SPRAWOZDANIE

Zajęcia: Uczenie maszynowe

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 3 Data 06.12.2024 Temat: Uczenie maszynowe w praktyce: analiza skupień. Wariant 8	Imię Nazwisko Hubert Mentel Informatyka II stopień, niestacjonarne, 1 semestr, gr.1a

1. Zadanie:

Metoda PCA

- Wykonaj analizę PCA na własnym zbiorze danych.
- Wykonaj wizualizację skupień dla 2 lub 3 głównych składowych.
- Porównaj wyniki klasteryzacji przed i po redukcji wymiarowości.

Metody niehierarchiczne

- Wykonaj klasteryzację k-means dla wybranego zbioru danych.
- Przeprowadź analizę dla różnych wartości kk. Wybierz optymalne kk, korzystając z metody "łokcia" (ang. elbow method).
- Porównaj wyniki z wcześniejszą analizą PCA (jeśli dane zostały wcześniej zredukowane wymiarowościowo).

Metody hierarchiczne

- Wykonaj klasteryzację hierarchiczną na dowolnym zbiorze danych. Przeanalizuj wpływ różnych metod łączenia (np. Ward, single linkage, complete linkage) na strukturę dendrogramu.
- Wyodrębnij klastry na różnych poziomach dendrogramu. Porównaj otrzymane wyniki z klasteryzacją k-means.
- Wykorzystaj dane wielowymiarowe i wykonaj redukcję wymiarowości (np. PCA) przed zastosowaniem klasteryzacji hierarchicznej.

Pliki dostępne są na GitHubie pod linkiem:

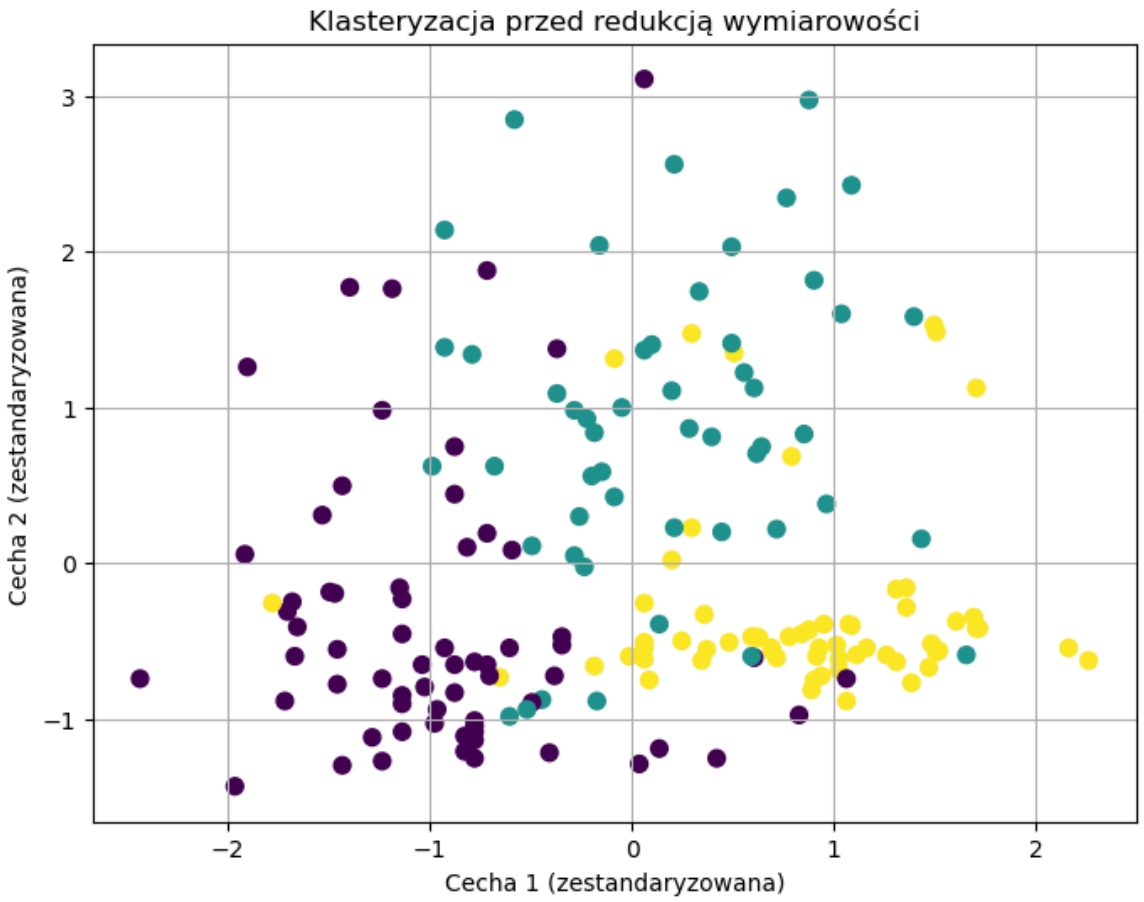
https://github.com/HubiPX/NOD/tree/master/UM/Zadanie%203

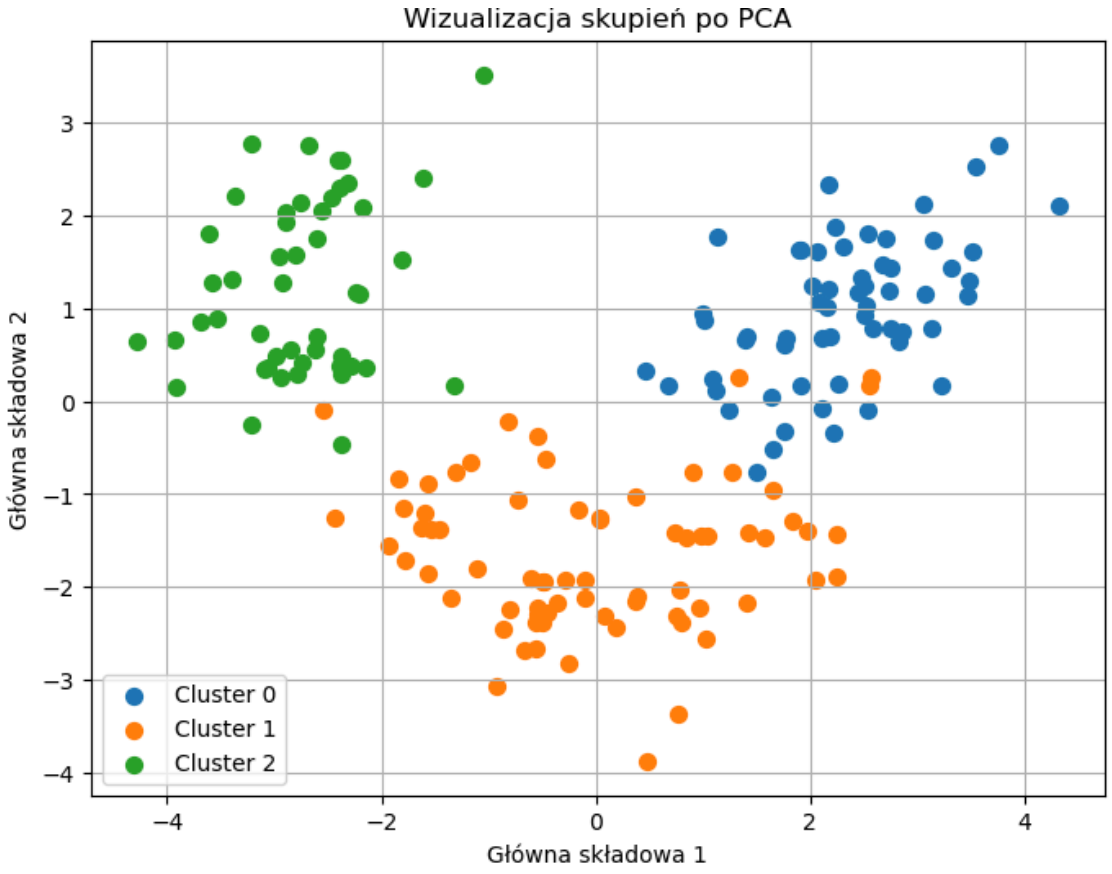
2. Opis programu opracowanego (kody źródłowe, zrzuty ekranu)

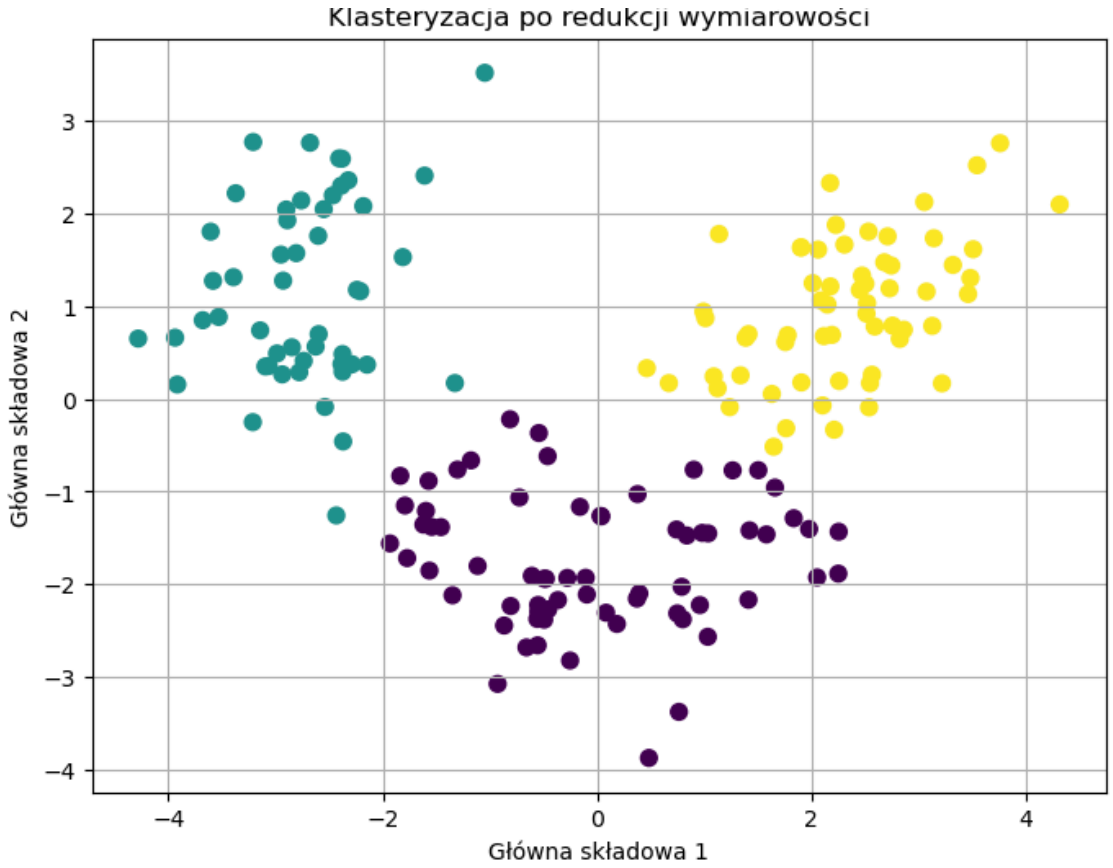
```
import os
os.environ["OMP_NUM_THREADS"] = "2"
os.environ["LOKY_MAX_CPU_COUNT"] = "1"

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.datasets import load_wine

# 1. Wczytanie danych
data = load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
labels = data.target

# 2. Standaryzacja danych
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df)

# 3. Redukcja wymiarowości za pomocą PCA
```

```
pca = PCA(n_components=2)  # Redukcja do 2 głównych składowych
data_pca = pca.fit_transform(data_scaled)

# 4. Klasteryzacja przed redukcją wymiarowości
kmeans_full = KMeans(n_clusters=3, random_state=42)
labels_full = kmeans_full.fit_predict(data_scaled)

# 5. Klasteryzacja po redukcji wymiarowości
kmeans_pca = KMeans(n_clusters=3, random_state=42)
labels_pca = kmeans_pca.fit_predict(data_pca)

# 6. Wizualizacja wyników klasteryzacji przed redukcją
plt.figure(figsize=(8, 6))
plt.scatter(data_scaled[:, 0], data_scaled[:, 1], c=labels_full, cmap='viridis',
s=50)
plt.title('Klasteryzacja przed redukcją wymiarowości')
plt.xlabel('Cecha 1 (zestandaryzowana)')
plt.ylabel('Cecha 2 (zestandaryzowana)')
plt.grid()
plt.show()

# 7. Wizualizacja wyników PCA
plt.figure(figsize=(8, 6))
for label in np.unique(labels):
    plt.scatter(
        data_pca[labels == label, 0],
        data_pca[labels == label, 1],
```

```
        label=f"Cluster {label}",
        s=50
    )
plt.xlabel('Główna składowa 1')
plt.ylabel('Główna składowa 2')
plt.title('Wizualizacja skupień po PCA')
plt.legend()
plt.grid()
plt.show()

# 8. Wizualizacja wyników klasteryzacji po redukcji
plt.figure(figsize=(8, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=labels_pca, cmap='viridis', s=50)
plt.title('Klasteryzacja po redukcji wymiarowości')
plt.xlabel('Główna składowa 1')
plt.ylabel('Główna składowa 2')
plt.grid()
plt.show()

# 9. Wyjaśnienie wariancji przez główne składowe
explained_variance = pca.explained_variance_ratio_
print(f"Wyjaśniona wariancja przez PCA: {explained_variance}")
```


Klasteryzacja przed redukcją wymiarowości
Cecha 2 (zestandaryzowana)
Cecha 1 (zestandaryzowana)
3
2
1
0
−1
−2
−1
0
1
2


Wizualizacja skupień po PCA
Główna składowa 2
Główna składowa 1
Cluster 0
Cluster 1
Cluster 2
3
2
1
0
−1
−2
−3
−4
−4
−2
0
2
4

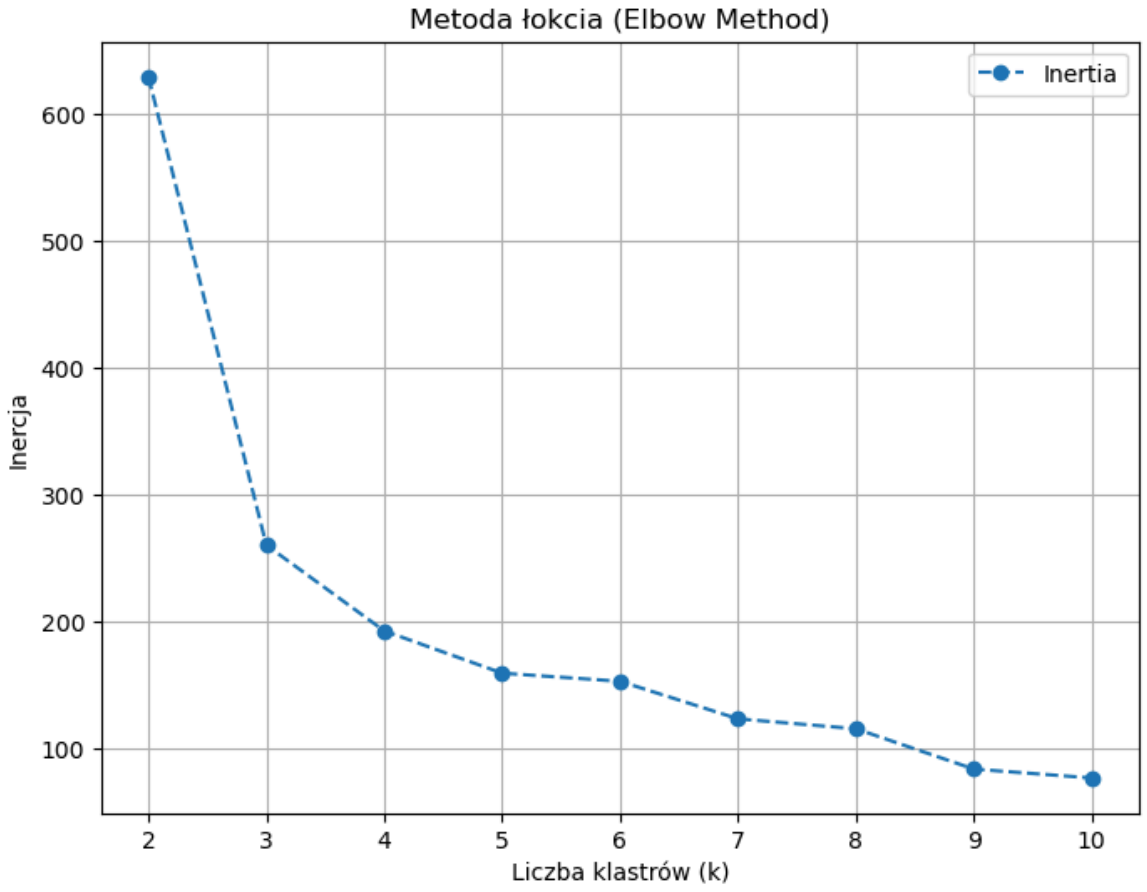


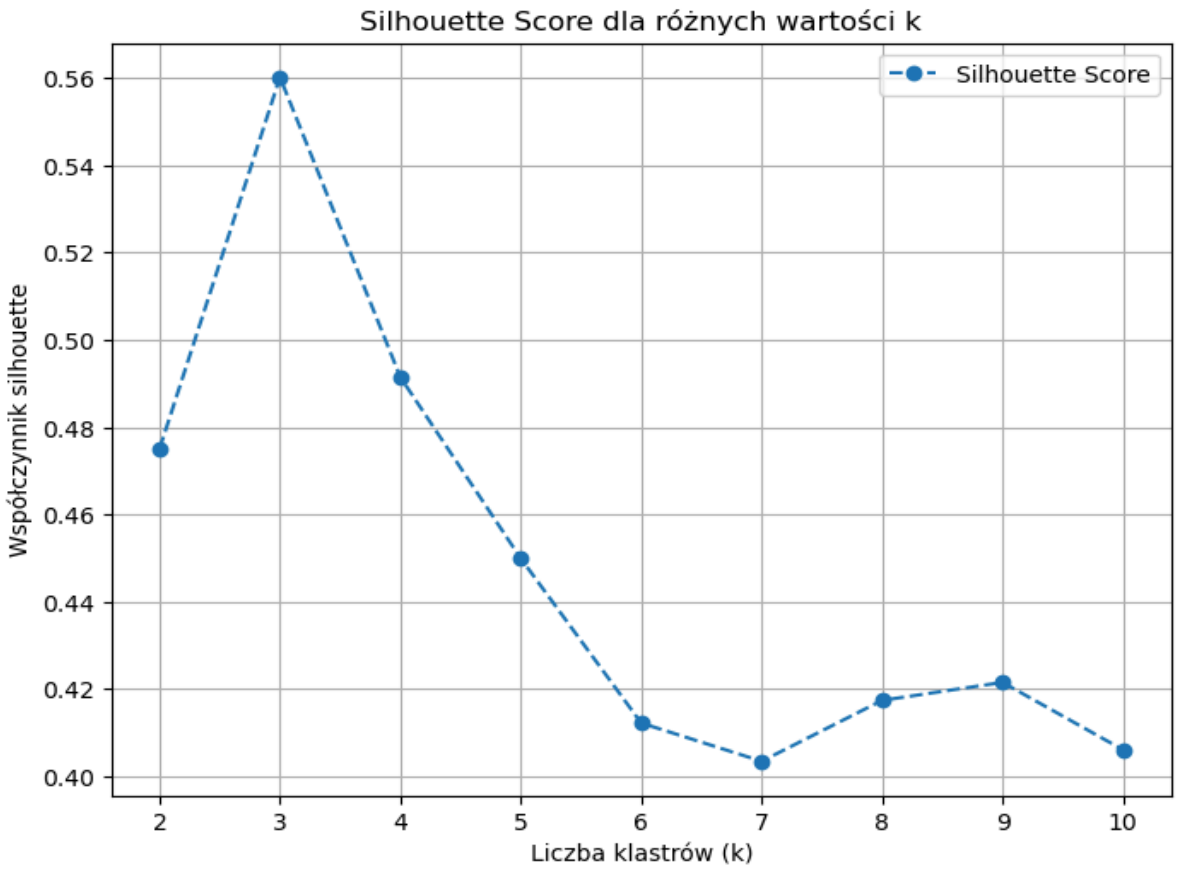

```
Wyjaśniona wariancja przez PCA: [0.36198848 0.1920749 ]
```

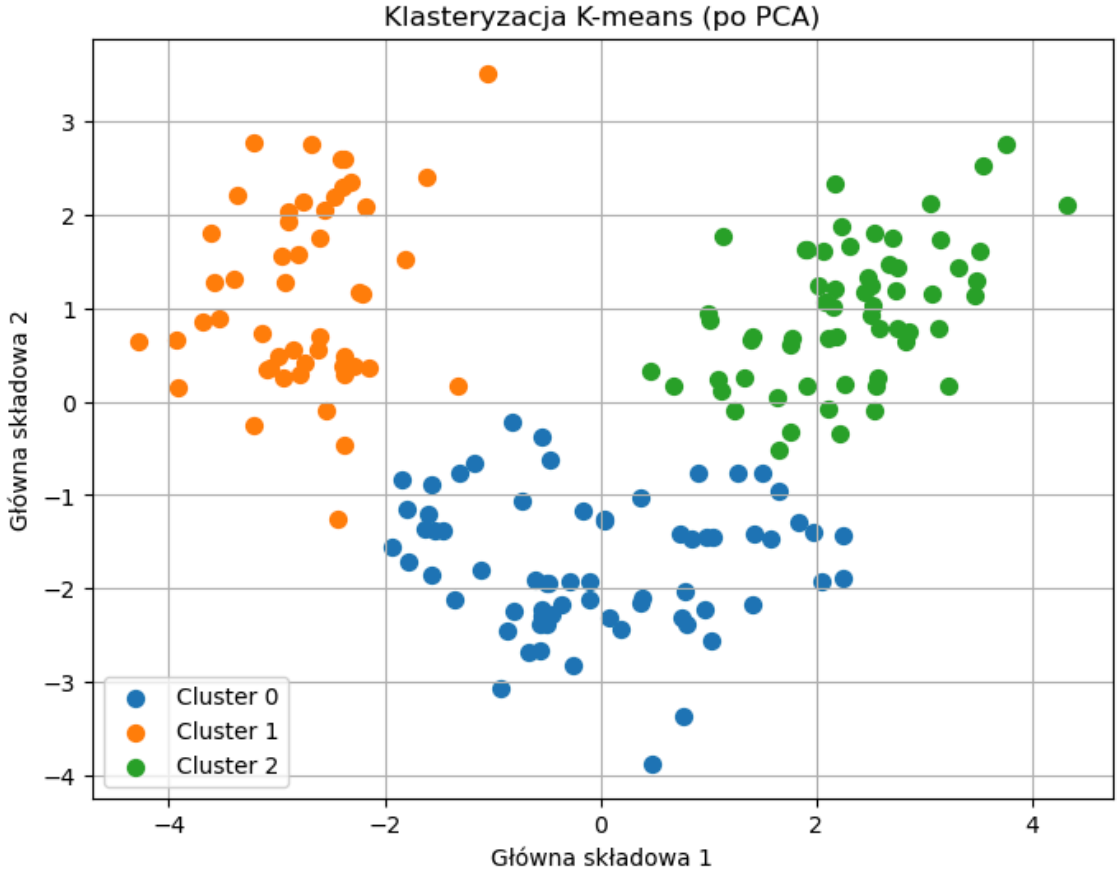
```
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Ustawienie liczby wątków na 1, aby uniknąć problemów z MKL na Windows
os.environ["OMP_NUM_THREADS"] = "1"

# 1. Klasteryzacja K-means dla różnych wartości k
inertia = []  # Lista na wartości inercji
silhouette_scores = []  # Lista na wyniki metryki silhouette
k_values = range(2, 11)  # Zakres wartości k (od 2 do 10)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data_pca)  # Użycie zredukowanych danych PCA
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(data_pca, kmeans.labels_))

# 2. Wizualizacja metody "łokcia"
plt.figure(figsize=(8, 6))
plt.plot(k_values, inertia, marker='o', linestyle='--', label='Inertia')
plt.xlabel('Liczba klastrów (k)')
plt.ylabel('Inercja')
plt.title('Metoda łokcia (Elbow Method)')
```

```
plt.grid()
plt.legend()
plt.show()

# 3. Wizualizacja współczynnika silhouette
plt.figure(figsize=(8, 6))
plt.plot(k_values, silhouette_scores, marker='o', linestyle='--', label='Silhouette Score')
plt.xlabel('Liczba klastrów (k)')
plt.ylabel('Współczynnik silhouette')
plt.title('Silhouette Score dla różnych wartości k')
plt.grid()
plt.legend()
plt.show()

# 4. Wybór optymalnego k (np. na podstawie metody łokcia)
optimal_k = 3  # Przykładowa wartość wybrana po analizie
kmeans_optimal = KMeans(n_clusters=optimal_k, random_state=42)
kmeans_optimal.fit(data_pca)

# 5. Wizualizacja wyników klasteryzacji K-means (optymalne k)
plt.figure(figsize=(8, 6))
for cluster in range(optimal_k):
    plt.scatter(
        data_pca[kmeans_optimal.labels_ == cluster, 0],
        data_pca[kmeans_optimal.labels_ == cluster, 1],
        label=f'Cluster {cluster}',
```

```
        s=50
    )
plt.xlabel('Główna składowa 1')
plt.ylabel('Główna składowa 2')
plt.title('Klasteryzacja K-means (po PCA)')
plt.legend()
plt.grid()
plt.show()
```

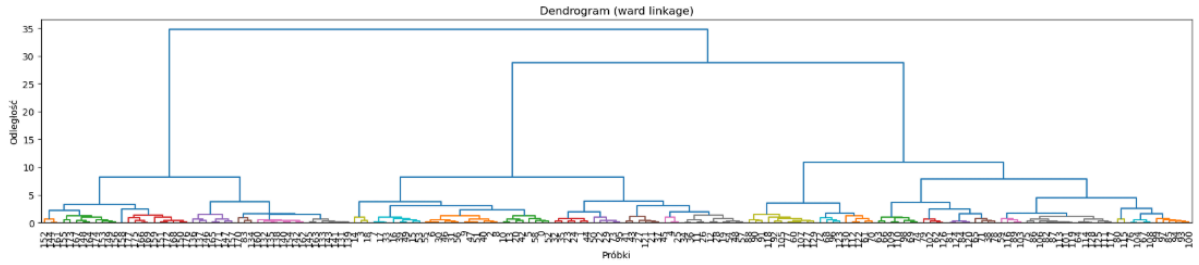


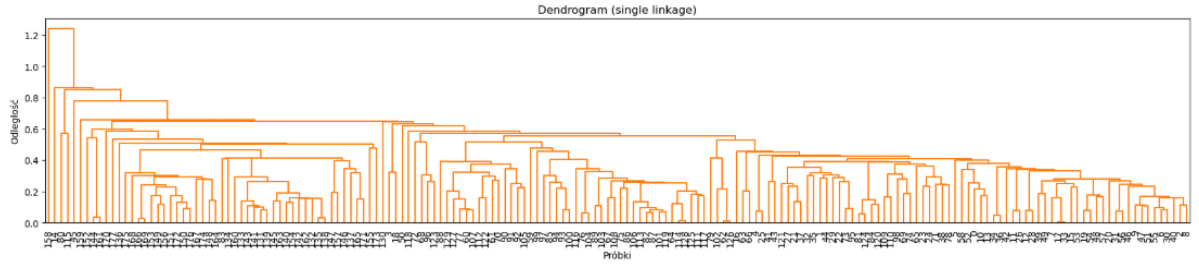

Silhouette Score dla różnych wartości k
Silhouette Score
Współczynnik silhouette
Liczba klastrów (k)
0.56
0.54
0.52
0.50
0.48
0.46
0.44
0.42
0.40
2
3
4
5
6
7
8
9
10

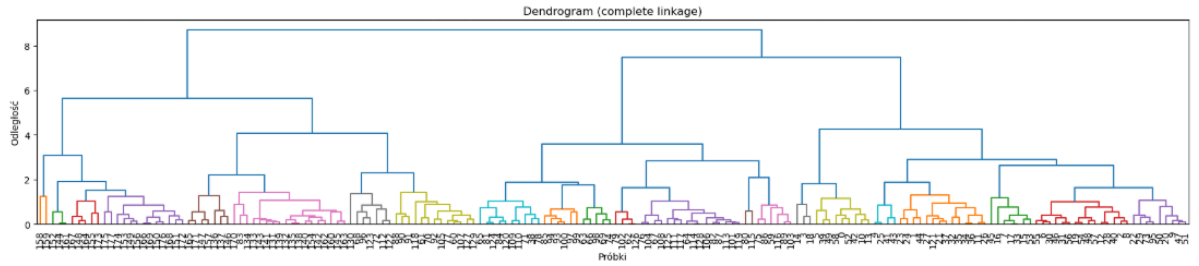

Klasteryzacja K-means (po PCA)
Cluster 0
Cluster 1
Cluster 2
Główna składowa 1
Główna składowa 2
3
2
1
0
−1
−2
−3
−4
−4
−2
0
2
4

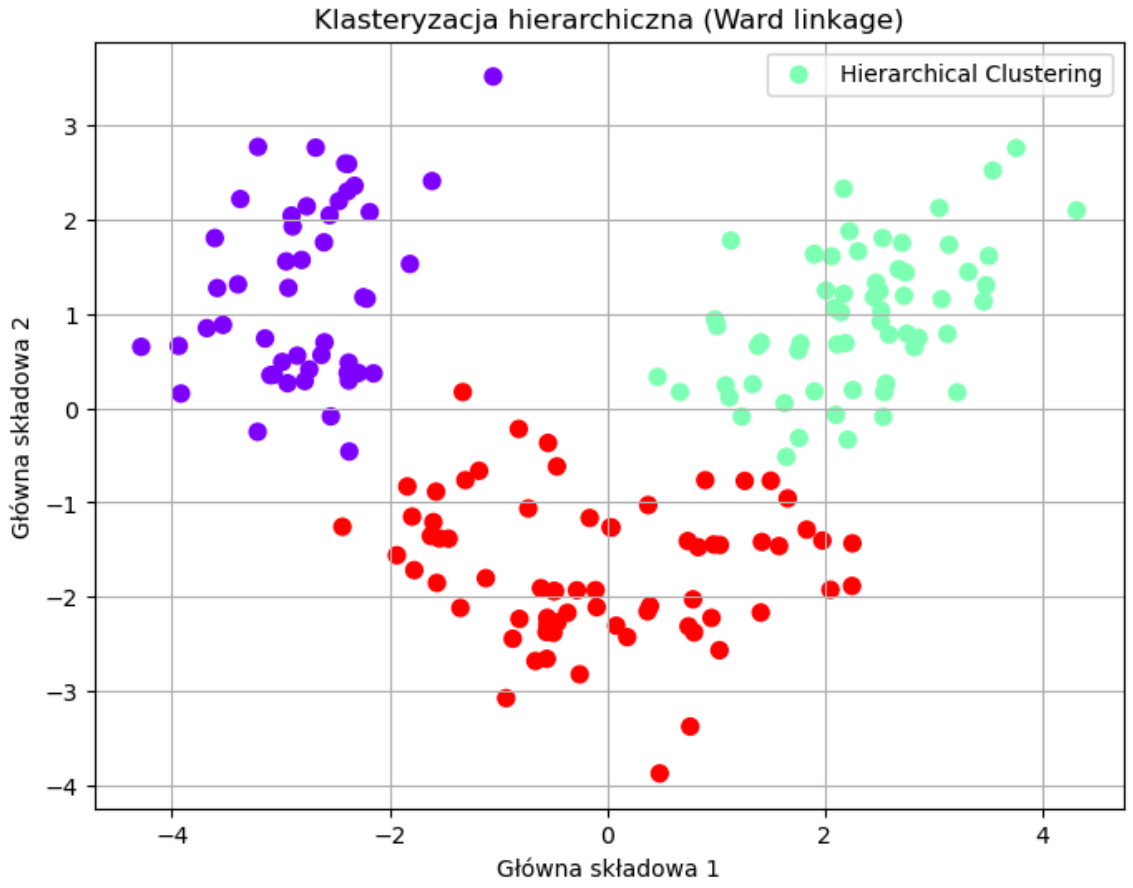
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster

# Ustawienie liczby wątków na 1, aby uniknąć problemów z MKL na Windows
os.environ["OMP_NUM_THREADS"] = "1"

# 1. Wczytanie danych (przykładowy zbiór Iris)
data = load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
labels = data.target

# 2. Standaryzacja danych
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df)

# 3. Redukcja wymiarowości za pomocą PCA
pca = PCA(n_components=2)  # Redukcja do 2 głównych składowych
data_pca = pca.fit_transform(data_scaled)

# 4. Klasteryzacja hierarchiczna
```

```
methods = ['ward', 'single', 'complete']
plt.figure(figsize=(18, 12))  # Zwiększenie rozmiaru wykresów
for i, method in enumerate(methods):
    plt.subplot(3, 1, i + 1)  # Każdy dendrogram w osobnym wierszu
    Z = linkage(data_pca, method=method)
    dendrogram(Z, leaf_rotation=90, leaf_font_size=10, color_threshold=1.5)
    plt.title(f'Dendrogram ({method} linkage)')
    plt.xlabel('Próbki')
    plt.ylabel('Odległość')
plt.tight_layout()
plt.show()

# 5. Wyodrębnienie klastrów na różnych poziomach dendrogramu
Z = linkage(data_pca, method='ward')
clusters_hierarchical = fcluster(Z, t=3, criterion='maxclust')

# 6. Porównanie wyników klasteryzacji hierarchicznej z K-means
plt.figure(figsize=(8, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=clusters_hierarchical,
cmap='rainbow', s=50, label='Hierarchical Clustering')
plt.xlabel('Główna składowa 1')
plt.ylabel('Główna składowa 2')
plt.title('Klasteryzacja hierarchiczna (Ward linkage)')
plt.legend()
plt.grid()
plt.show()
```


Dendrogram (ward linkage)
Odległość
Próbki


Dendrogram (single linkage)
Odległość
Próbki


Dendrogram (complete linkage)
Odległość
Próbki


Klasteryzacja hierarchiczna (Ward linkage)
Hierarchical Clustering
Główna składowa 2
Główna składowa 1

3. Wnioski

Analiza skupień jest kluczową metodą uczenia nienadzorowanego, umożliwiającą identyfikację naturalnych grup w danych. W przeprowadzonych badaniach wykorzystano redukcję wymiarowości przy pomocy PCA, co pozwoliło na uproszczenie wielowymiarowego zbioru danych do dwóch głównych składowych. Taka transformacja nie tylko ułatwia wizualizację struktur, ale także pomaga zachować istotną część informacji, co ma bezpośredni wpływ na jakość przeprowadzonej klasteryzacji.

W przypadku klasteryzacji niehierarchicznej zastosowano metodę K-means. Analiza dla różnych wartości liczby klastrów przy użyciu metody "łokcia" oraz obliczenie współczynnika silhouette umożliwiły wskazanie optymalnej liczby skupień, co potwierdziło zgodność wyników z wizualizacją PCA. Wybór odpowiedniej liczby klastrów jest niezbędny, ponieważ wybór metryki odległości i algorytmu klasteryzacji (np. K-means czy metody hierarchiczne) zależy od specyfiki analizowanego problemu oraz charakterystyki danych.

Klasteryzacja hierarchiczna została przeprowadzona przy użyciu różnych metod łączenia, takich jak Ward, single oraz complete linkage. Porównanie uzyskanych dendrogramów pokazało, że metoda łączenia znacząco wpływa na strukturę grupowania. Redukcja wymiarowości przed zastosowaniem klasteryzacji hierarchicznej dodatkowo ułatwia interpretację wyników, umożliwiając wyodrębnienie klastrów na różnych poziomach oraz porównanie ich z wynikami metody K-means.

Podsumowując, integracja PCA z metodami klasteryzacji stanowi solidne narzędzie do wykrywania naturalnych skupień w danych. Dobór odpowiedniej metody klasteryzacji oraz metryk odległości jest kluczowy i powinien być dostosowany do specyfiki problemu, co pozwala na trafniejsze określenie liczby klastrów i lepsze zrozumienie struktury danych.