SPRAWOZDANIE

Zajęcia: Nauka o danych 2

Prowadzący: prof. dr hab. inż. Vasyl Martsenyuk

Laboratorium Nr 1 **Data** 16.10.2025 **Temat:** "Uczenie maszynowe w praktyce: Zaawansowane techniki ensemble learning" **Wariant** 10	Jakub Janik Informatyka II stopień, stacjonarne, 2 semestr, gr. CB

1. Cel ćwiczenia:

Celem ćwiczenia była praktyczna eksploracja i analiza skupień (ang. *cluster analysis*) na zbiorze danych rzeczywistych. Zadanie polegało na zastosowaniu i porównaniu trzech głównych algorytmów grupowania oraz na ocenie jakości uzyskanego podziału za pomocą metryk wewnętrznych i zewnętrznych.

Wybrany wariant: 10. World Happiness.

Charakterystyka danych: Zbiór zawiera 156 obserwacji (krajów) z 6 cechami numerycznymi

Miara podobieństwa: W algorytmach K-means i DBSCAN zastosowano domyślnie **odległość Euklidesową**

2. Przebieg ćwiczenia:

Metodologia i Algorytmy

Do grupowania zastosowano trzy główne algorytmy: K-means (centroidowy), DBSCAN (oparty na gęstości) i Agglomerative Clustering (hierarchiczny)

K-means

Algorytm ma na celu minimalizację wariancji wewnątrz skupień. Wymaga z góry zdefiniowanej liczby skupień (k)

DBSCAN

Algorytm identyfikujący skupienia jako gęsto zaludnione regiony, radzący sobie z szumem. Wymaga parametrów eps (promień sąsiedztwa) i min_samples (minimalna liczba punktów rdzeniowych).

Agglomerative Clustering

Metoda hierarchiczna, która łączy punkty w coraz większe grupy, aż do uzyskania zadanej liczby skupień.

Kod źródłowy:

```
import kagglehub
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import glob
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import silhouette_score, calinski_harabasz_score, da-
vies_bouldin_score
from scipy.cluster.hierarchy import dendrogram, linkage
import warnings

warnings.filterwarnings("ignore")

# ========================================
# 2. POBIERANIE I WCZYTANIE DANYCH
# ========================================
print("Pobieranie danych z Kaggle...")
try:
    path = kagglehub.dataset_download("unsdsn/world-happiness")

    # Szukamy wszystkich plików .csv w pobranym folderze
    all_files = glob.glob(os.path.join(path, "*.csv"))

    if all_files:
        latest_file = sorted(all_files)[-1]
        df = pd.read_csv(latest_file)
    else:
        raise FileNotFoundError("Nie znaleziono plików CSV w pobranym zbio-
rze.")

    # Wybór kolumn numerycznych
    possible_cols = ['GDP per capita', 'Social support', 'Healthy life expec-
tancy',
                     'Freedom to make life choices', 'Generosity', 'Percep-
tions of corruption']

    X_raw = df[possible_cols] if set(possible_cols).issubset(df.columns) else
df.select_dtypes(include=[np.number])
```

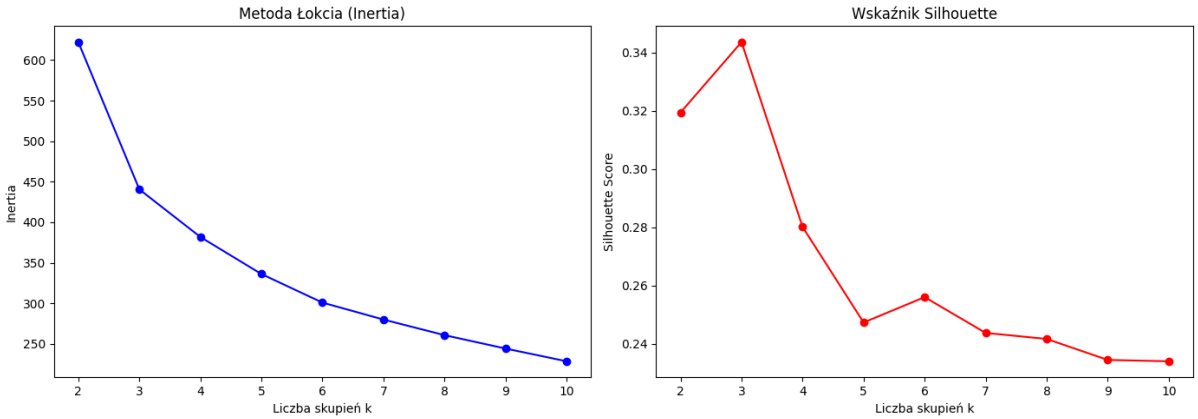
```
    X_raw = X_raw.dropna()

    # Standaryzacja
    scaler = StandardScaler()
    X = scaler.fit_transform(X_raw)
    print(f"Dane przygotowane. Liczba krajów: {X.shape[0]}, Liczba cech:
{X.shape[1]}")

except Exception as e:
    print(f"WYSTĄPIŁ BŁĄD: {e}")
    X = None

if X is not None:
    # ========================================
    # 3. K-MEANS: METODA ŁOKCIA I SILHOUETTE
    # ========================================
    inertia = []
    sil_scores = []
    k_range = range(2, 11)

    for k in k_range:
        km = KMeans(n_clusters=k, random_state=42, n_init=10)
        labels = km.fit_predict(X)
        inertia.append(km.inertia_)
        sil_scores.append(silhouette_score(X, labels))

    # Wykresy
    fig, ax = plt.subplots(1, 2, figsize=(14, 5))
    ax[0].plot(k_range, inertia, 'bo-')
    ax[0].set_title('Metoda łokcia (Inertia)')
    ax[0].set_xlabel('Liczba skupień k')
    ax[0].set_ylabel('Inertia')

    ax[1].plot(k_range, sil_scores, 'ro-')
    ax[1].set_title('Wskaźnik Silhouette')
    ax[1].set_xlabel('Liczba skupień k')
    ax[1].set_ylabel('Silhouette Score')
    plt.tight_layout()
    plt.show()

    # Finalny model K-means (zakładamy k=3 jako optymalne)
    best_k = 3
    kmeans_model = KMeans(n_clusters=best_k, random_state=42, n_init=10)
    y_kmeans = kmeans_model.fit_predict(X)

    # ========================================
    # 4. DBSCAN
    # ========================================
```

```
    print("\n--- Testowanie parametrów DBSCAN ---")
    for eps in [0.5, 1.0, 1.5]:
        for min_samples in [3, 5]:
            db = DBSCAN(eps=eps, min_samples=min_samples)
            labels_db = db.fit_predict(X)
            unique = set(labels_db)
            if len(unique) > 1:
                score = silhouette_score(X, labels_db)
                n_clusters = len(unique) - (1 if -1 in labels_db else 0)
                print(f"eps={eps}, min_samples={min_samples} -> Skupień:
{n_clusters}, Silhouette: {score:.3f}")

    # ==========================================
    # 5. METODA HIERARCHICZNA
    # ==========================================
    plt.figure(figsize=(10, 5))
    linked = linkage(X, method='ward')
    dendrogram(linked, truncate_mode='lastp', p=30)
    plt.title('Dendrogram (Metoda Warda)')
    plt.show()

    agg_model = AgglomerativeClustering(n_clusters=best_k)
    y_agg = agg_model.fit_predict(X)

    # ==========================================
    # 6. OCENA I PORÓWNANIE
    # ==========================================
    metrics_data = []
    for name, labels in [('K-means', y_kmeans), ('Agglomerative', y_agg)]:
        metrics_data.append({
            'Model': name,
            'Calinski-Harabasz': calinski_harabasz_score(X, labels),
            'Davies-Bouldin': davies_bouldin_score(X, labels),
            'Silhouette': silhouette_score(X, labels)
        })

    print("\n--- Tabela metryk porównawczych ---")
    results_df = pd.DataFrame(metrics_data)
    print(results_df)

    # ==========================================
    # 7. WIZUALIZACJA PCA (2D)
    # ==========================================
    pca = PCA(n_components=2)
    X_pca = pca.fit_transform(X)

    plt.figure(figsize=(14, 6))
    plt.subplot(1, 2, 1)
```

```
    plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_kmeans, cmap='viridis',
edgecolor='k', s=50)
    plt.title(f'K-means (k={best_k})')

    plt.subplot(1, 2, 2)
    plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_agg, cmap='plasma',
edgecolor='k', s=50)
    plt.title(f'Hierarchiczne (k={best_k})')
    plt.show()
```

Wyniki:

Dobór liczby skupień k (K-means) [Zadanie 1]



Wartość k została oszacowana na podstawie Metody Łokcia (analiza spadku Inertii) oraz Wskaźnika Silhouette. Wybrano k=3, jako kompromis między spadkiem sumy kwadratów błędów (W(k)) a ogólną jakością separacji.

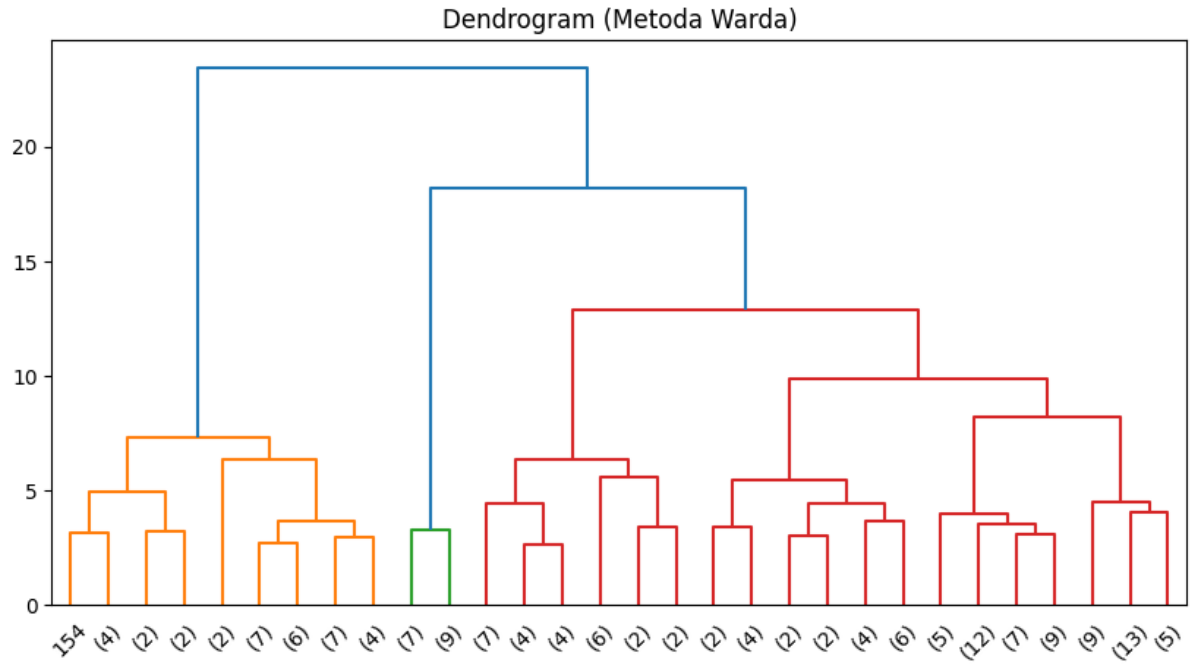
- Wskaźnik Silhouette dla k=3 (K-means): 0.3436
    - Interpretacja: Wartość ta jest umiarkowana (daleka od 1), co potwierdza, że grupy krajów się częściowo nakładają (skupienia nie są idealnie spójne i odseparowane)

Testowanie parametrów DBSCAN [Zadanie 2]

Przetestowano różne kombinacje parametrów, aby znaleźć optymalny podział

Parametry (eps, min_samples)		Liczba skupień	Silhouette Score
0.5	3	2	0.199
1.0	3	4	0.074
1.5	3	1	0.279

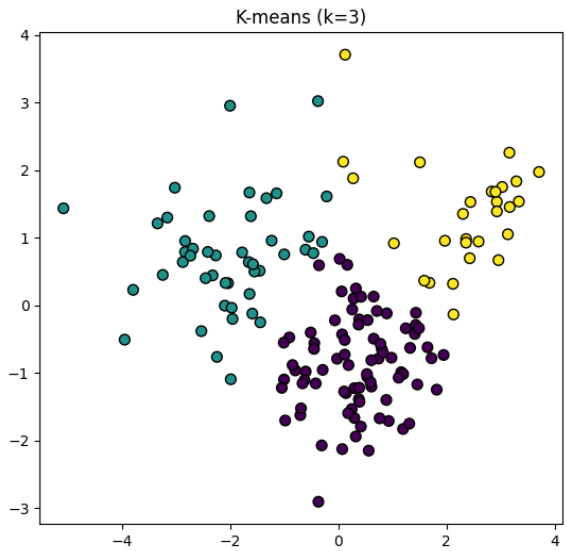
Wizualizacja Hierarchiczna

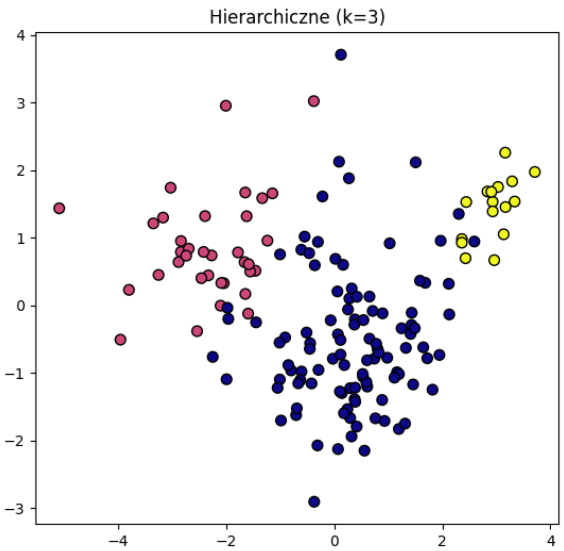


Analiza Dendrogramu dla Agglomerative Clustering (Metoda Warda) wizualnie potwierdziła, że największe zróżnicowanie w wysokości połączeń (odległości) występuje na poziomie dzielącym zbiór na 4 główne grupy

Porównanie i Ocena Jakości

Model	Calinski-Harabasz	Davies-Bouldin	Silhouette
K-means	86.0226	1.0703	0.3436
Agglomerative	68.2269	1.0019	0.3104





Link do repozytorium:https://github.com/Uczelniane/NODII.git

3. Wnioski

- Najlepszy algorytm: W oparciu o metryki CH i Silhouette, K-means jest najbardziej efektywną metodą do partycjonowania tego zbioru danych na k=3 grupy
- Jakość grupowania: Ogólna jakość grupowania (wskaźnik Silhouette ok. 0.34) jest umiarkowana, co sugeruje, że podział na 3 wyraźne i idealnie odseparowane grupy krajów jest trudny, a granice między nimi są płynne
- DBSCAN okazał się nieodpowiedni dla tego zbioru danych, wskazując na brak wyraźnych regionów wysokiej gęstości w przestrzeni cech