SPRAWOZDANIE

Zajęcia: Nauka o danych I
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 8 Data 11.01.2025 Temat: Praktyczne zastosowanie analizy skupień (clustering) do zbiorów danych Wariant drugi (13)	Bartosz Bieniek Informatyka II stopień, stacjonarne, 1 semestr, gr.A

1. Polecenie: Wariant trzynasty zadania

W Pythonie zastosuj Agglomerative Clustering na danych Iris i porównaj wyniki z k-means. Zwizualizuj wyniki na wykresie 2D.

2. Opis programu opracowanego [Kod źródłowy github.com/mindgoner]

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.decomposition import PCA

iris = load_iris()
X = iris.data
```

[1] ✓ 1.8s

Rys. 1. Wczytanie wymaganych bibliotek oraz załadowanie danych

Standardowym krokiem w wykonywaniu programów do analizy danych jest przygotowanie wymaganych bibliotek oraz załadowanie danych. Dane zostały załadowane do zmiennej “X”.

```
# 1. K-Means (analiza skupień)
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(X)
```

[2] ✓ 0.1s

Rys. 2. Analiza skupień K-means

Wykonano analizę skupień K-Means dla pobranych danych a następnie przypisano je do grup, stosując metodę fit_predict(). Wynik wywołania tej metody to grupy danych, których użyto w formie etykiet.

```
# 2. Agglomerative Clustering
agglo = AgglomerativeClustering(n_clusters=3)
agglo_labels = agglo.fit_predict(X)
```

[3] ✓ 0.0s

Rys. 3. Agglomerative Clustering

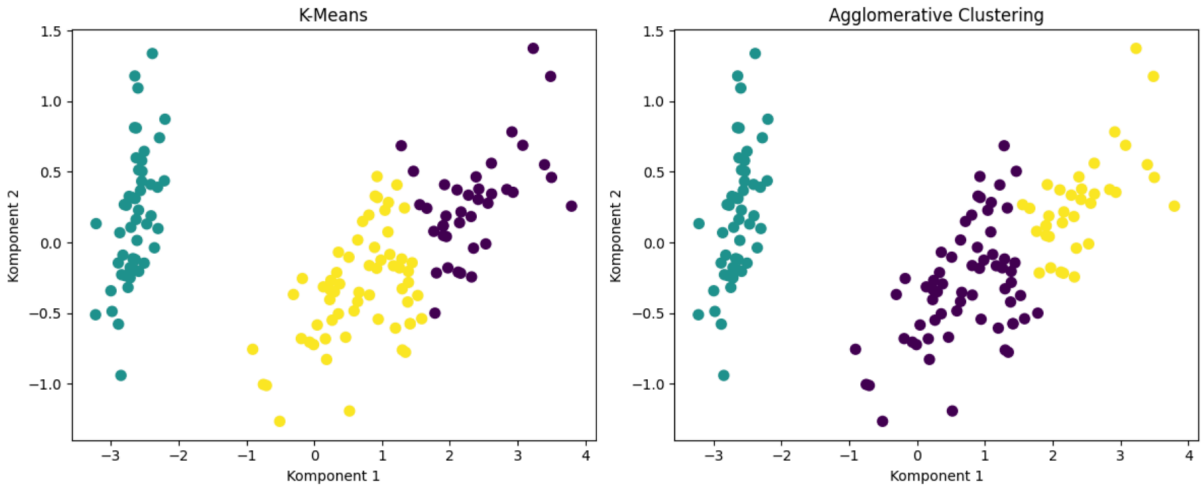
Metoda AC to metoda klasteryzacji danych, która podobnie jak K-Means zezwala na utworzenie etykiet, lecz zasada działania jest nieco inna. Na początku każdy punkt jest osobną grupą, czyli liczba punktów i grup jest taka sama. W kolejnym kroku dane znajdujace się najbliżej siebie łączone są w pary, trójki i coraz większe grupy. Proces trwa dopóki nie utworzy się tyle grup, ile wynosi wartość “n_clusters”. Wiadomo, że w danych Irys znajdują się 3 grupy irysów: Iris setosa, Iris versicolor oraz Iris virginica, stąd wartość n_components=3.

```
# 3. Porównanie AC and K-means
pca = PCA(n_components=2) # 2 wymiarowe dane
X_reduced = pca.fit_transform(X)

# KM
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=kmeans_labels, cmap='viridis', s=50)
plt.title('K-Means')
plt.xlabel('Komponent 1')
plt.ylabel('Komponent 2')

# AC
plt.subplot(1, 2, 2)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=agglo_labels, cmap='viridis', s=50)
plt.title('Agglomerative Clustering')
plt.xlabel('Komponent 1')
plt.ylabel('Komponent 2')

plt.tight_layout()
plt.show()
```

[8] ✓ 0.1s

Rys. 4. Spłaszczenie i wizualizacja danych.

W ostatnim kroku dane są spłaszczane do dwóch wymiarów przy pomocy PCA. Ostatnie dwa bloki stanowią wizualizacje danych przy pomocy biblioteki pyplot.



Rys. 4. Porównanie KMeans i AC.

Jak widać na powyższym rysunku, dane pogrupowane metodami KMeans i AC są identyczne. Oznacza to, że obydwie metody poprawnie klasteryzują dane.

3. Wnioski

W K-Means wybierana jest liczba grup na wstępie, a algorytm “szuka” najlepszych środków tych grup. W AC każdy punkt jest oddzielną grupą, a następnie następuje połączenie w celu zmniejszenia ilości grup.

AC pozwala na wydzielenie grup, jeżeli ich ilość jest nieznana.

Można używając odpowiednich argumentów dla metod określić “odległość” między grupami. Mogą to być najbliższe punkty, najdalsze lub środek grup.