
UMB W06: klasteryzacja (grupowanie, analiza skupień)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

import umb_tools as umb
```

```
# konfiguracja
plt.rcParams["figure.figsize"] = [5, 4]
pd.set_option("display.float_format", lambda x: "%.4f" % x)
```

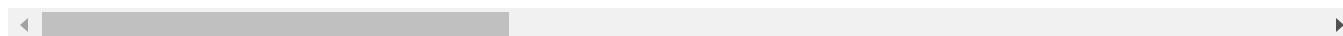
1. Zbiór danych

Wczytanie i normalizacja danych

```
# odczyt pliku TSV (zwracane są: zbiór danych w postaci DataFrame biblioteki Pandas oraz lista kolumn)
(df, c_names) = umb.read_data("data/Leukemia_500_2.txt")
df
```

#Leukemia_500	labels	M23197_at	U05259_rna1_at	X03934_at	U23852_s_at	M31523_at	M84371_rna1_at
ALLB 01	1	8.0279	13.1870	9.2831	9.6166	10.3663	11.2455
ALLB 46	1	5.2095	11.7326	8.6582	8.1799	9.9366	10.3663
ALLB 47	1	8.1649	11.3214	8.2192	9.1898	10.0389	9.9366
ALLB 48	1	7.7944	12.4594	8.4051	8.4512	11.2455	11.2455
ALLB 49	1	8.6073	13.1812	9.0471	9.5411	10.5186	11.2455
...
AML 30	2	10.1421	8.4221	8.3129	7.2574	7.9658	7.9658
AML 29	2	9.1163	8.6865	8.0980	5.1293	8.1241	7.9658
AML 65	2	9.1293	8.2336	8.4388	7.4178	8.5508	8.5508
AML 50	2	9.8580	8.7964	8.4263	8.0768	7.6220	9.9366
AML 66	2	8.4136	9.2831	8.7313	8.8392	9.0444	9.9366

72 rows × 501 columns



```
# pobranie etykiet klas
labels = df.iloc[:, 0].to_numpy()

# pobranie nazw próbek
samples = df.index

# pobranie macierzy danych
data = df.iloc[:, 1:].to_numpy()

# normalizacja
data = StandardScaler().fit_transform(data)
```

Analiza składowych głównych

```
# wykres PCA
umb.pca_plot(data, labels, c_names, show=True)
```

2. Klasteryzacja hierarchiczna (*hierarchical clustering*)

Macierz niepodobieństwa

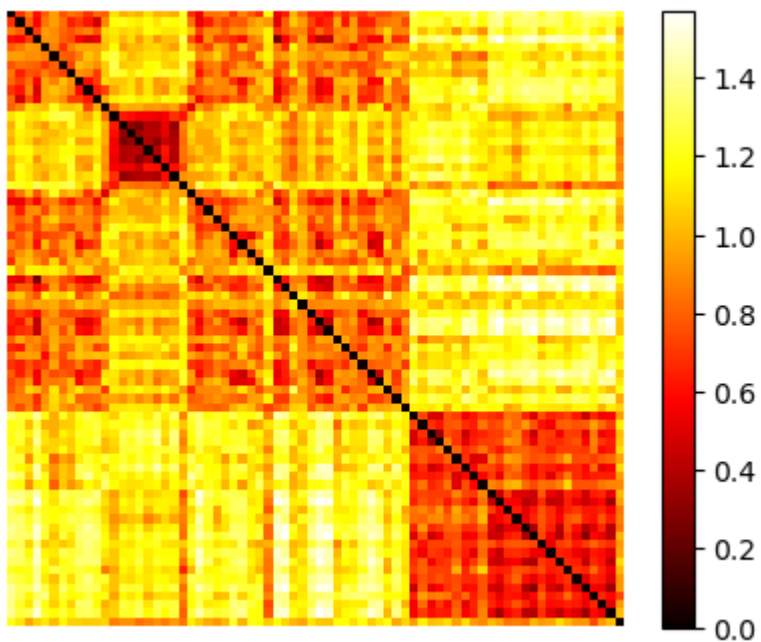
```
import scipy.spatial.distance as dist

# wyznaczenie macierzy niepodobieństwa (miarą jest odległość wynikająca ze współczynnika korelacji)
dmtx = dist.pdist(data, metric="correlation")

# rysowanie
fig, ax = plt.subplots()

im = ax.imshow(dist.squareform(dmtx), interpolation="none", cmap="hot")
fig.colorbar(im, ax=ax)

ax.set_axis_off()
```



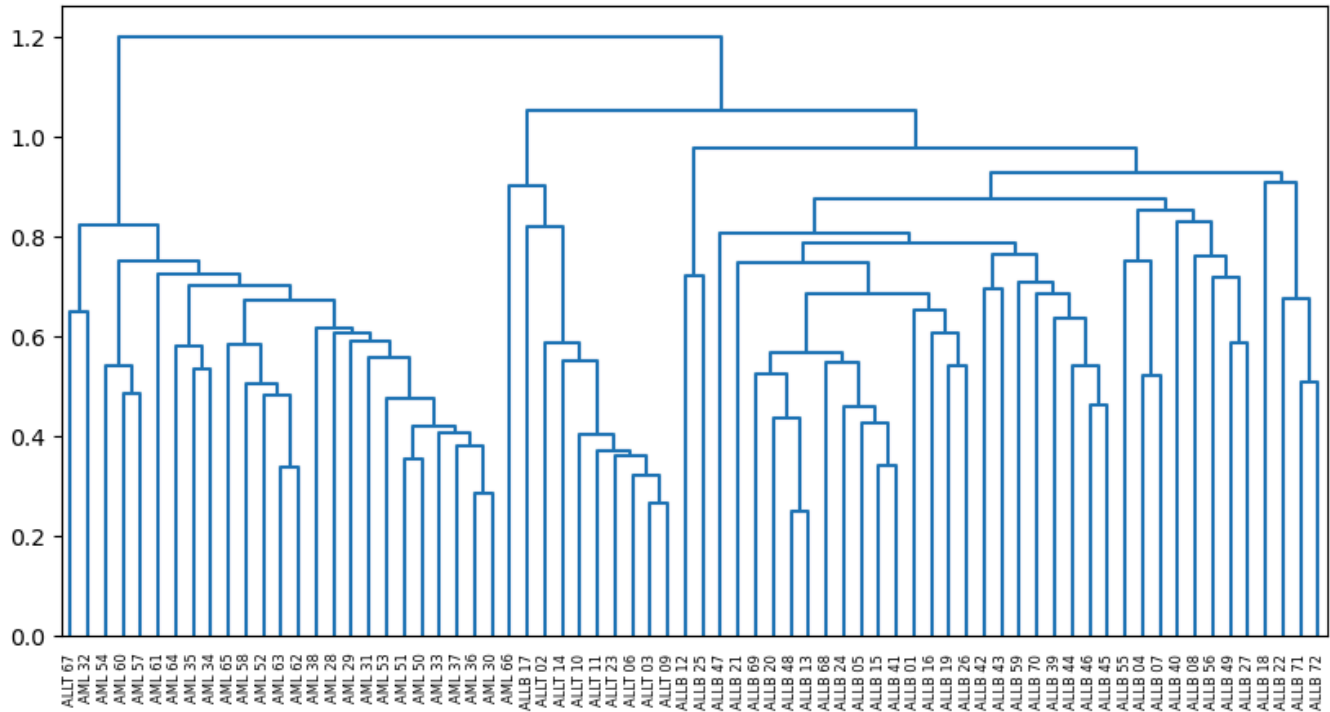
Dendrogram

```
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster

# tworzenie dendrogramu
lnk = linkage(dmtx, "average")

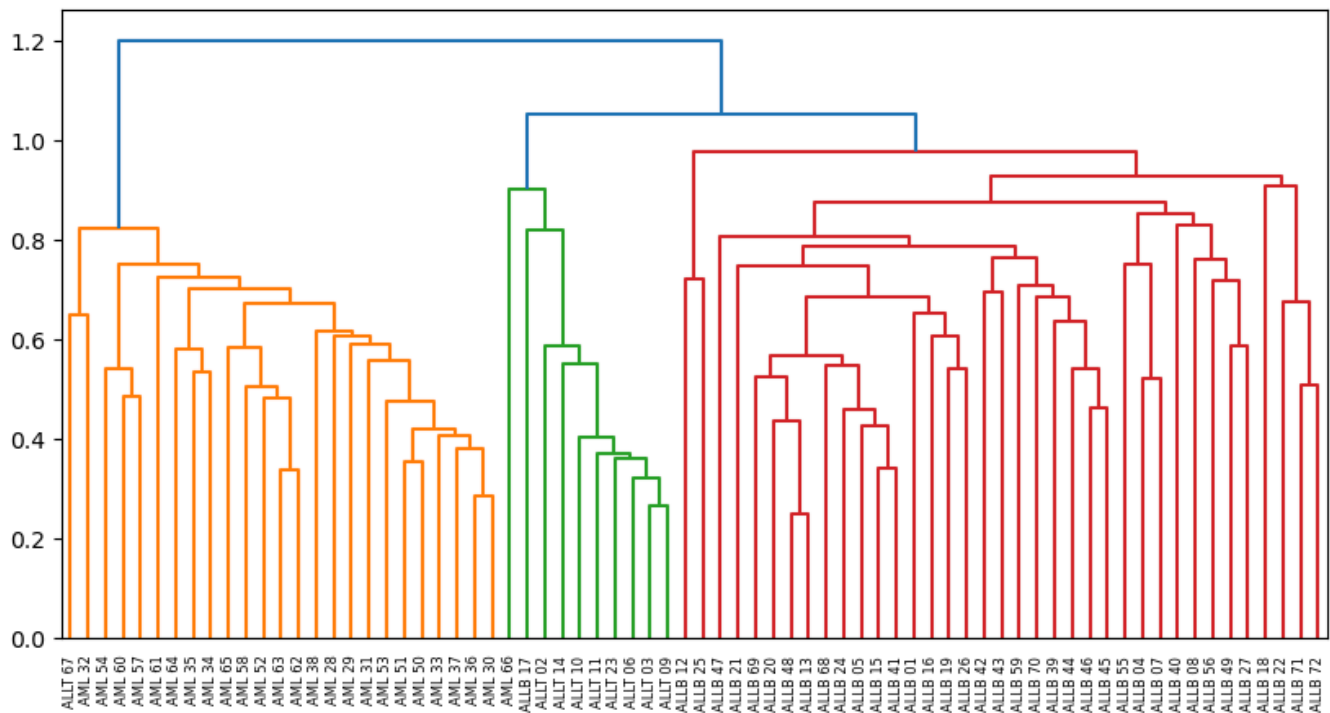
# rysowanie dendrogramu
fig, ax = plt.subplots(figsize=(10, 5))
```

```
dendrogram(lnk, ax=ax, labels=samples, color_threshold=0)
plt.show()
```



Tworzenie klastrow

```
# rysowanie dendrogramu z podziałem na klastry (kryterium jest odległość > 1)
fig, ax = plt.subplots(figsize=(10, 5))
dd = dendrogram(lnk, ax=ax, labels=samples, color_threshold=1)
plt.show()
```



```
# utworzenie klastrów
clusters = fcluster(lnk, t=1, criterion="distance")

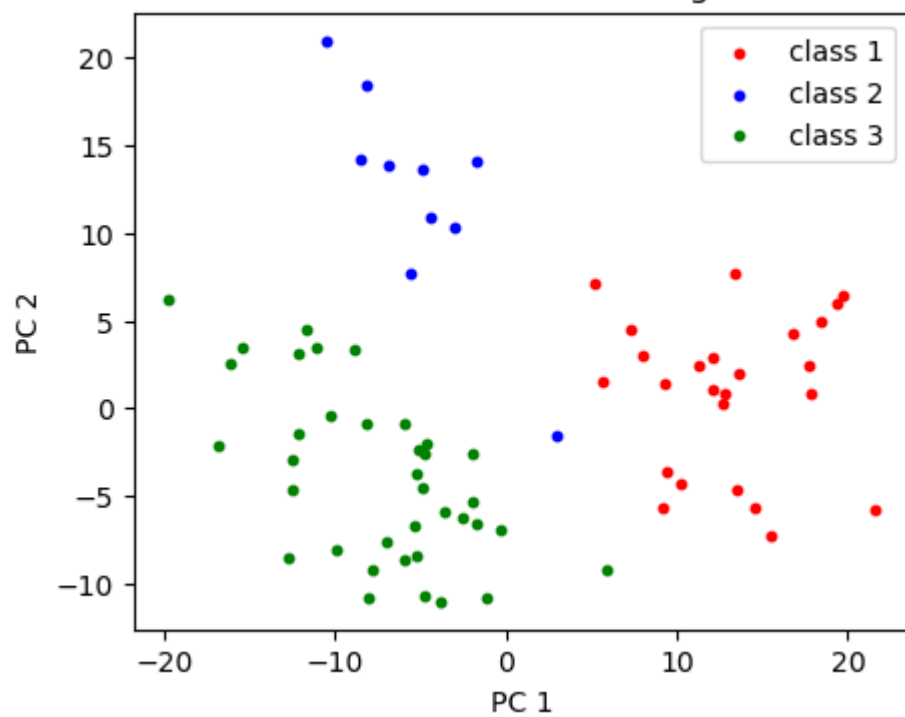
c_df = pd.DataFrame({"samples": samples,
                     "labels": clusters})
c_df
```

	samples	labels
0	ALLB 01	3
1	ALLB 46	3
2	ALLB 47	3
3	ALLB 48	3
4	ALLB 49	3
...
67	AML 30	1
68	AML 29	1
69	AML 65	1
70	AML 50	1
71	AML 66	2

72 rows × 2 columns

```
# wykres PCA z uwzględnieniem wyników klasteryzacji
umb.pca_plot(data, c_df.labels, title="Hierarchical clustering")
```

Hierarchical clustering



3. Algorytm K-średnich (K-means)

```
from sklearn.cluster import KMeans
```

```
# wykonanie klasteryzacji
km_model = KMeans(n_clusters=3, init="random", n_init=10, random_state=0)
km_model = km_model.fit(data)

# pobranie informacji o klastrach
c_df = pd.DataFrame({"samples": samples,
                     "labels": km_model.labels_})
c_df
```

e:\Projects\Python\Anaconda\jupyter\lib\site-packages\sklearn\cluster_kmeans.py:1440: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.

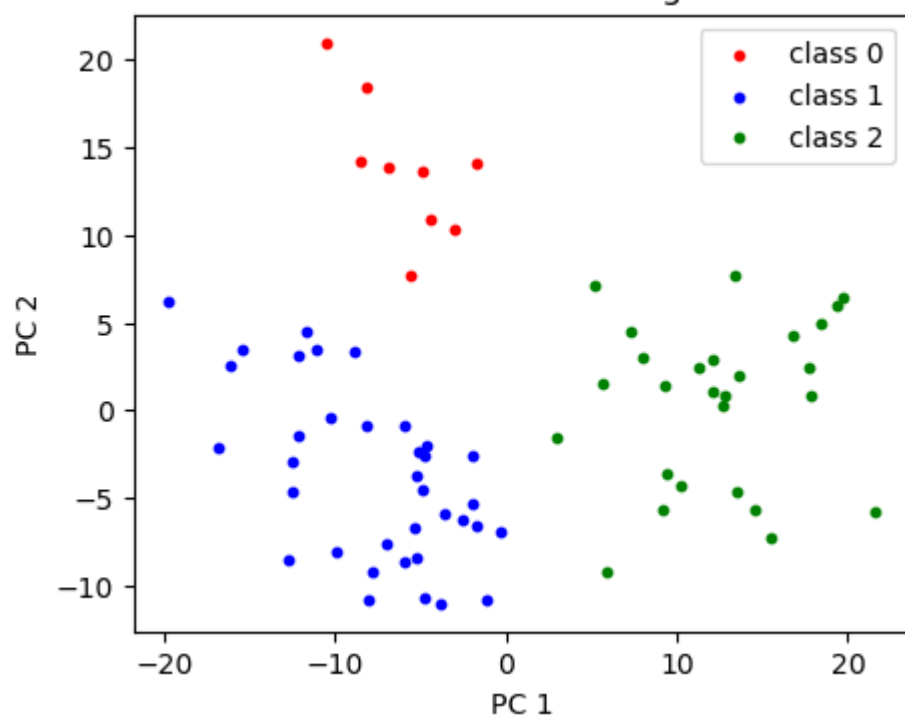
warnings.warn(

	samples	labels
0	ALLB 01	1
1	ALLB 46	1
2	ALLB 47	1
3	ALLB 48	1
4	ALLB 49	1
...
67	AML 30	2
68	AML 29	2
69	AML 65	2
70	AML 50	2
71	AML 66	2

72 rows × 2 columns

```
# wykres PCA z uwzględnieniem wyników klasteryzacji
umb.pca_plot(data, c_df.labels, title="k-means clustering")
```


k-means clustering



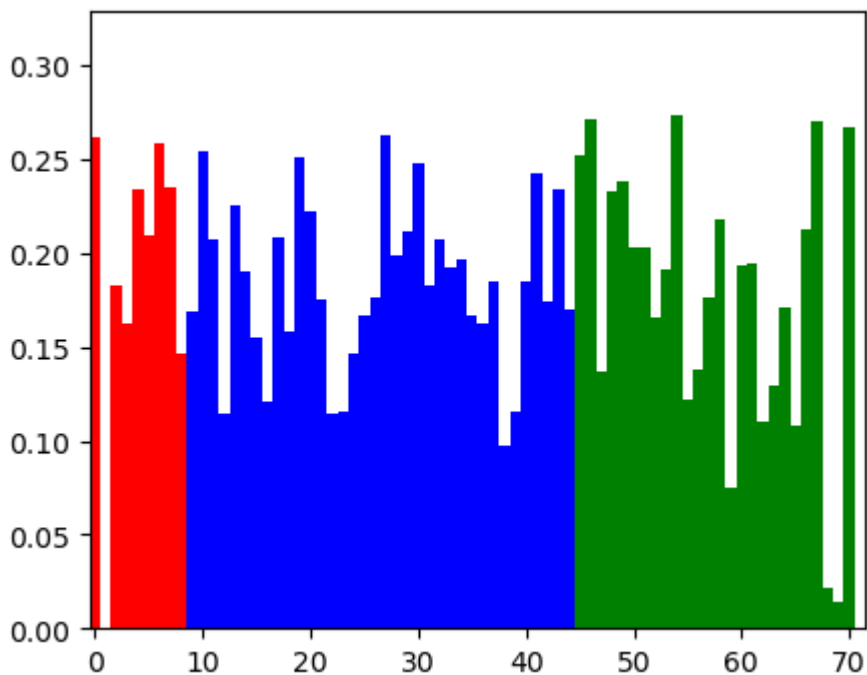
2.4 Ocena wyników klasteryzacji (miara *silhouette*)

```
from sklearn.metrics import silhouette_samples, silhouette_score
```

```
# wyznaczenie wartości miary silhouette dla próbek
sil = silhouette_samples(data, labels=c_df.labels, metric="euclidean")

# rysowanie wartości miary silhouette dla próbek
index=np.argsort(c_df.labels)

umb.bar_plot(sil[index], c_df.labels[index])
```



```
# wartość średnia miary silhouette
mean_sil = silhouette_score(data, labels=c_df.labels, metric="euclidean")

print(f"\nmean sil = {mean_sil:.4f} ({np.mean(sil):.4f})")
```

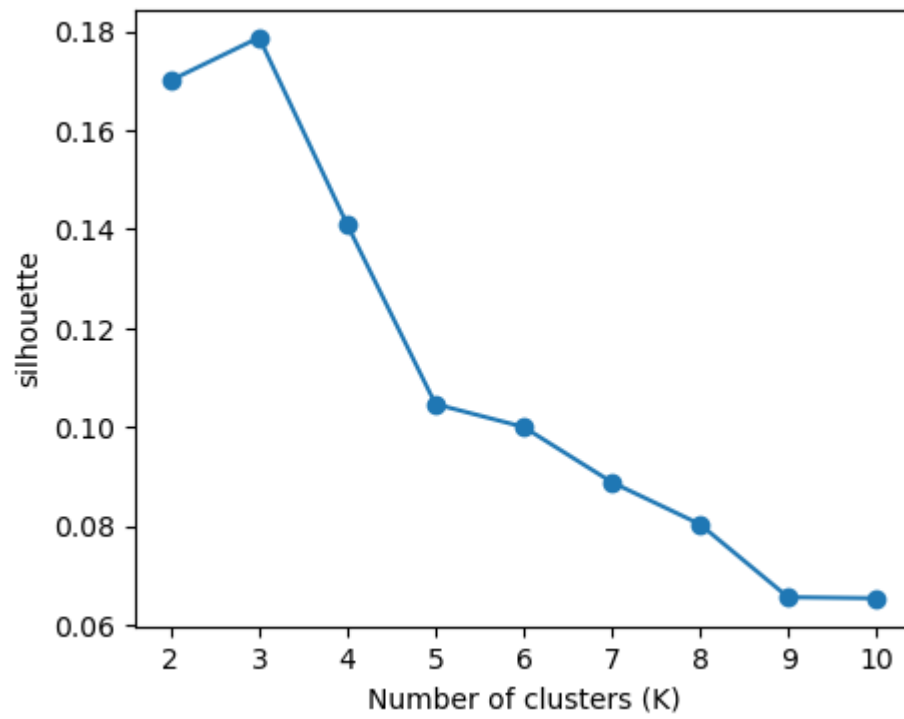
```
mean sil = 0.1788 (0.1788)
```

Estymacja liczby klastrow dla algorytmu *K*-średnich

```
# uruchomienie alg. K-średnich dla K = 2, 3, ..., 10 i wyznaczenie średniej miary silhouette

sil_scores = []
for k in range(2, 11):
    km_model = (KMeans(n_clusters=k, init="random", n_init=10, random_state=0)).fit(data)

    mean_sil = silhouette_score(data, km_model.labels_)
```

Optimal number of clusters: 3
