SPRAWOZDANIE

Zajęcia: Nauka o danych II

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 8 Data 16.06.2025 Temat: Zaawansowane techniki analizy skupień Wariant 6	Imię Nazwisko Hubert Mentel Informatyka II stopień, niestacjonarne, 2 semestr, gr.1a

1. Zadanie:

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

Treść:
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)

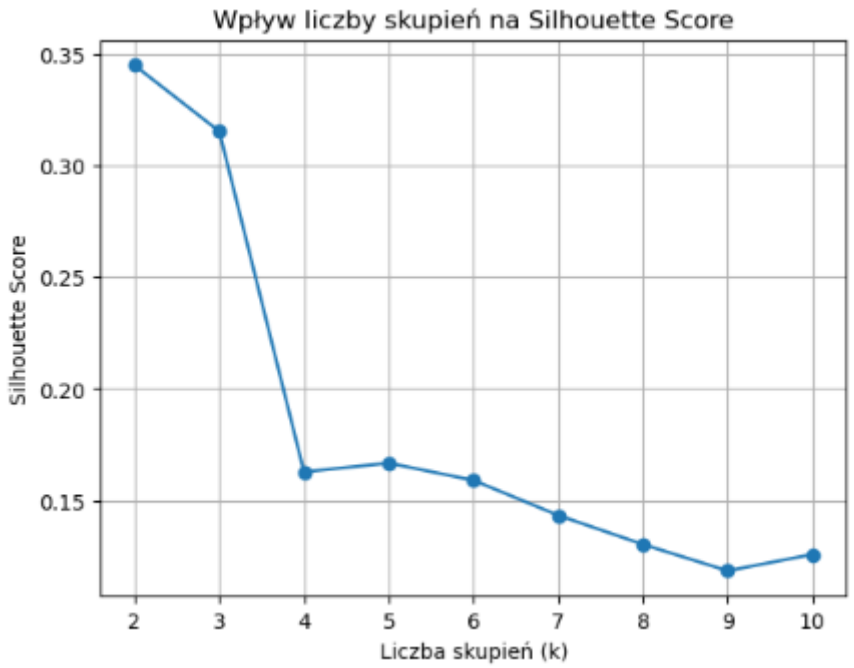
Pliki dostępne są pod linkiem:
https://github.com/HubiPX/NOD/tree/master/NOD2/Zadanie%208

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

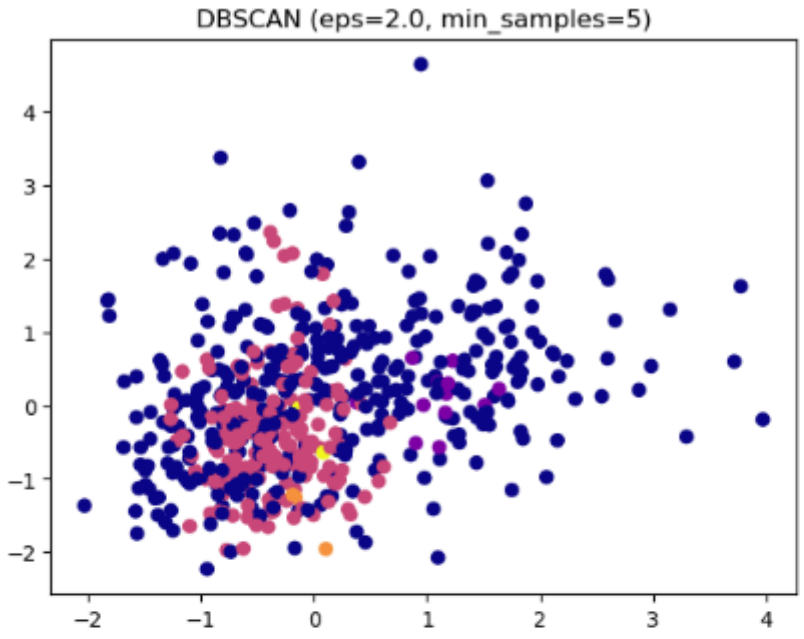
[4]:
```
import pandas as pd
#nazwy kolumn
column_names = [
    "ID", "Diagnosis",
    "radius1", "texture1", "perimeter1", "area1", "smoothness1", "compactness1", "concavity1", "concave_points1",
    "symmetry1", "fractal_dimension1",
    "radius2", "texture2", "perimeter2", "area2", "smoothness2", "compactness2", "concavity2", "concave_points2",
    "symmetry2", "fractal_dimension2",
    "radius3", "texture3", "perimeter3", "area3", "smoothness3", "compactness3", "concavity3", "concave_points3",
    "symmetry3", "fractal_dimension3"
]

# Ścieżka do pliku CSV
sciezka = 'wdbc.csv'

# Wczytywanie pliku CSV do DataFrame
df = pd.read_csv(sciezka, header=None, names=column_names)

# Wyświetlenie pierwszych 5 wierszy
print(df.head())
```

```
         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

[5 rows x 32 columns]
```

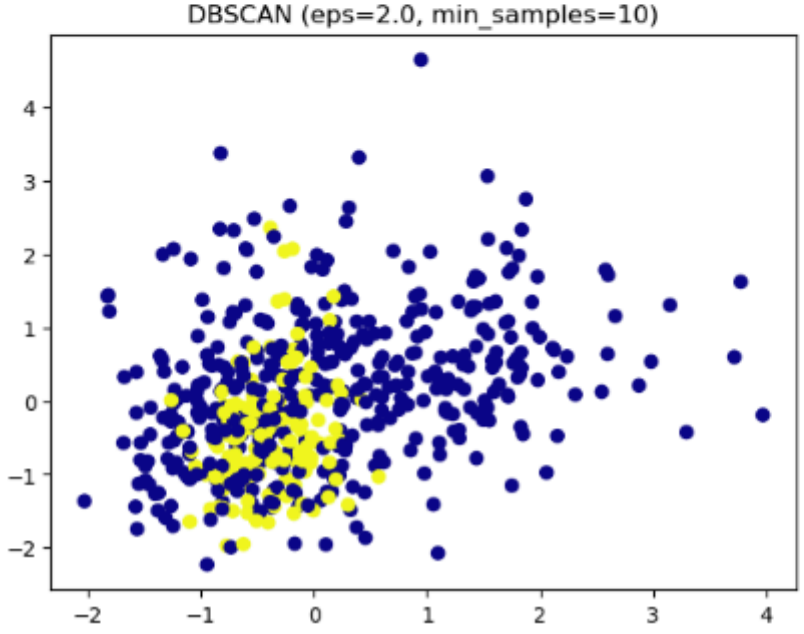
[6]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# 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()
```

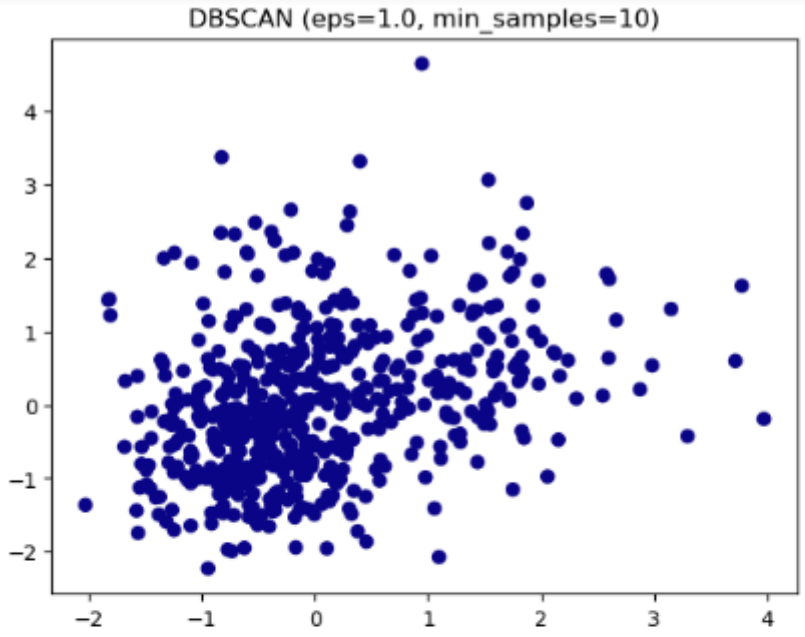


[8]:
```python
# --- Zadanie 2: DBSCAN ---
eps_values = [1.0, 2.0,3.0]
min_samples_values = [3, 5, 10]

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()
```

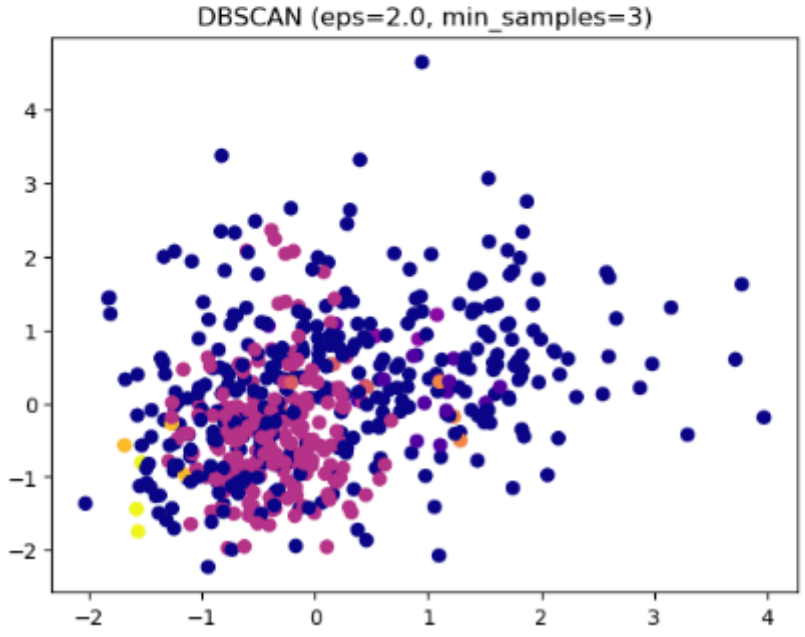



```
DBSCAN (eps=2.0, min_samples=10): zbyt mało klastrów do obliczenia Silhouette
```

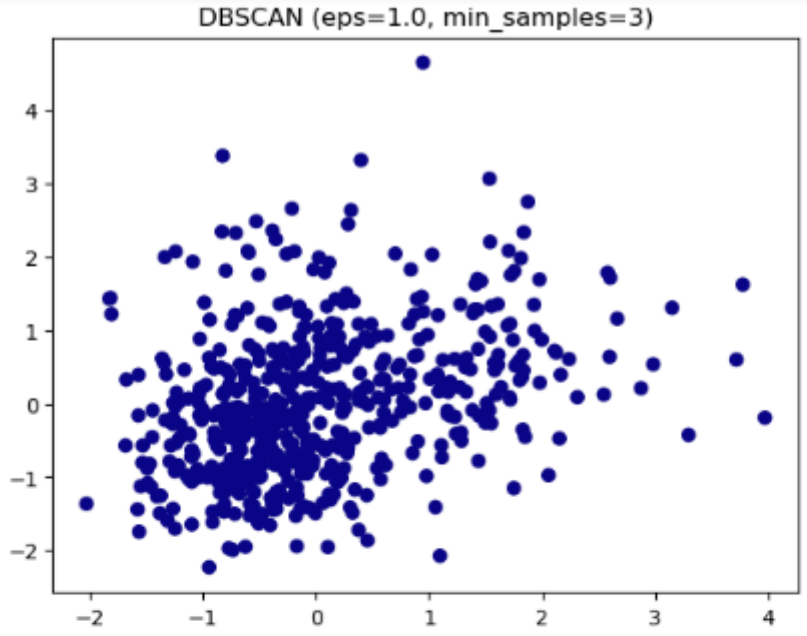




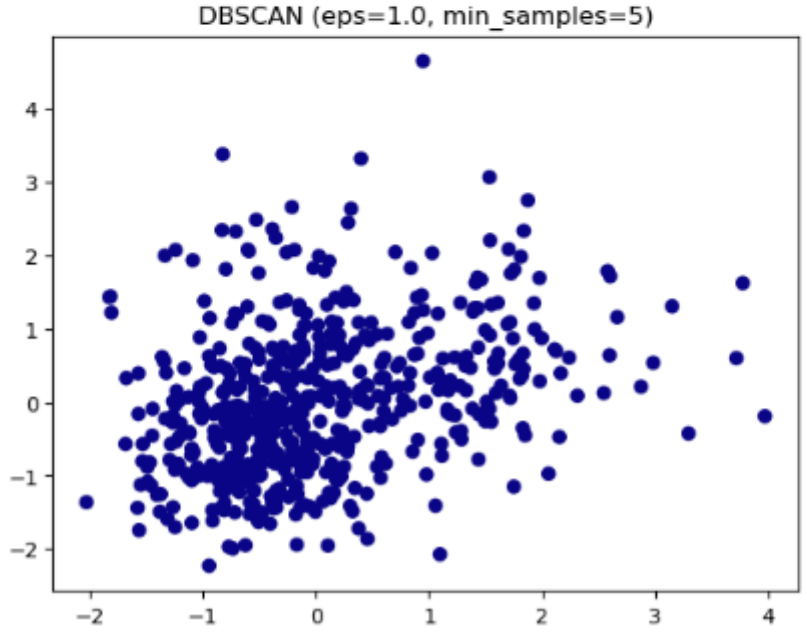

```
DBSCAN (eps=2.0, min_samples=3): Silhouette Score = -0.20, klastry = 7
```




```
DBSCAN (eps=2.0, min_samples=5): Silhouette Score = -0.20, klastry = 4
```



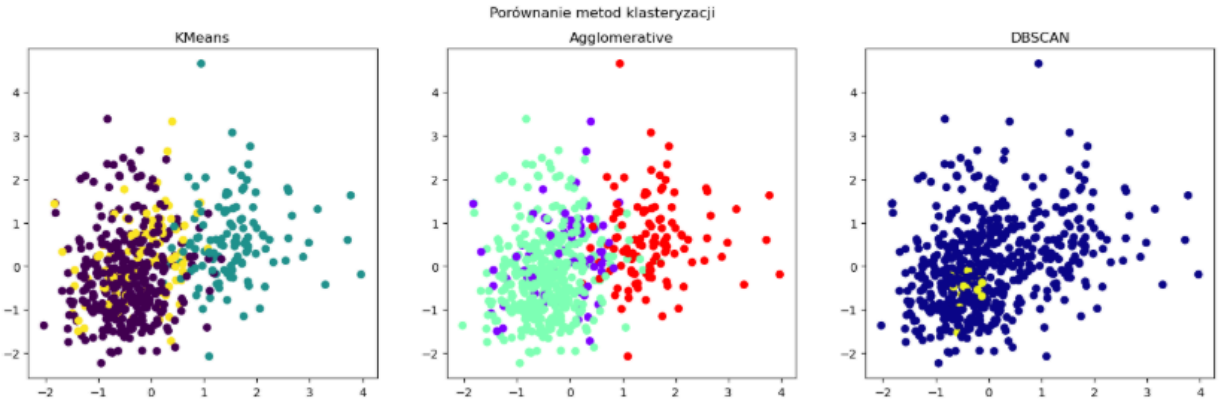

```
DBSCAN (eps=1.0, min_samples=5): zbyt mało klastrów do obliczenia Silhouette
```




```
DBSCAN (eps=1.0, min_samples=10): zbyt mało klastrów do obliczenia Silhouette
```

```python
# --- Zadanie 3: Porównanie metod ---
# Wybór konkretnych konfiguracji
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans_labels = kmeans.fit_predict(X_scaled)
kmeans_score = silhouette_score(X_scaled, kmeans_labels)

agg = AgglomerativeClustering(n_clusters=3)
agg_labels = agg.fit_predict(X_scaled)
agg_score = silhouette_score(X_scaled, agg_labels)

dbscan = DBSCAN(eps=1.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_scaled)
dbscan_score = silhouette_score(X_scaled, dbscan_labels) if len(set(dbscan_labels)) > 1 else -1

# Wyświetlenie wyników
print("\nPorównanie metod (Silhouette Score):")
print(f"- KMeans: {kmeans_score:.2f}")
print(f"- Agglomerative: {agg_score:.2f}")
print(f"- DBSCAN: {dbscan_score:.2f}")

fig, axs = plt.subplots(1, 3, figsize=(18, 5))
axs[0].scatter(X_scaled[:, 0], X_scaled[:, 1], c=kmeans_labels, cmap='viridis')
axs[0].set_title("KMeans")

axs[1].scatter(X_scaled[:, 0], X_scaled[:, 1], c=agg_labels, cmap='rainbow')
axs[1].set_title("Agglomerative")

axs[2].scatter(X_scaled[:, 0], X_scaled[:, 1], c=dbscan_labels, cmap='plasma')
axs[2].set_title("DBSCAN")

plt.suptitle("Porównanie metod klasteryzacji")
plt.show()
```

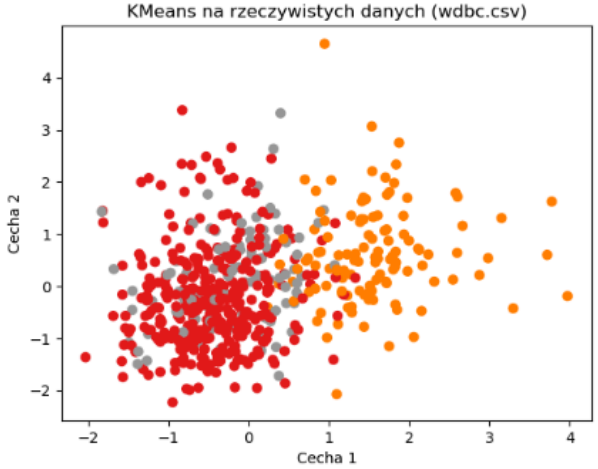
```
Porównanie metod (Silhouette Score):
- KMeans: 0.32
- Agglomerative: 0.33
- DBSCAN: -0.22
```



[10]:

```
# --- Zadanie 4: Zastosowanie KMeans do danych rzeczywistych (oryginalny DataFrame) ---
kmeans_real = KMeans(n_clusters=3, random_state=0)
kmeans_real_labels = kmeans_real.fit_predict(X_scaled)
kmeans_real_score = silhouette_score(X_scaled, kmeans_real_labels)

plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=kmeans_real_labels, cmap='Set1')
plt.title("KMeans na rzeczywistych danych (wdbc.csv)")
plt.xlabel("Cecha 1")
plt.ylabel("Cecha 2")
plt.show()

print(f"Silhouette Score (wdbc.csv, KMeans): {kmeans_real_score:.2f}")
```

```
C:\Users\48664\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```




```
Silhouette Score (wdbc.csv, KMeans): 0.32
```

3. Wnioski

Na podstawie wykresu Silhouette Score dla różnych wartości liczby klastrów (k) w algorytmie K-Means można stwierdzić, że najlepsze wyniki uzyskiwane są przy k=2 i k=3, gdzie współczynniki wynoszą odpowiednio około 0.35 i 0.32. Od k=4 wzwyż następuje gwałtowny spadek wartości Silhouette Score, który następnie utrzymuje się na niskim poziomie (poniżej 0.18), co oznacza, że dalsze dzielenie danych na więcej klastrów prowadzi do pogorszenia jakości klasteryzacji.

Wyniki DBSCAN pokazują, że jakość klasteryzacji silnie zależy od doboru parametrów eps i min_samples. Przy niskim eps i wysokim min_samples nie tworzono wystarczającej liczby klastrów. Najlepszy rezultat uzyskano dla eps=3.0 i min_samples=3, gdzie Silhouette Score wyniósł 0.26. Pozostałe konfiguracje dawały niższe wyniki lub zbyt rozproszone klastry.

Porównanie trzech metod klasteryzacji pokazuje, że najlepszy wynik osiągnięto dla Agglomerative Clustering (Silhouette Score: 0.33), tuż za nim K-Means (0.32), natomiast DBSCAN wypadł znacznie słabiej z ujemnym wynikiem (-0.22). Zarówno K-Means, jak i Agglomerative dobrze oddzieliły grupy danych, natomiast DBSCAN nie poradził sobie z ich strukturą. Wnioskując po wynikach, dane lepiej dopasowują się do metod zakładających konkretną liczbę klastrów.

Zastosowanie algorytmu K-Means do rzeczywistych danych z pliku `wdbc.csv` pozwoliło na wyodrębnienie trzech klastrów o umiarkowanej jakości, co potwierdza współczynnik Silhouette wynoszący **0.32**. Wizualizacja pokazuje, że klastry są częściowo rozdzielne, choć widać pewne nakładanie się grup. Wskazuje to, że dane zawierają wyraźne, ale nie idealnie oddzielone struktury, które K-Means potrafił uchwycić w rozsądny sposób.