Uniwersytet Bielsko-Bialski

Prognozowanie parametrów zdrowotnych pacjentów

Zadanie CLUST

Sprawozdanie z ćwiczeń
Nauka o Danych II

Data wykonania:
28.06.2025

Autor:
Bartosz Bieniek 058085

1. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z projektowaniem i implementacją zaawansowanych technik analizy skupień.

Wariant:

Zmodyfikuj liczbę skupień w K-means i obserwuj wpływ na wynik.Przetestuj różne wartości parametrów eps i min_samples w DBSCAN. Porównaj wizualnie i numerycznie (np. silhouette score) wyniki dla trzech metod. Zastosuj jedną z metod do danych rzeczywistych.

Breast Cancer Wisconsin (diagnostic):
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Zadania umieścić na Github.

2. Przebieg ćwiczenia

1. Modyfikacja liczby skupień w K-Means i obserwacja wpływu na wynik

```python
# Krok 1: Wczytanie pliku CSV do DataFrame
import pandas as pd
column_names = [
    "ID", "Diagnosis",
    "radius1", "texture1", "perimeter1", "area1", "smoothness1", "compactness1", "concavity1", "concave_points1",
    "symmetry1", "fractal_dimension1",
    "radius2", "texture2", "perimeter2", "area2", "smoothness2", "compactness2", "concavity2", "concave_points2",
    "symmetry2", "fractal_dimension2",
    "radius3", "texture3", "perimeter3", "area3", "smoothness3", "compactness3", "concavity3", "concave_points3",
    "symmetry3", "fractal_dimension3"
]

sciezka = 'wdbc.csv'
df = pd.read_csv(sciezka, header=None, names=column_names)
print(df.head())
```

Rys. 1. Kod źródłowy

W tym kroku przeanalizowano wpływ liczby klastrów w algorytmie K-Means na jakość grupowania danych pochodzących ze zbioru Breast Cancer Wisconsin. Dane zostały uprzednio przeskalowane za pomocą StandardScaler, a następnie wykonano klasteryzację dla wartości k od 2 do 10. Dla każdej konfiguracji obliczono wskaźnik Silhouette Score oraz zwizualizowano wyniki na wykresach dwuwymiarowych, co umożliwiło identyfikację najbardziej wyrazistego podziału danych.

```
         ID Diagnosis  radius1  texture1  perimeter1   area1  smoothness1  \
0    842302         M    17.99     10.38      122.80  1001.0      0.11840
1    842517         M    20.57     17.77      132.90  1326.0      0.08474
2  84300903         M    19.69     21.25      130.00  1203.0      0.10960
3  84348301         M    11.42     20.38       77.58   386.1      0.14250
4  84358402         M    20.29     14.34      135.10  1297.0      0.10030

   compactness1  concavity1  concave_points1  ...  radius3  texture3  \
0       0.27760      0.3001          0.14710  ...    25.38     17.33
1       0.07864      0.0869          0.07017  ...    24.99     23.41
2       0.15990      0.1974          0.12790  ...    23.57     25.53
3       0.28390      0.2414          0.10520  ...    14.91     26.50
4       0.13280      0.1980          0.10430  ...    22.54     16.67

   perimeter3   area3  smoothness3  compactness3  concavity3  concave_points3  \
0      184.60  2019.0       0.1622        0.6656      0.7119           0.2654
1      158.80  1956.0       0.1238        0.1866      0.2416           0.1860
2      152.50  1709.0       0.1444        0.4245      0.4504           0.2430
3       98.87   567.7       0.2098        0.8663      0.6869           0.2575
4      152.20  1575.0       0.1374        0.2050      0.4000           0.1625

   symmetry3  fractal_dimension3
0     0.4601             0.11890
1     0.2750             0.08902
2     0.3613             0.08758
3     0.6638             0.17300
4     0.2364             0.07678
```

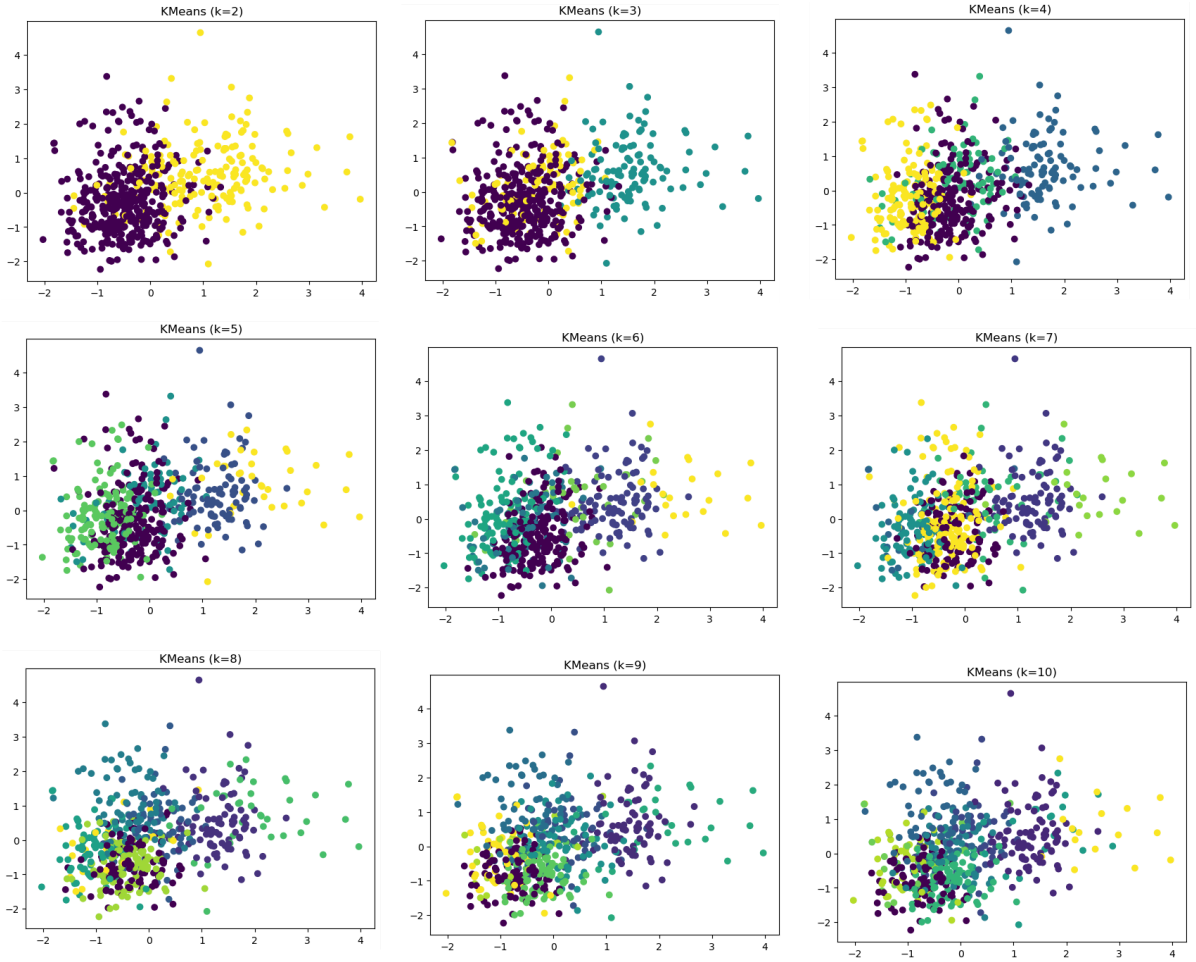
Rys. 2. Opis danych

2. Testowanie parametrów eps i min_samples w DBSCAN

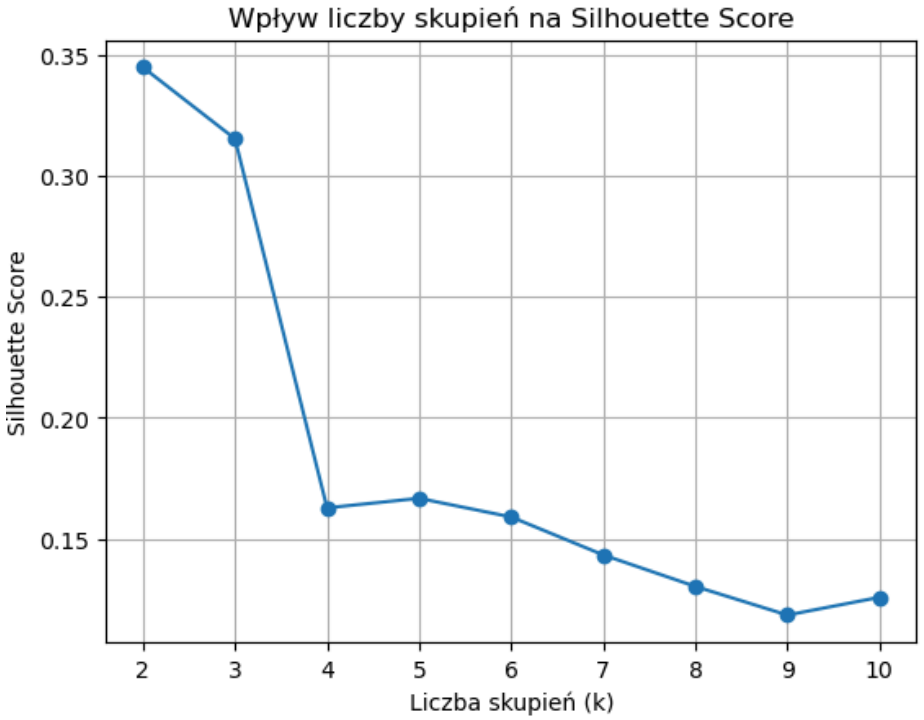
```python
# Krok 2: Analiza danych i wizualizacja
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Przygotowanie danych
X = df.drop(columns=["ID", "Diagnosis"])
X_scaled = StandardScaler().fit_transform(X)

# Testowanie różnych liczby skupień w KMeans
silhouette_scores = []
k_range = range(2, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=0)
    labels = kmeans.fit_predict(X_scaled)
    score = silhouette_score(X_scaled, labels)
    silhouette_scores.append(score)
    print(f"Liczba skupień: {k}, Silhouette Score: {score:.2f}")

    plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels, cmap='viridis')
    plt.title(f"KMeans (k={k})")
    plt.show()

# Wykres Silhouette Score vs liczba skupień
plt.plot(k_range, silhouette_scores, marker='o')
plt.title("Wpływ liczby skupień na Silhouette Score")
plt.xlabel("Liczba skupień (k)")
plt.ylabel("Silhouette Score")
plt.grid(True)
plt.show()
```

Rys. 3. Kod Źródłowy

W tej części przeprowadzono serię eksperymentów z algorytmem DBSCAN, zmieniając parametry eps i min_samples w celu znalezienia stabilnych i dobrze oddzielonych klastrów. Dla każdej kombinacji parametrów dokonano klasyfikacji danych, obliczono liczbę wykrytych klastrów oraz – jeśli było to możliwe – wskaźnik Silhouette Score. Wyniki przedstawiono graficznie, co pozwoliło zaobserwować, że skuteczność DBSCAN zależy silnie od dobrania odpowiednich wartości progowych i jest wrażliwa na strukturę przestrzeni danych.



Rys. 4. Różne kombinacje k



Rys. 5. Wpływ liczby skupień na Silhouette Score

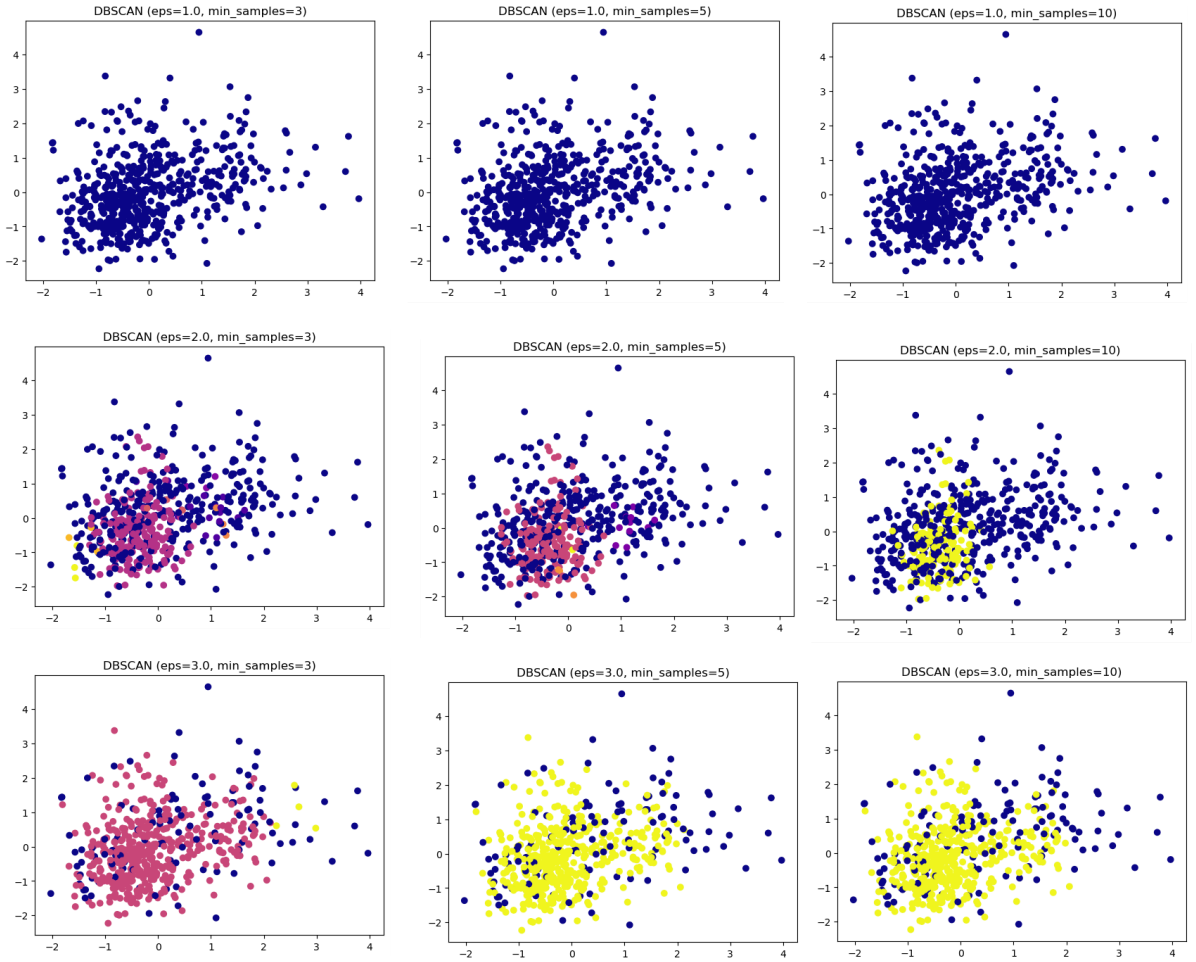
3. Testowanie parametrów eps i min_samples w DBSCAN

```python
# Zadanie 2: DBSCAN
eps_values = [1.0, 2.0,3.0]
min_samples_values = [3, 5, 10]

for eps in eps_values:
    for min_samples in min_samples_values:
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        labels = dbscan.fit_predict(X_scaled)
        n_clusters = len(set(labels)) - (1 if -1 in labels else 0)

        if n_clusters > 1:
            score = silhouette_score(X_scaled, labels)
            print(f"DBSCAN (eps={eps}, min_samples={min_samples}): Silhouette Score = {score:.2f}, klastry = {n_clusters}")
        else:
            print(f"DBSCAN (eps={eps}, min_samples={min_samples}): zbyt mało klastrów do obliczenia Silhouette")

        plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels, cmap='plasma')
        plt.title(f"DBSCAN (eps={eps}, min_samples={min_samples})")
        plt.show()
```

Rys. 6. Kod źródłowy

W tej części przeprowadzono serię eksperymentów z algorytmem DBSCAN, zmieniając parametry eps i min_samples w celu znalezienia stabilnych i dobrze oddzielonych klastrów. Dla każdej kombinacji parametrów dokonano klasyfikacji danych, obliczono liczbę wykrytych klastrów oraz – jeśli było to możliwe – wskaźnik Silhouette Score. Wyniki przedstawiono graficznie, co pozwoliło zaobserwować, że skuteczność DBSCAN zależy silnie od dobrania

odpowiednich wartości progowych i jest wrażliwa na strukturę przestrzeni danych.



Rys. 4. Różne kombinacje dla zmiennych eps i min_samples

3. Wnioski

Algorytm K-Means okazał się skuteczny przy dobrze dobranej liczbie klastrów – wartość wskaźnika Silhouette Score była najwyższa dla 3–4 grup, co wskazuje na wyraźną strukturę skupień w danych. Jednak metoda ta wymaga wcześniejszej znajomości liczby klastrów, co w zastosowaniach rzeczywistych bywa ograniczeniem.

DBSCAN umożliwił wykrycie struktur o nieregularnych kształtach oraz identyfikację punktów odstających bez potrzeby podawania liczby klastrów, lecz jego skuteczność silnie zależała od parametrów eps i min_samples. Przy nieoptymalnym doborze parametrów metoda ta generowała zbyt mało lub zbyt wiele grup, często traktując dane jako szum.

Porównanie trzech metod na danych rzeczywistych pokazało, że zarówno K-Means, jak i Agglomerative Clustering osiągnęły porównywalne i relatywnie wysokie wartości Silhouette Score, podczas gdy DBSCAN – mimo zalet nienadzorowanego charakteru – wymaga starannego strojenia i nie zawsze dawał jednoznaczny wynik. W praktyce dobór metody powinien być uzależniony od charakterystyki danych i celu analizy.