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 drugi (2)	Bartosz Bieniek Informatyka II stopień, stacjonarne, 1 semestr, gr.A

1. Polecenie:

1. Określenie czy dane nadają się do analizy skupień (klasteryzacji).

2. Metoda PCA

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

3. Metody niehierarchiczne

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

4. Metody hierarchiczne

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

2. Opis programu opracowanego [Kod źródłowy Github]

Weryfikacja danych

Przed przystąpieniem do wykonywania analizy skupień warto upewnić się, że przygotowane dane nadają się do wykonania tego procesu. W celu sprawdzenia, czy informacje pozyskane z pliku data.csv nadają się do analizy skupień (klasteryzacji), należy spełnić podstawowe wymagania klasteryzacji:

a) Odpowiednie zmienne: Metody klasteryzacji wymagają danych numerycznych. Należy więc sprawdzić, czy odpowiednie kolumny zawierają dane liczbowe (np. age, income, outcome, savings, credit_score, spending_score).
b) Brak wartości brakujących: Dane nie mogą zawierać braków, chyba że zostaną one odpowiednio obsłużone (np. w wyniku imputacji).
c) Normalizacja skal danych: Metody takie jak PCA lub k-means wymagają danych w podobnych skalach.
d) Wystarczająca liczba obserwacji: Dane muszą zawierać wystarczającą liczbę rekordów, aby analiza klasteryzacji miała sens.

```python
# 1. Sprawdzenie czy dane nadają się do analizy skupień (klasteryzacji)

import pandas as pd
import numpy as np

file_path = "data.csv"
data = pd.read_csv(file_path)

def check_data_for_clustering(data):
    results = {}

    numeric_cols = data.select_dtypes(include=[np.number]).columns.tolist() # Kolumny numeryczne
    results['numeric_columns'] = numeric_cols

    missing_values = data.isnull().sum().sum() # Sprawdzenie brakujących wartości
    results['missing_values'] = missing_values

    non_numeric_cols = [col for col in data.columns if col not in numeric_cols] # Kolumny nienumeryczne (sprawdzanie unikalnych wartości)
    unique_values = {col: data[col].nunique() for col in non_numeric_cols}
    results['unique_values_in_non_numeric'] = unique_values

    ranges = {col: (data[col].min(), data[col].max()) for col in numeric_cols} # Zakresy numeryczne
    results['numeric_column_ranges'] = ranges

    results['num_observations'] = len(data) # Liczba obserwacji
    results['scaling_needed'] = True # Wymagana normalizacja

    return results

results = check_data_for_clustering(data)

# Wyświetanie wyników
for key, value in results.items():
    print(f"{key}: {value}")
```

Rys. 1. Sprawdzanie danych przed przystąpieniem do analiz.

W zaprezentowanym na powyższym rysunku programie załadowano dane z pliku csv. Były one wykorzystywane w każdej z trzech metod. Następnie zdefiniowano funkcję chack_data_fol_clustering(), która sprawdza czy kolumny są liczbowe, uzupełnia braki w danych, sprawdza unikalność w kolumnach nienumerycznych, definiuje zakresy liczbowe w kolumnach numerycznych i upewnia się, że jest wystarczająco dużo danych obserwacyjnych dodatkowo ustawiając flagę wymaganej normalizacji na True.

```
numeric_columns: ['id', 'age', 'income', 'outcome', 'savings', 'children', 'credit_score', 'spending_score']
missing_values: 0
unique_values_in_non_numeric: {'name': 100, 'employment_status': 4, 'city': 13}
numeric_column_ranges: {'id': (1, 100), 'age': (18, 80), 'income': (2479.6, 28730.73), 'outcome': (508.03, 3410.55), 'savings': (42.28, 640217.0), 'children': (0, 5), 'credit_score': (320.18, 842.98), 'spending_score': (2.4, 99.74)}
num_observations: 100
scaling_needed: True
```

Rys. 2. Wyniki sprawdzenia danych.

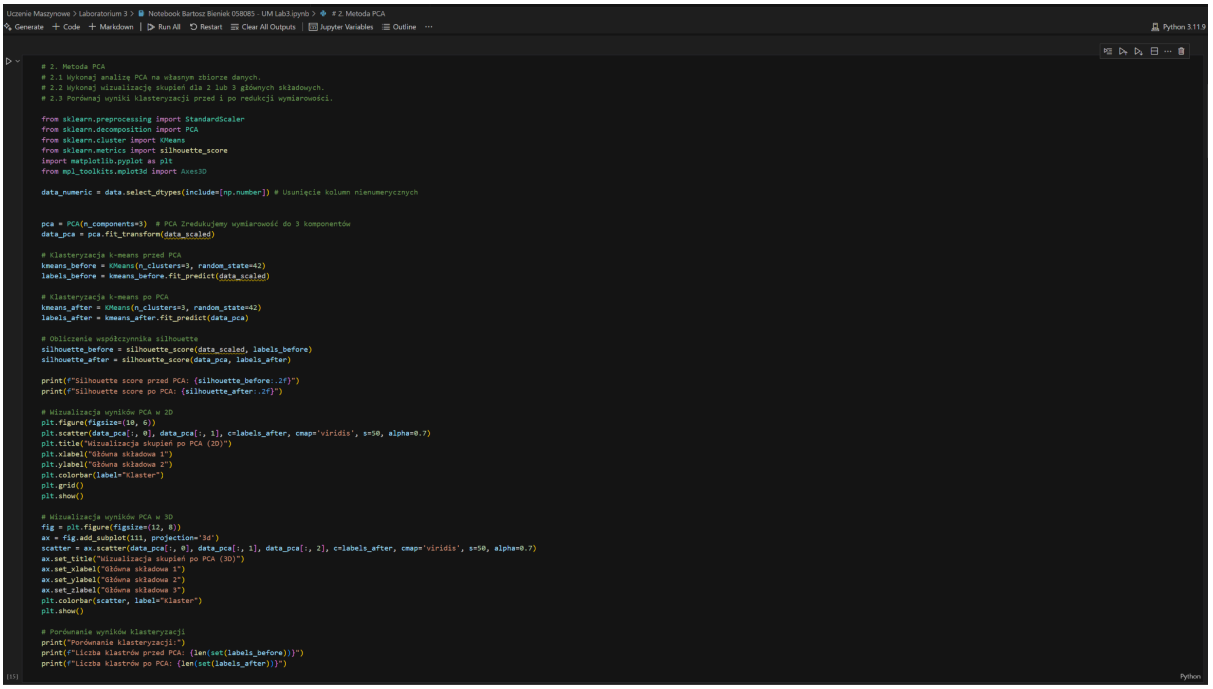
Na podstawie przedstawionych na rysunku drugim wyników można w dużej mierze ocenić, że dane nadają się do analizy klasteryzacj, ze względu na:

a) Kolumny numeryczne: Jest wiele kolumn numerycznych, które mogą być użyte do klasteryzacji: age, income, outcome, savings, credit_score, spending_score. Te dane mogą stanowić podstawę informacji dla analizy skupień.
b) Brak braków danych: Wartość missing_values wynosi 0, co oznacza, że dane nie są wybrakowane.
c) Zakresy kolumn numerycznych: Kolumny numeryczne mają zróżnicowane zakresy, a to wskazuje na potrzebę normalizacji.
d) Liczba obserwacji: num_observations wynosi 100, czyli jest to ilość wystarczająca dla większości podstawowych analiz klasteryzacji.

Dane idealnie dopasowywują się do wymagań zadania, ze względu na:

a) Obecność kolumn nienumerycznych: Kolumny name, city oraz employment_status są nienumeryczne. Employment_status i city mogą być użyte w klasteryzacji, ale wymagają zakodowania lub numeracji.
b) Rozkład danych: Skrajnie rozkłady, takie jak silna skośność w savings, mogą wpływać na wyniki klasteryzacji.
c) Dominacja jednej zmiennej: Savings ma znacznie szerszy zakres (do 640217.0), sugerujący powstanie dominowacji ponad inne zmienne. Normalizacja (standaryzacja) jest dla tych danych konieczna.

Metoda PCA



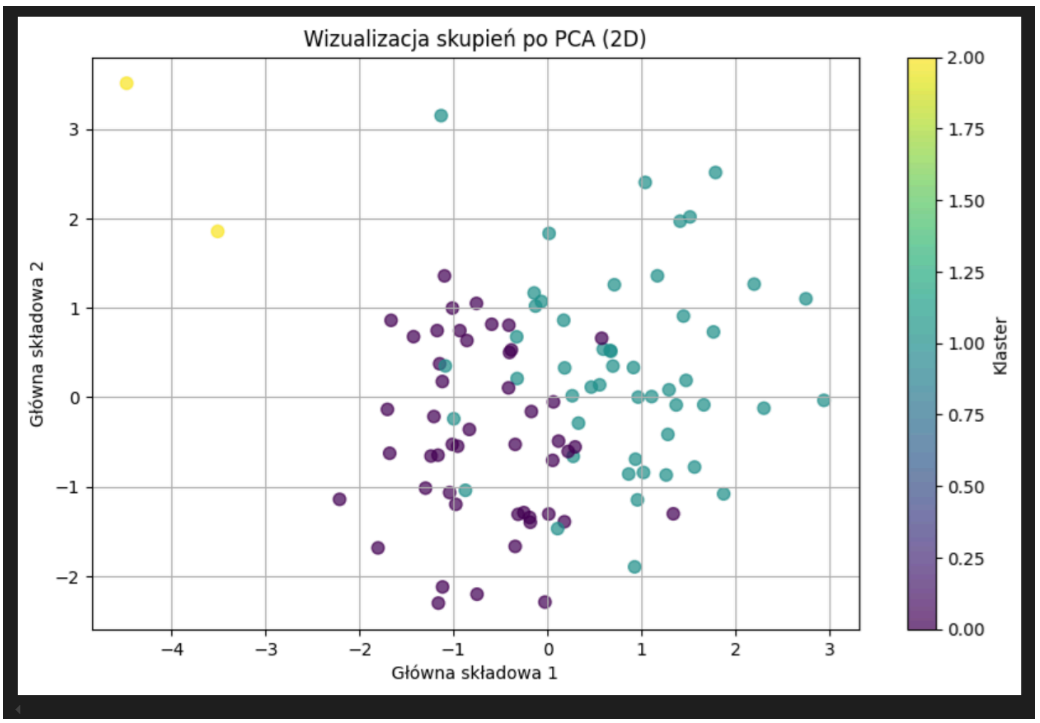
```
# 2. Metoda PCA
# 2.1 Wykonaj analizę PCA na własnym zbiorze danych.
# 2.2 Wykonaj wizualizację skupień dla 2 lub 3 głównych składowych.
# 2.3 Porównaj wyniki klasteryzacji przed i po redukcji wymiarowości.

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

data_numeric = data.select_dtypes(include=[np.number]) # Usunięcie kolumn nienumerycznych


pca = PCA(n_components=3)  # PCA Zredukujemy wymiarowość do 3 komponentów
data_pca = pca.fit_transform(data_scaled)

# Klasteryzacja k-means przed PCA
kmeans_before = KMeans(n_clusters=3, random_state=42)
labels_before = kmeans_before.fit_predict(data_scaled)

# Klasteryzacja k-means po PCA
kmeans_after = KMeans(n_clusters=3, random_state=42)
labels_after = kmeans_after.fit_predict(data_pca)

# Obliczenia współczynnika silhouette
silhouette_before = silhouette_score(data_scaled, labels_before)
silhouette_after = silhouette_score(data_pca, labels_after)

print(f"Silhouette score przed PCA: {silhouette_before:.2f}")
print(f"Silhouette score po PCA: {silhouette_after:.2f}")

# Wizualizacja wyników PCA w 2D
plt.figure(figsize=(10, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=labels_after, cmap='viridis', s=50, alpha=0.7)
plt.title("Wizualizacja skupień po PCA (2D)")
plt.xlabel("Główna składowa 1")
plt.ylabel("Główna składowa 2")
plt.colorbar(label="Klaster")
plt.grid()
plt.show()

# Wizualizacja wyników PCA w 3D
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(data_pca[:, 0], data_pca[:, 1], data_pca[:, 2], c=labels_after, cmap='viridis', s=50, alpha=0.7)
ax.set_title("Wizualizacja skupień po PCA (3D)")
ax.set_xlabel("Główna składowa 1")
ax.set_ylabel("Główna składowa 2")
ax.set_zlabel("Główna składowa 3")
plt.colorbar(scatter, label="Klaster")
plt.show()

# Porównanie wyników klasteryzacji
print("Porównanie klasteryzacji:")
print(f"Liczba klastrów przed PCA: {len(set(labels_before))}")
print(f"Liczba klastrów po PCA: {len(set(labels_after))}")
```

Rys. 3. Program metody PCA.

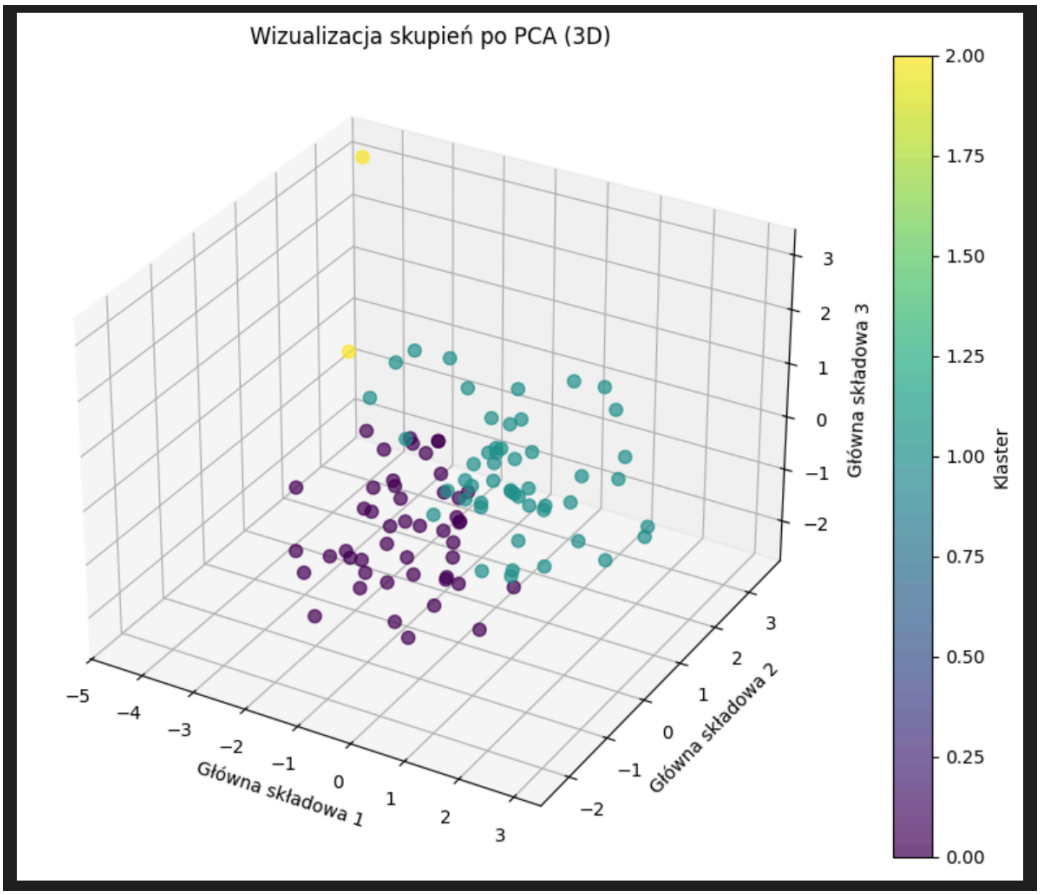
Kod przedstawiony na powyższym rysunku dokonuje redukcji wymiarowości danych za pomocą PCA, ograniczając je do trzech głównych składowych, aby uprościć analizę i umożliwić wizualizację w 3D. Przeprowadzono klasteryzację k-means zarówno na danych oryginalnych, jak i na danych po redukcji wymiarowości, co pozwoliło porównać jakość uzyskanych klastrów. Obliczono współczynniki silhouette przed i po PCA, oceniając spójność klastrów w obu przypadkach. Wyniki wskazały, jak redukcja wymiarowości wpływała na zdolność algorytmu do identyfikacji skupień. Wizualizację klastrów przeprowadzono w dwóch i trzech wymiarach, co ułatwiło interpretację wyników klasteryzacji. Na koniec porównano liczbę klastrów przed i po PCA, aby zaobserwować różnice w strukturze danych.

Silhouette score przed PCA: 0.12
Silhouette score po PCA: 0.26
Porównanie klasteryzacji:
Liczba klastrów przed PCA: 3
Liczba klastrów po PCA: 3

Uzyskane wyniki wykazały, że współczynnik silhouette po zastosowaniu PCA wzrósł z 0.12 do 0.26, co oznacza, że klastry stały się bardziej spójne i wyraźniejsze. Liczba klastrów przed i po redukcji wymiarowości pozostała taka sama (3), jednak lepsza wartość silhouette sugeruje, że PCA poprawiło jakość podziału danych na skupienia.



Rys. 4. Wizualizacja 2D.



Rys. 5. Wizualizacja 3D.

Metoda niehierarchiczna

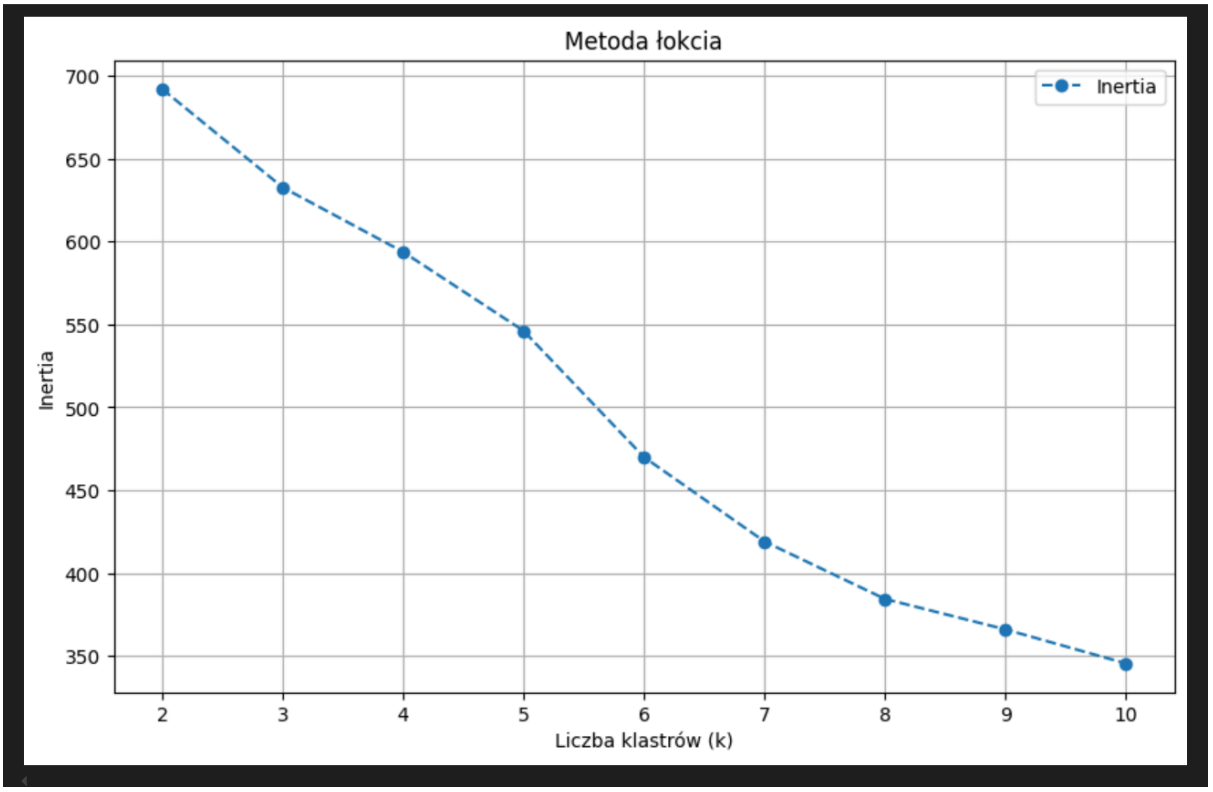
```
# 3.1 Klasteryzacja k-means dla różnych wartości k (metoda łokcia)
inertia = []
silhouette_scores = []
k_values = range(2, 11)  # Testujemy od 2 do 10 klastrów

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(data_scaled)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(data_scaled, labels))

# Wykres metody łokcia
plt.figure(figsize=(10, 6))
plt.plot(k_values, inertia, marker='o', linestyle='--', label='Inertia')
plt.title("Metoda łokcia")
plt.xlabel("Liczba klastrów (k)")
plt.ylabel("Inertia")
plt.xticks(k_values)
plt.legend()
plt.grid()
plt.show()

# Wykres współczynnika silhouette
plt.figure(figsize=(10, 6))
plt.plot(k_values, silhouette_scores, marker='o', linestyle='--', color='orange', label='Silhouette Score')
plt.title("Wartość silhouette dla różnych k")
plt.xlabel("Liczba klastrów (k)")
plt.ylabel("Silhouette Score")
plt.xticks(k_values)
plt.legend()
plt.grid()
plt.show()

# Wybór optymalnego k na podstawie metody łokcia (manualny wybór lub analiza wykresu)
optimal_k = 3  # Zakładamy, że wybraliśmy k=3 na podstawie wykresu

# 3.2 Klasteryzacja k-means dla optymalnego k
kmeans_optimal = KMeans(n_clusters=optimal_k, random_state=42)
labels_optimal = kmeans_optimal.fit_predict(data_scaled)

# Porównanie wyników z wcześniejszym PCA
print(f"Silhouette Score dla optymalnego k={optimal_k}: {silhouette_score(data_scaled, labels_optimal):.2f}")

pca = PCA(n_components=2) # Redukcja do 2 komponentów
data_pca = pca.fit_transform(data_scaled)

kmeans_pca = KMeans(n_clusters=optimal_k, random_state=42) # Klasteryzacja po PCA
labels_pca = kmeans_pca.fit_predict(data_pca)

silhouette_pca = silhouette_score(data_pca, labels_pca) # Silhouette po PCA
print(f"Silhouette Score po PCA (k={optimal_k}): {silhouette_pca:.2f}")

# Wizualizacja klasteryzacji na zredukowanych wymiarach (2D)
plt.figure(figsize=(10, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=labels_pca, cmap='viridis', s=50, alpha=0.7)
plt.title(f"Wizualizacja skupień po PCA (k={optimal_k})")
plt.xlabel("Główna składowa 1")
plt.ylabel("Główna składowa 2")
plt.colorbar(label="Klaster")
plt.grid()
plt.show()
```

Rys. 6. Program metody hierarchicznej.

Wykonano analizę klasteryzacji przy użyciu metody k-means dla różnych wartości liczby klastrów kkk, testując zakres od 2 do 10 klastrów. Dla każdej wartości kkk obliczono inercję (sumę odległości punktów od centroidów klastrów) oraz współczynnik silhouette, oceniając jakość podziału danych. Na podstawie tych wyników wygenerowano dwa wykresy: metodę łokcia, ukazującą zmianę inercji w zależności od liczby klastrów, oraz wykres współczynnika silhouette, pomagający w wyborze optymalnego kkk.

Wybrano optymalną liczbę klastrów k=3, co uzasadniono analizą wykresu metody łokcia oraz wartości silhouette. Następnie przeprowadzono klasteryzację k-means na danych przeskalowanych dla k=3k = 3k=3 i obliczono współczynnik silhouette, który wyniósł 0.26. Dodatkowo, dokonano redukcji wymiarowości danych do dwóch głównych składowych za pomocą PCA, co pozwoliło na przeprowadzenie klasteryzacji i wizualizację wyników w 2D.

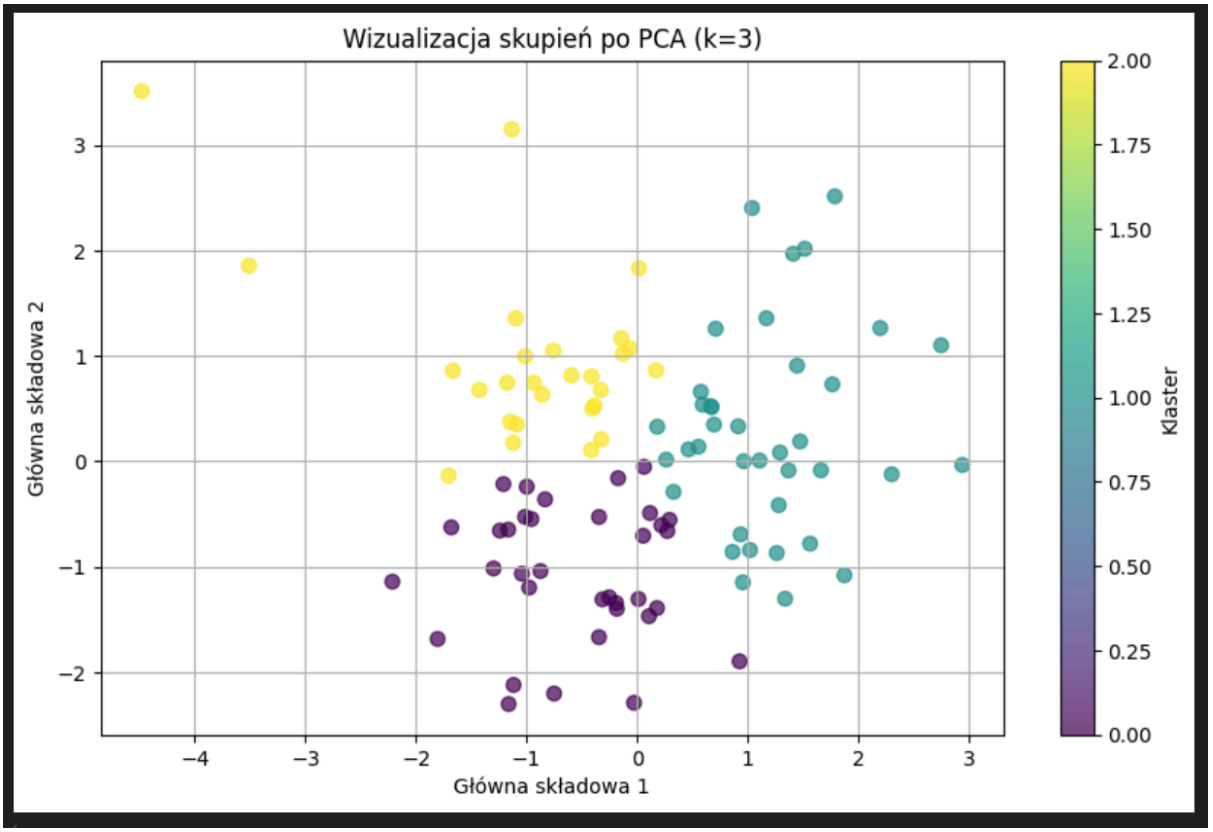
Silhouette score po PCA również obliczono, co umożliwiło porównanie jakości klasteryzacji przed i po redukcji wymiarowości. Wyniki przedstawiono na wykresie, ilustrując rozkład klastrów w przestrzeni dwóch głównych składowych, co ułatwiło interpretację i wizualizację struktury danych.



Rys. 7. Wykres zależności Inertii od liczby klastrów dla metody łokcia.

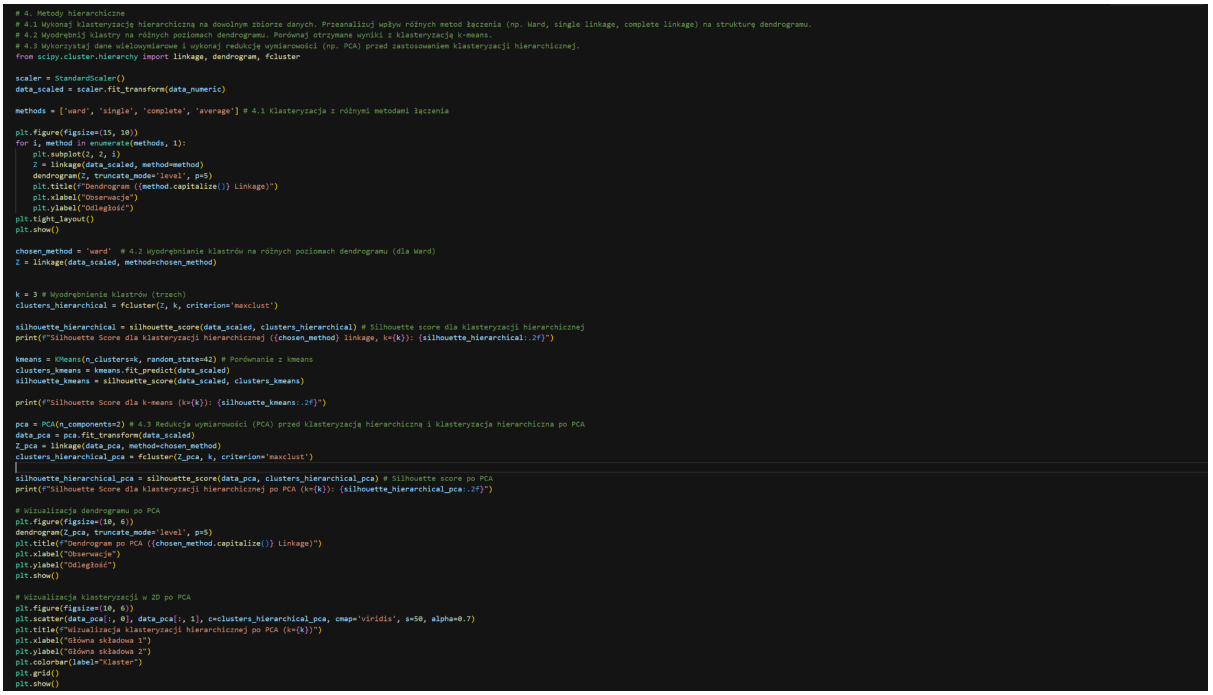
Rys. 8. Wykres zależności Silhouette Score od liczby klastrów.

Silhouette Score dla optymalnego k=3: 0.12
Silhouette Score po PCA (k=3): 0.36



Rys. 9. Wizualizacja skupień po PCA dla liczby klastrów równej 3.

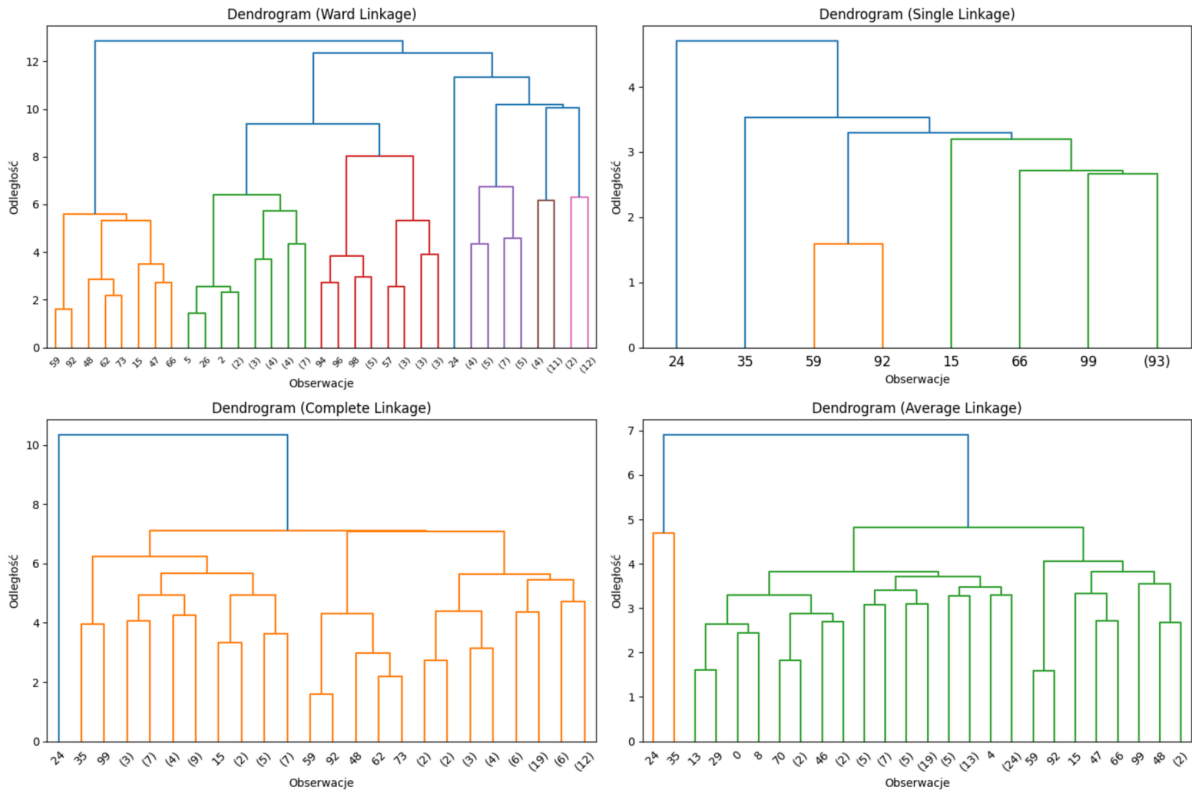
Metoda hierarchiczna



```python
# 4. Metody hierarchiczne
# 4.1 Wykonaj klasteryzację hierarchiczną na dowolnym zbiorze danych. Przeanalizuj wpływ różnych metod łączenia (np. Ward, single linkage, complete linkage) na strukturę dendrogramu.
# 4.2 Wyodrębnij klastry na różnych poziomach dendrogramu. Porównaj otrzymane wyniki z klasteryzacją k-means.
# 4.3 Wykorzystaj dane wielowymiarowe i wykonaj redukcję wymiarowości (np. PCA) przed zastosowaniem klasteryzacji hierarchicznej.
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_numeric)

methods = ['ward', 'single', 'complete', 'average'] # 4.1 Klasteryzacja z różnymi metodami łączenia

plt.figure(figsize=(15, 10))
for i, method in enumerate(methods, 1):
    plt.subplot(2, 2, i)
    Z = linkage(data_scaled, method=method)
    dendrogram(Z, truncate_mode='level', p=5)
    plt.title(f"Dendrogram ({method.capitalize()} Linkage)")
    plt.xlabel("Obserwacje")
    plt.ylabel("Odległość")
plt.tight_layout()
plt.show()

chosen_method = 'ward'  # 4.2 Wyodrębnianie klastrów na różnych poziomach dendrogramu (dla Ward)
Z = linkage(data_scaled, method=chosen_method)


k = 3 # Wyodrębnienie klastrów (trzech)
clusters_hierarchical = fcluster(Z, k, criterion='maxclust')

silhouette_hierarchical = silhouette_score(data_scaled, clusters_hierarchical) # Silhouette score dla klasteryzacji hierarchicznej
print(f"Silhouette Score dla klasteryzacji hierarchicznej ({chosen_method} linkage, k={k}): {silhouette_hierarchical:.2f}")

kmeans = KMeans(n_clusters=k, random_state=42) # Porównanie z kmeans
clusters_kmeans = kmeans.fit_predict(data_scaled)
silhouette_kmeans = silhouette_score(data_scaled, clusters_kmeans)

print(f"Silhouette Score dla k-means (k={k}): {silhouette_kmeans:.2f}")

pca = PCA(n_components=2) # 4.3 Redukcja wymiarowości (PCA) przed klasteryzacją hierarchiczną i klasteryzacja hierarchiczna po PCA
data_pca = pca.fit_transform(data_scaled)
Z_pca = linkage(data_pca, method=chosen_method)
clusters_hierarchical_pca = fcluster(Z_pca, k, criterion='maxclust')

silhouette_hierarchical_pca = silhouette_score(data_pca, clusters_hierarchical_pca) # Silhouette score po PCA
print(f"Silhouette Score dla klasteryzacji hierarchicznej po PCA (k={k}): {silhouette_hierarchical_pca:.2f}")

# Wizualizacja dendrogramu po PCA
plt.figure(figsize=(10, 6))
dendrogram(Z_pca, truncate_mode='level', p=5)
plt.title(f"Dendrogram po PCA ({chosen_method.capitalize()} Linkage)")
plt.xlabel("Obserwacje")
plt.ylabel("Odległość")
plt.show()

# Wizualizacja klasteryzacji w 2D po PCA
plt.figure(figsize=(10, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=clusters_hierarchical_pca, cmap='viridis', s=50, alpha=0.7)
plt.title(f"Wizualizacja klasteryzacji hierarchicznej po PCA (k={k})")
plt.xlabel("Główna składowa 1")
plt.ylabel("Główna składowa 2")
plt.colorbar(label="Klaster")
plt.grid()
plt.show()
```

Rys. 10. Program metody hierarchicznej

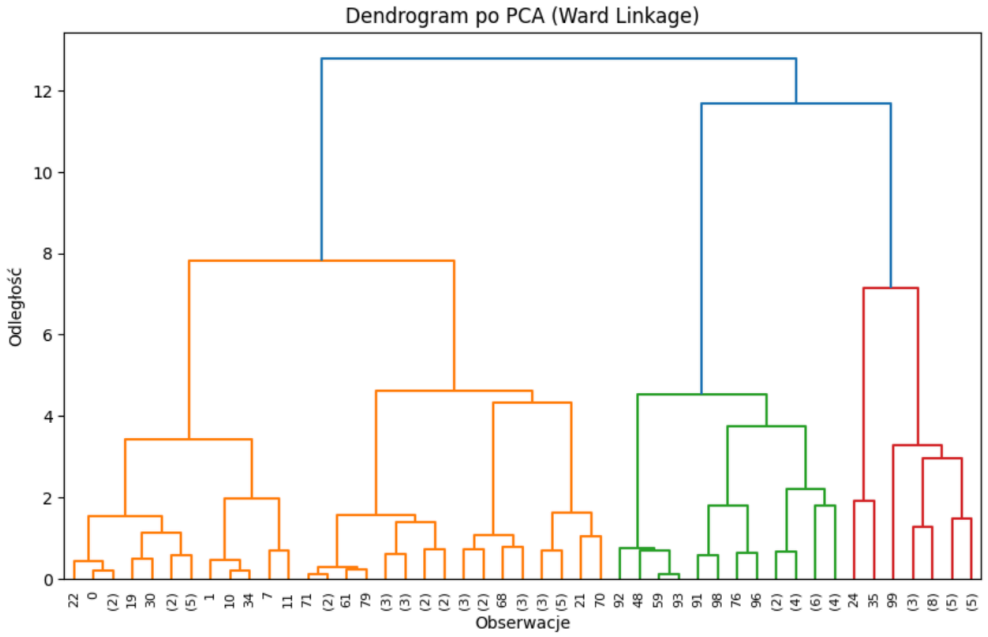
Wykonano klasteryzację hierarchiczną na standaryzowanych danych, analizując wpływ różnych metod łączenia (Ward, single linkage, complete linkage, average linkage) na strukturę dendrogramu. Dla każdej metody wygenerowano osobny dendrogram, przedstawiający hierarchię grupowania. Różnice między metodami uwidoczniły się w kształcie dendrogramów, co pozwoliło zaobserwować, jak wybór sposobu łączenia wpływa na ostateczną strukturę klastrów.



Rys. 11. Dendogamy różnych metod łączenia.

Silhouette Score dla klasteryzacji hierarchicznej (ward linkage, k=3): 0.11
Silhouette Score dla k-means (k=3): 0.12
Silhouette Score dla klasteryzacji hierarchicznej po PCA (k=3): 0.33

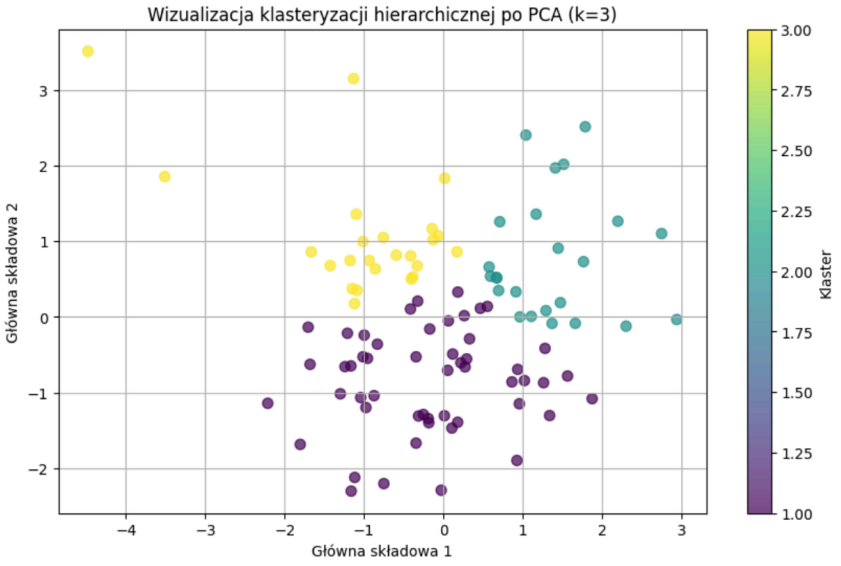
W kolejnym kroku wyodrębniono klastry z dendrogramu, wybierając metodę Ward oraz ustawiając liczbę klastrów na k=3k = 3k=3. Obliczono współczynnik silhouette dla tej klasteryzacji, uzyskując wartość 0.11. Wyniki te porównano z klasteryzacją k-means dla tej samej liczby klastrów, gdzie silhouette score wyniósł 0.12. Porównanie to pozwoliło ocenić, jak różne podejścia do grupowania radzą sobie z tym samym zestawem danych.



Rys. 12. Dendrogram po PCA (Ward Linkage)

Przeprowadzono również redukcję wymiarowości za pomocą PCA, upraszczając dane do dwóch głównych składowych przed zastosowaniem klasteryzacji hierarchicznej. Następnie wygenerowano dendrogram dla zredukowanych danych oraz wyodrębniono klastry. Współczynnik silhouette po redukcji wymiarowości osiągnął wartość 0.33, co wskazuje, że zastosowanie PCA mogło poprawić jakość klasteryzacji.

Na zakończenie wizualizowano wyniki klasteryzacji w przestrzeni dwóch głównych składowych, prezentując rozkład klastrów w 2D. Taka wizualizacja umożliwiła łatwiejszą interpretację struktury danych i ich podziału na grupy.



Rys. 13. Wizualizacja klasteryzacji hierarchicznej po PCA (k=3)

3. Wnioski

Różne metody łączenia w klasteryzacji hierarchicznej znacząco wpływały na kształt dendrogramu i sposób grupowania danych. W szczególności metoda Ward pozwoliła na uzyskanie bardziej spójnych klastrów, co znalazło odzwierciedlenie w wyższej wartości współczynnika silhouette w porównaniu z innymi metodami.

Algorytm k-means okazał się skuteczniejszy na danych w oryginalnej przestrzeni, osiągając wyższy współczynnik silhouette niż klasteryzacja hierarchiczna. Jednak po zastosowaniu PCA klasteryzacja hierarchiczna poprawiła swoje wyniki, co podkreśla znaczenie redukcji wymiarowości w analizie wielowymiarowych danych.