101.284845	85.096034
398	17.474107	61.890175

399 rows × 2 columns

In [20]:

```python
plt.figure(figsize=(6,6))
plt.scatter(X.iloc[:,0], X.iloc[:,1])
plt.show()
```

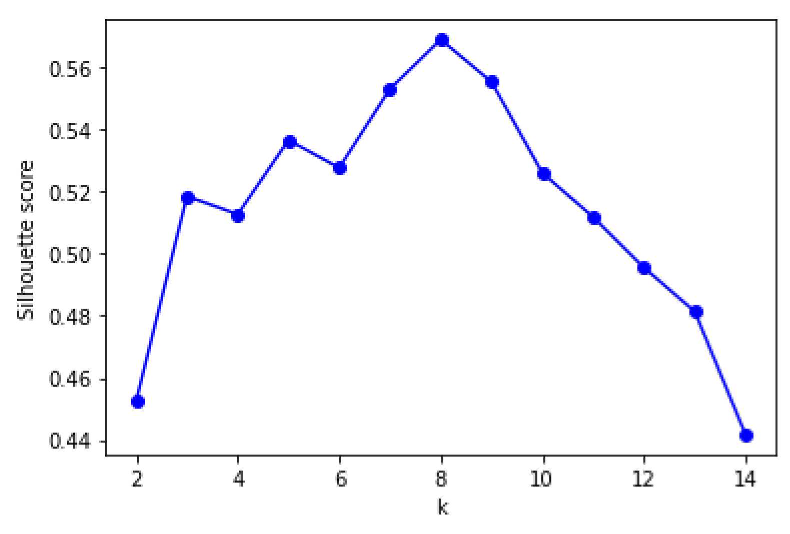


Metoda k-średnich

Znajdziemy najbardziej optymalną liczbę klastrów z użyciem metody Silhouette.

In [38]:

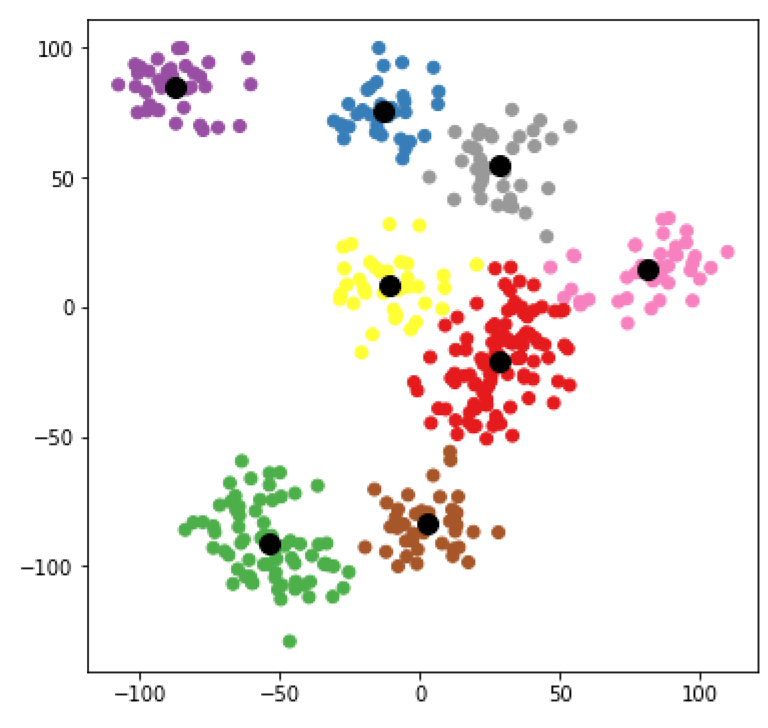
```python
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
ss = []
k = range(2, 15)
for i in k:
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X)
    labels = kmeans.predict(X)
    score = silhouette_score(X,labels, random_state=42)
    ss.append(score)
plt.plot(k,ss,'bo-')
plt.xlabel('k')
plt.ylabel('Silhouette score')
plt.show()
```



Widzimy, że najwyższa wartość Silhouette Score jest dla k=8. Zobaczmy jak wygląda wizualizacja przypisania naszych danych do ośmiu klastrów metodą k-średnich.

In [33]:

```python
kmeans = KMeans(n_clusters=8, random_state=42)
kmeans.fit(X)
preds = kmeans.predict(X)
centers = kmeans.cluster_centers_
plt.figure(figsize=(6,6))
plt.scatter(X.iloc[:,0],X.iloc[:,1],c=preds,cmap="Set1")
plt.scatter(centers[:,0],centers[:,1],s=100,c="black")
plt.show()
```



Hierarchiczna metoda aglomeracyjna z połączeniami Warda

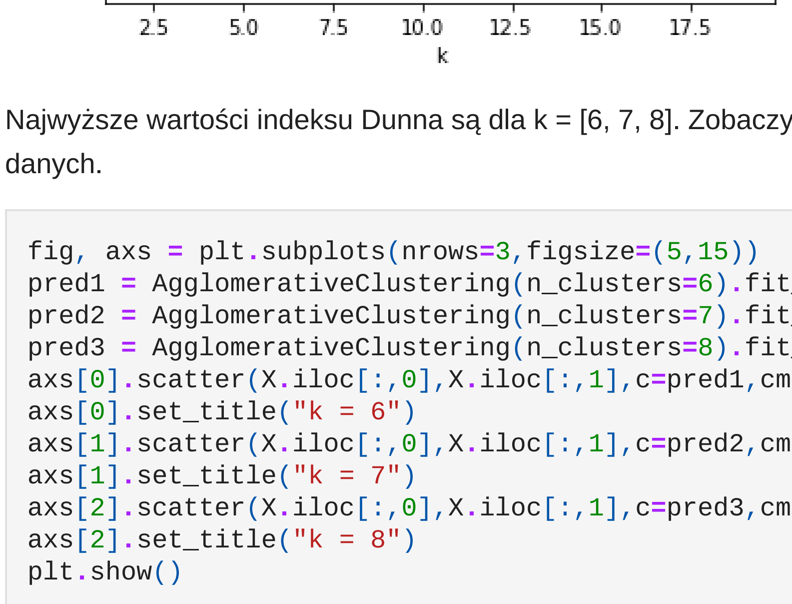
Na poczatku znajdziemy optymalną liczbe klastrów przy użyciu mertyki Dunn Index. Poniższa implementacja pochodzi z https://github.com/jqmviegas/jqm_cvi/blob/master/jqmcvi/base.py. Wylicza ona indeks na podstawie najmniejszej odległości między punktami najbliższych klastrów i najwiekszej odległości pomiędzy klastrami

In [40]:

```python
def delta(ck, cl):
    values = np.ones([len(ck), len(cl)])*10000

    for i in range(0, len(ck)):
        for j in range(0, len(cl)):
            values[i, j] = np.linalg.norm(ck[i]-cl[j])

    return np.min(values)

def big_delta(ci):
    values = np.zeros([len(ci), len(ci)])

    for i in range(0, len(ci)):
        for j in range(0, len(ci)):
            values[i, j] = np.linalg.norm(ci[i]-ci[j])

    return np.max(values)

def dunn(k_list):
    deltas = np.ones([len(k_list), len(k_list)])*1000000
    big_deltas = np.zeros([len(k_list), 1])
    l_range = list(range(0, len(k_list)))

    for k in l_range:
        for l in (l_range[0:k]+l_range[k+1:]):
            deltas[k, l] = delta(k_list[k], k_list[l])

        big_deltas[k] = big_delta(k_list[k])

    di = np.min(deltas)/np.max(big_deltas)
    return di
```

In [60]:

```python
from sklearn.cluster import AgglomerativeClustering
scores = []
k = range(2,20)
for i in k:
    labels = pd.DataFrame(AgglomerativeClustering(n_clusters=i).fit_predict(X))
    labels.columns=["Label"]
    pred = pd.concat([X,labels],axis=1)
    clusters = []
    for j in range(0,i):
        obs = pred.loc[pred.Label==j]
        clusters.append(obs.iloc[:,[0,1]].values)
    scores.append(dunn(clusters))
plt.plot(k,scores,'bo-')
plt.xlabel('k')
plt.ylabel('Dunn Index')
plt.show()
```



Najwyższe wartości indeksu Dunna są dla k = [6, 7, 8]. Zobaczymy jak będą wyglądać klastry naszych danych.

In [73]:

```python
fig, axs = plt.subplots(nrows=3,figsize=(5,15))
pred1 = AgglomerativeClustering(n_clusters=6).fit_predict(X)
pred2 = AgglomerativeClustering(n_clusters=7).fit_predict(X)
pred3 = AgglomerativeClustering(n_clusters=8).fit_predict(X)
axs[0].scatter(X.iloc[:,0],X.iloc[:,1],c=pred1,cmap="Set1")
axs[0].set_title("k = 6")
axs[1].scatter(X.iloc[:,0],X.iloc[:,1],c=pred2,cmap="Set1")
axs[1].set_title("k = 7")
axs[2].scatter(X.iloc[:,0],X.iloc[:,1],c=pred3,cmap="Set1")
axs[2].set_title("k = 8")
plt.show()
```