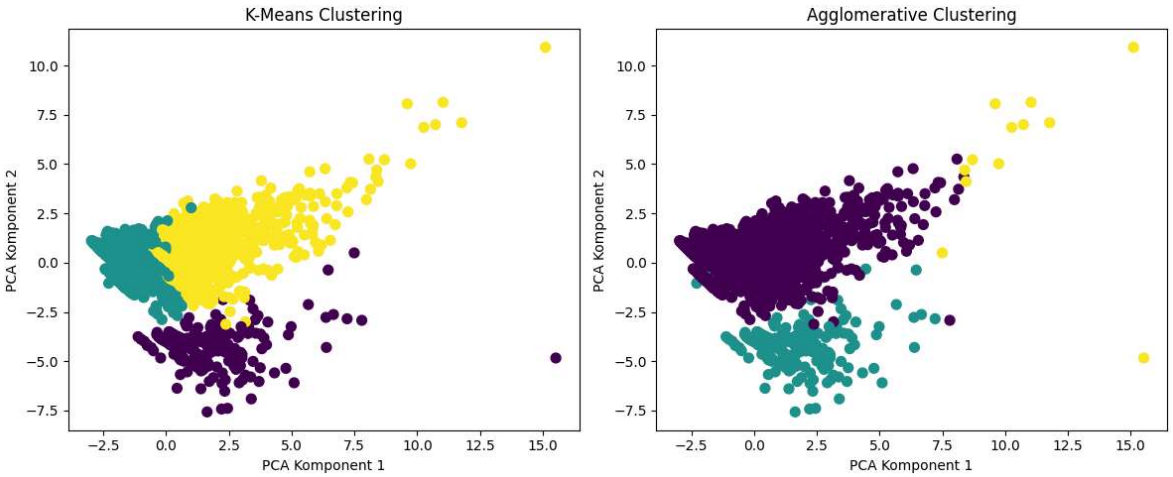
In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import arff
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering

#  Wczytanie pliku ARFF
data, meta = arff.loadarff("seismic-bumps.arff")

# Konwersja do Pandas DataFrame
df = pd.DataFrame(data)

# Dekodowanie wartości kategorycznych (jeśli istnieją)
for col in df.select_dtypes([object]):
    df[col] = df[col].str.decode('utf-8')

#  Podział na cechy (X) i etykiety (y)
X = df.iloc[:, :-1]  # Wszystkie kolumny oprócz ostatniej
y = df.iloc[:, -1]   # Ostatnia kolumna jako etykieta

# Sprawdzenie, które kolumny są tekstowe (kategoryczne)
categorical_cols = X.select_dtypes(include=['object']).columns

#  Konwersja zmiennych kategorycznych na liczbowe (One-Hot Encoding)
X = pd.get_dummies(X, columns=categorical_cols)

#  Standaryzacja cech
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#  Redukcja wymiarowości za pomocą PCA (2D)
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X_scaled)

#  K-Means (analiza skupień)
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans_labels = kmeans.fit_predict(X_scaled)

#  Agglomerative Clustering (klasteryzacja hierarchiczna)
agglo = AgglomerativeClustering(n_clusters=3)
agglo_labels = agglo.fit_predict(X_scaled)

#  Wizualizacja wyników PCA + Klasteryzacja
plt.figure(figsize=(12, 5))

# K-Means
plt.subplot(1, 2, 1)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=kmeans_labels, cmap='viridis', s
plt.title('K-Means Clustering')
plt.xlabel('PCA Komponent 1')
plt.ylabel('PCA Komponent 2')

# Agglomerative Clustering
plt.subplot(1, 2, 2)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=agglo_labels, cmap='viridis', s=
```

```
plt.title('Agglomerative Clustering')
plt.xlabel('PCA Komponent 1')
plt.ylabel('PCA Komponent 2')

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

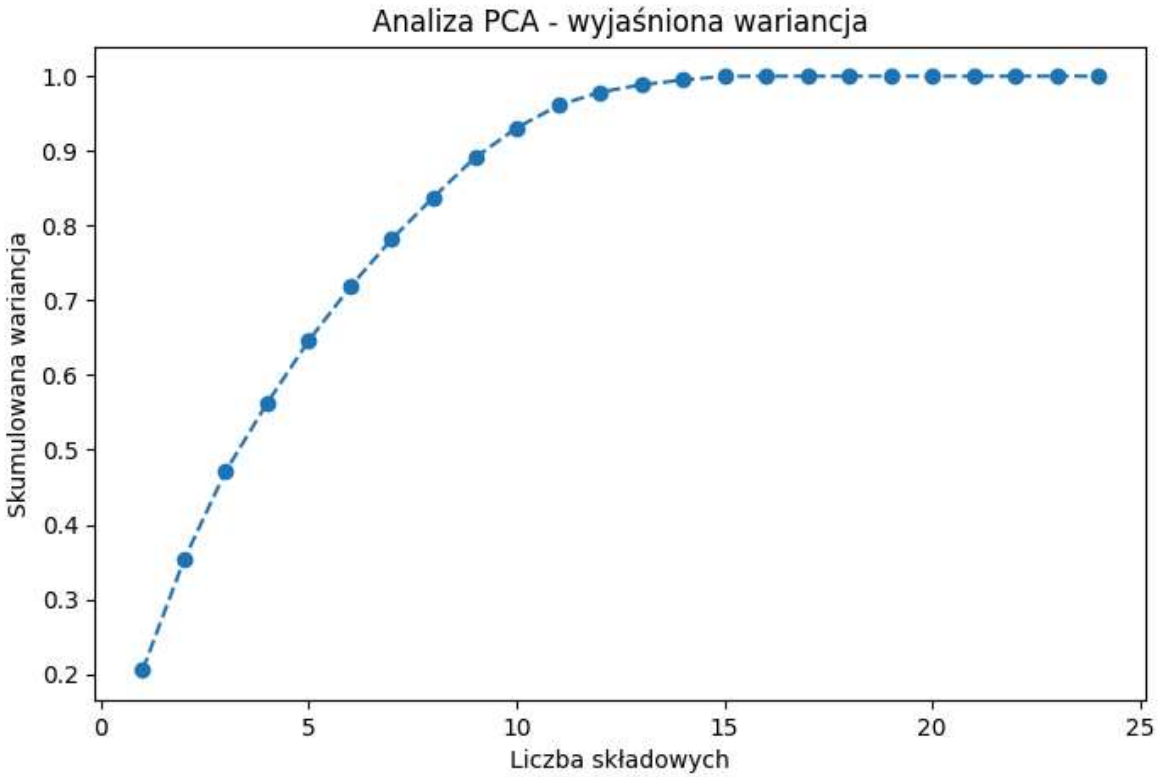


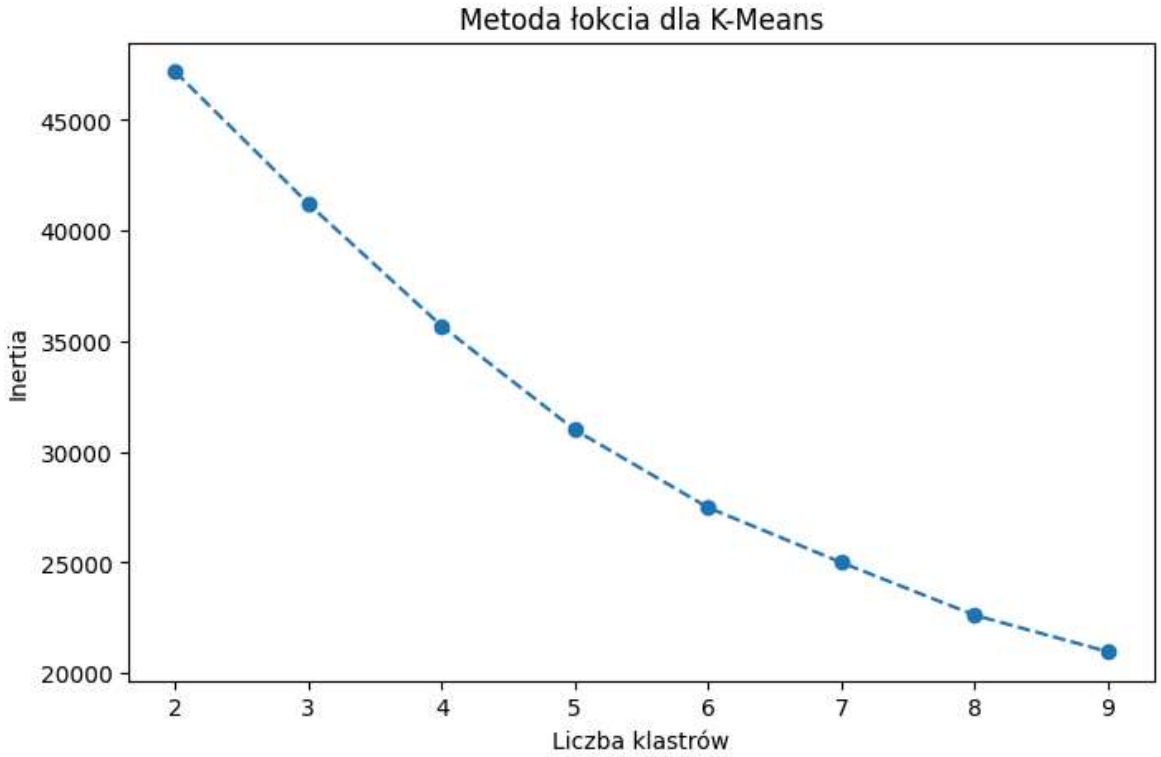
In [2]:

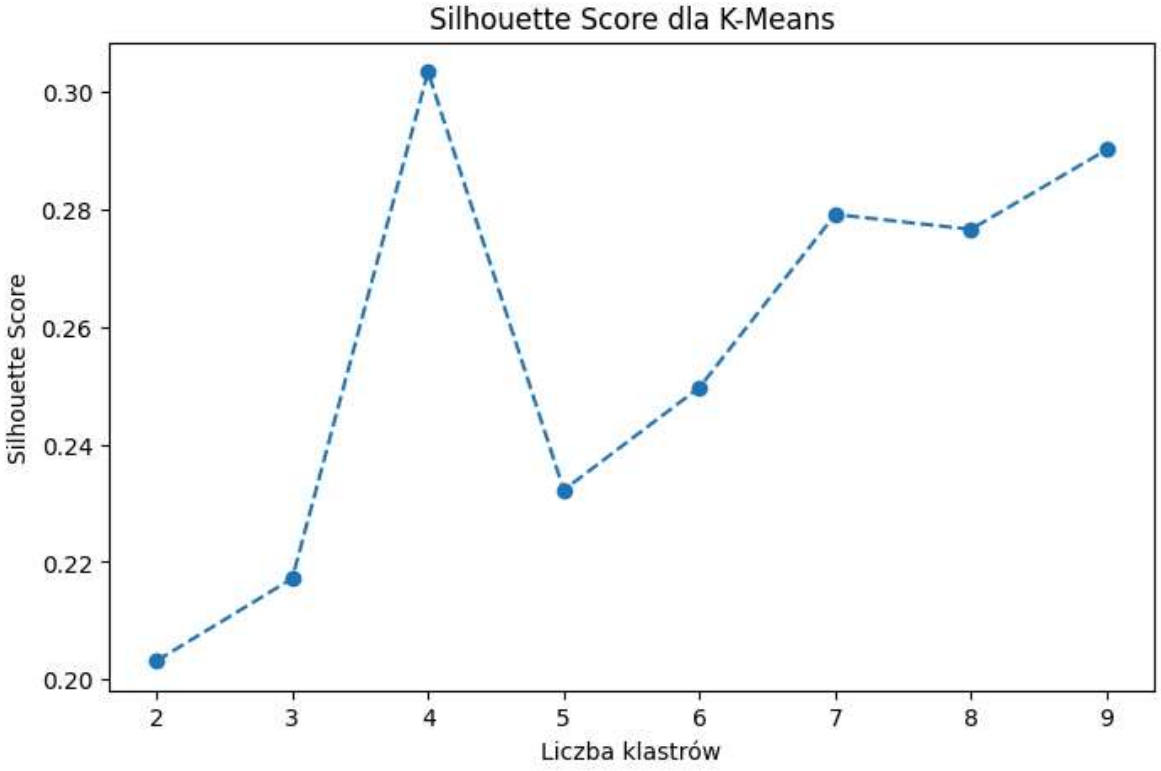
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import arff
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.metrics import silhouette_score

#  Wczytanie pliku ARFF
data, meta = arff.loadarff("seismic-bumps.arff")

# Konwersja do Pandas DataFrame
df = pd.DataFrame(data)

# Dekodowanie wartości kategorycznych (jeśli istnieją)
for col in df.select_dtypes([object]):
    df[col] = df[col].str.decode('utf-8')

#  Podział na cechy (X) i etykiety (y)
X = df.iloc[:, :-1]  # Wszystkie kolumny oprócz ostatniej
y = df.iloc[:, -1]   # Ostatnia kolumna jako etykieta

# Sprawdzenie, które kolumny są tekstowe (kategoryczne)
categorical_cols = X.select_dtypes(include=['object']).columns

#  Konwersja zmiennych kategorycznych na liczbowe (One-Hot Encoding)
X = pd.get_dummies(X, columns=categorical_cols)

#  Standaryzacja cech
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#  Analiza PCA - wykres wyjaśnionej wariancji
pca = PCA()
X_pca = pca.fit_transform(X_scaled)
```

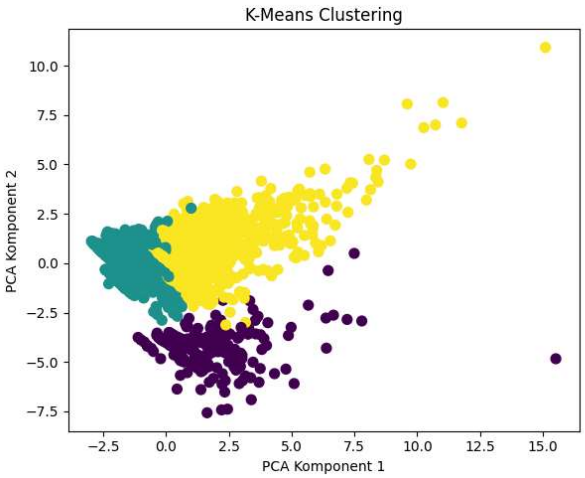
```
plt.figure(figsize=(8, 5))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), np.cumsum(pca.explain
plt.xlabel('Liczba składowych')
plt.ylabel('Skumulowana wariancja')
plt.title('Analiza PCA - wyjaśniona wariancja')
plt.show()

# Redukcja wymiarowości do 2D dla wizualizacji
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X_scaled)

# Metoda łokcia dla K-Means
inertia = []
K_range = range(2, 10)
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(K_range, inertia, marker='o', linestyle='--')
plt.xlabel('Liczba klastrów')
plt.ylabel('Inertia')
plt.title('Metoda łokcia dla K-Means')
plt.show()

# Wybór optymalnej liczby klastrów dla K-Means (przykładowo 3)
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans_labels = kmeans.fit_predict(X_scaled)

# Obliczenie silhouette score dla różnych liczby klastrów
silhouette_scores = []
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X_scaled)
    silhouette_scores.append(silhouette_score(X_scaled, labels))

plt.figure(figsize=(8, 5))
plt.plot(K_range, silhouette_scores, marker='o', linestyle='--')
plt.xlabel('Liczba klastrów')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score dla K-Means')
plt.show()

# Agglomerative Clustering
agglo = AgglomerativeClustering(n_clusters=3)
agglo_labels = agglo.fit_predict(X_scaled)

# Wizualizacja wyników PCA + Klasteryzacja
plt.figure(figsize=(12, 5))

# K-Means
plt.subplot(1, 2, 1)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=kmeans_labels, cmap='viridis', s
plt.title('K-Means Clustering')
plt.xlabel('PCA Komponent 1')
plt.ylabel('PCA Komponent 2')

# Agglomerative Clustering
```

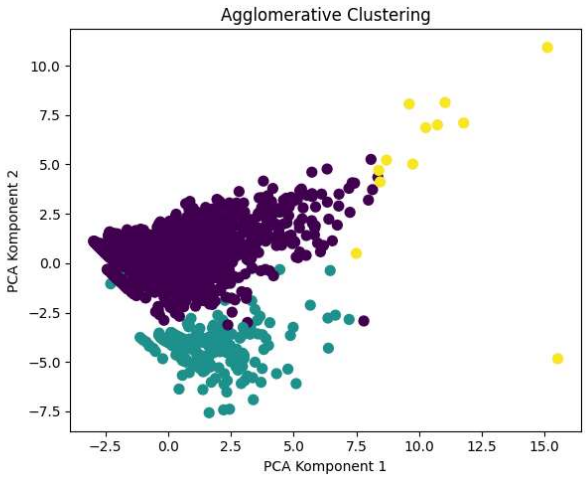
```
plt.subplot(1, 2, 2)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=agglo_labels, cmap='viridis', s=
plt.title('Agglomerative Clustering')
plt.xlabel('PCA Komponent 1')
plt.ylabel('PCA Komponent 2')

plt.tight_layout()
plt.show()

# Dendrogram dla hierarchicznej klasteryzacji
linked = linkage(X_scaled, method='ward')
plt.figure(figsize=(10, 5))
dendrogram(linked, truncate_mode='level', p=5)
plt.title('Dendrogram dla Hierarchicznej Klasteryzacji')
plt.xlabel('Próbki')
plt.ylabel('Odległość')
plt.show()
```

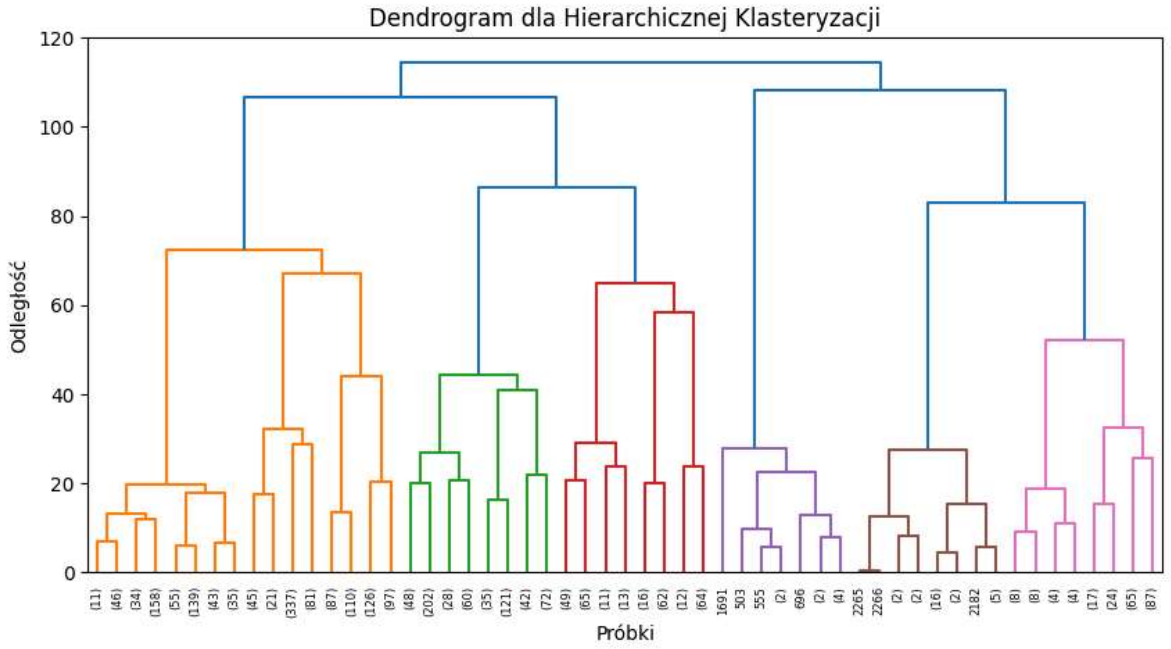



Metoda łokcia dla K-Means
Inertia
45000
40000
35000
30000
25000
20000
2
3
4
5
6
7
8
9
Liczba klastrów


Silhouette Score dla K-Means
Silhouette Score
0.30
0.28
0.26
0.24
0.22
0.20
2
3
4
5
6
7
8
9
Liczba klastrów


K-Means Clustering
PCA Komponent 2
PCA Komponent 1


Agglomerative Clustering
PCA Komponent 2
PCA Komponent 1


Dendrogram dla Hierarchicznej Klasteryzacji
Odległość
Próbki

In [ ]: