PRAKTEK SIGNAL PROCESSING

NAMA : AMALIKA ARI ANINDYA

NIM : 164221029

MATA KULIAH : DATA MINING II

Praktikum Minggu ke-6: Signal Processing

Mata Kuliah Data Mining II Kelas SD-A2

Import Library

```
import wave
import numpy as np
np.random.seed(42)
import matplotlib.pyplot as plt
import os
import librosa as lr
```

Import Dataset

```
datake = 0
file = os.listdir('./heartbeat/set_b')[datake]
wav_file = wave.open('./heartbeat/set_b/'+file, 'rb')
```

Data menggunakan dataset Heartbeat Sound pada set b yang didapatkan dari link Kaggle berikut https://www.kaggle.com/datasets/kinguistics/heartbeat-sounds

Basic Information

```
num_channels = wav_file.getnchannels()
sample_width = wav_file.getsampwidth()
frame_rate = wav_file.getframerate() # Samples per second
num_frames = wav_file.getnframes()
duration = num_frames / float(frame_rate) # dalam seconds

print(f"Channels: {num_channels}")
print(f"Sample Width: {sample_width} bytes")
print(f"Frame Rate: {frame_rate} Hz")
print(f"Number of frames: {num_frames}")
print(f"Duration: {duration:.2f} seconds")
```

```
Channels: 1
Sample Width: 2 bytes
Frame Rate: 4000 Hz
Number of frames: 42385
Duration: 10.60 seconds
```

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II

Atribut sinyal didapatkan dari beberapa metode, yaitu:

1. Channel
   Channel biasanya antara kanan dan kiri.
2. Sample Width
   Sample width menunjukkan jumlah bit yang dimiliki oleh data. Hasilnya ditunjukkan dalam bytes yang per bytesnya memiliki 8 bit. Satu bytes berati 8 bit dan 2 bytes berarti 16 bit.
3. Frame Rate
   Frame Rate merupakan Sample Rate yang didapatkan dari data sound. Data ditampilkan berbentuk frekuensi dengan satuan Hz.
4. Number of Frames
   Number of Frames atau jumlah total frame merupakan jumlah sampel data sound tersebut.
5. Duration
   Duration didapatkan dari number of frames / frame rate. Duration menunjukkan seberapa lama audio tersebut dalam satuan detik.

Ambil Array dari Sound

Membentuk feature vektor dasar menggunakan numpy

```
frames = wav_file.readframes(num_frames) # membaca jumlah frame dari audio
wav_file.close() # menutup audio
```

```
audio_data = np.frombuffer(frames, dtype=np.int16)
```

Variabel frames mengandung jumlah frames dari data audio (wav_file). Kemudian variabel audio_data berisi array yang diubah dari angka jumlah frames menggunakan numpy. Dtype dituliskan int16 karena berisi 2 bytes atau 16 bit dan dapat dilihat dari Sample Width. Hal tersebut akan mengefek pada nilai maksimum dan minimum amplitudo.

Apabila channel lebih dari satu, dapat digabungkan menjadi satu channel menggunakan numpy reshape. Channel dapat dibiarkan saja lebih dari satu namun jika dicluster akan berat.

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II

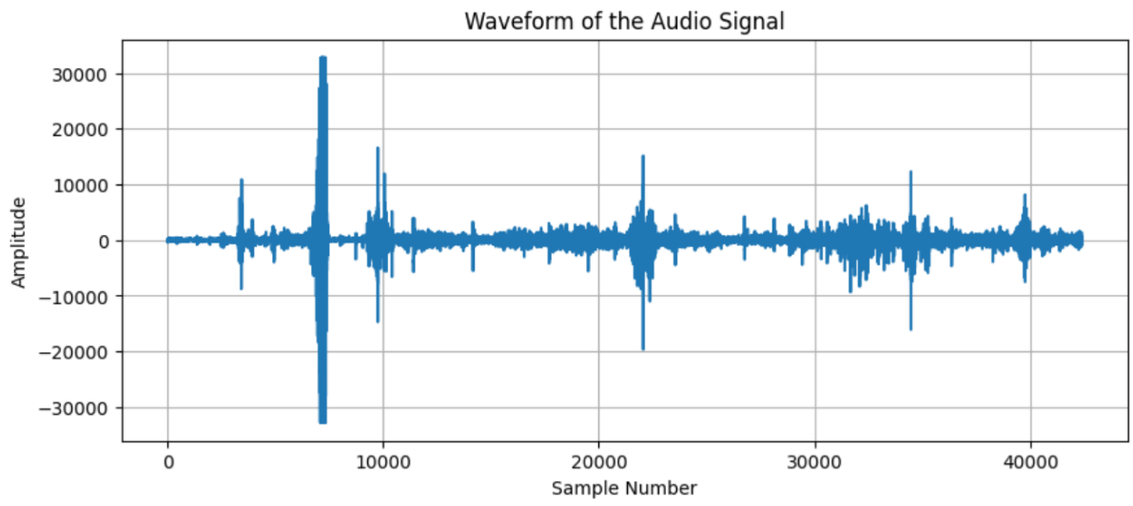
```
if num_channels > 1:
    audio_data = np.reshape(audio_data, (-1, num_channels))
```

```
print("Audio Array:", audio_data)
print("Array Shape:", audio_data.shape)
```

```
Audio Array: [-204 -117   90 ...  495  575  594]
Array Shape: (42385,)
```

Plot Waveform Menggunakan Matplotlib

```
plt.figure(figsize=(10,4))
if num_channels > 1:
    plt.plot(audio_data[:, 0])
else:
    plt.plot(audio_data)
plt.title("Waveform of the Audio Signal")
plt.xlabel("Sample Number")
plt.ylabel("Amplitude")
plt.grid()
plt.show()
```



PRAKTEK SIGNAL PROCESSING

NAMA : AMALIKA ARI ANINDYA

NIM : 164221029

MATA KULIAH : DATA MINING II

Feature Extraction

Feature extraction digunakan untuk mengubah audio data menjadi bentuk yang lebih padat namun tetap representatif menggunakan fitur audio yaitu Mel-frequency cepstral coefficients (MFCCs) dengan bantuan library librosa.

```
import librosa

mfcc_features = librosa.feature.mfcc(y=audio_data.astype(float), sr=frame_rate, n_mfcc=13)
```

Pre-processing

Normalisasi

Normalisasi menggunakan fungsi StandardScaler dari library sklearn.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
mfcc_scaled = scaler.fit_transform(mfcc_features.T)  # Transpose so each row is a time frame
```

Reduksi Dimensi

Reduksi dimensi berguna untuk membantu algoritma clustering dengan menyederhanakan fitur. Reduksi dimensi dijalankan menggunakan UMAP (Uniform Manifold Approximation & Projection) untuk membantu mengurangi noise dan meningkatkan struktur data yang penting.

```
import umap
umap_reducer = umap.UMAP(n_components=2, n_jobs=1, random_state=42)
mfcc_umap = umap_reducer.fit_transform(mfcc_scaled)
```

Clustering

K-Means

K-Means memerlukan pengguna untuk mengetahui jumlah cluster yang akan dibuat. Untuk itu seringnya pengguna menggunakan Elbow Method untuk menentukan jumlah cluster yang tepat. K-Means juga baik digunakan saat datanya terlihat terpisah dan besar. Karena K-Means sensitif terhadap outliers yang bisa menggeser centroids.

PRAKTEK SIGNAL PROCESSING

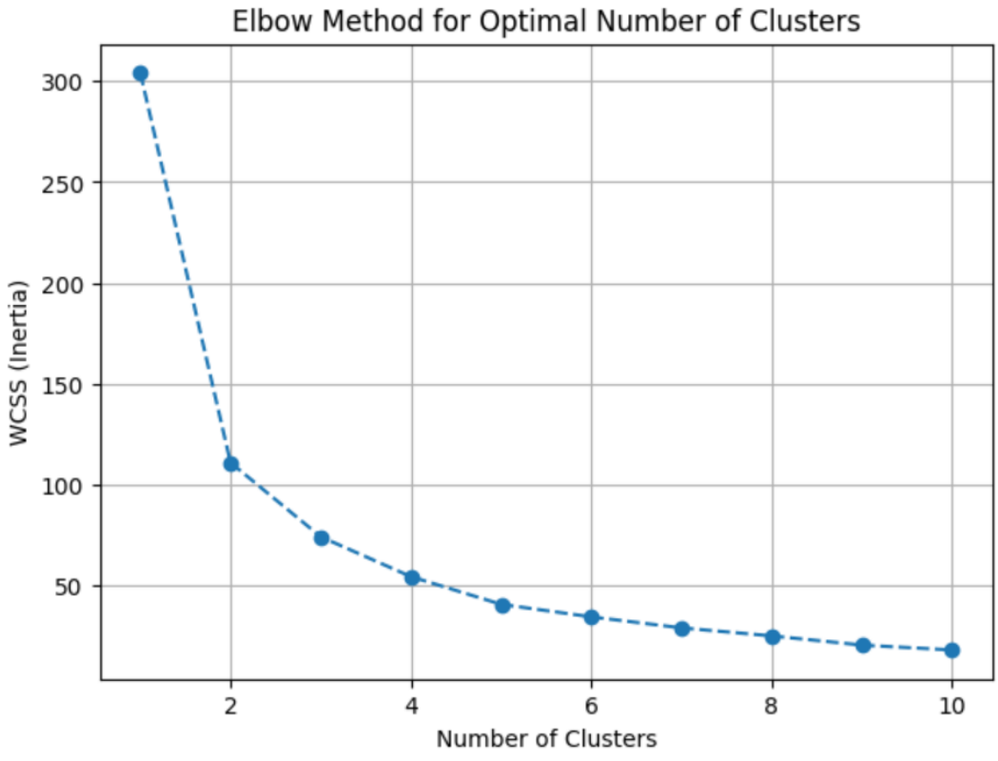
NAMA : AMALIKA ARI ANINDYA

NIM : 164221029

MATA KULIAH : DATA MINING II

Elbow Method

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Define the range for the number of clusters
cluster_range = range(1,11)
inertia = []

# Apply KMeans for each number of clusters and store the inertia (WCSS)
for k in cluster_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(mfcc_umap)
    inertia.append(kmeans.inertia_)  # Inertia is the WCSS

# Plot the number of clusters vs. WCSS (Inertia)
plt.figure(figsize=(8, 6))
plt.plot(cluster_range, inertia, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (Inertia)')
plt.grid(True)
plt.show()
```



Dari visualisasi di atas menunjukkan bahwa jumlah cluster yang tepar sebanyak 2 cluster.

PRAKTEK SIGNAL PROCESSING

NAMA : AMALIKA ARI ANINDYA

NIM : 164221029

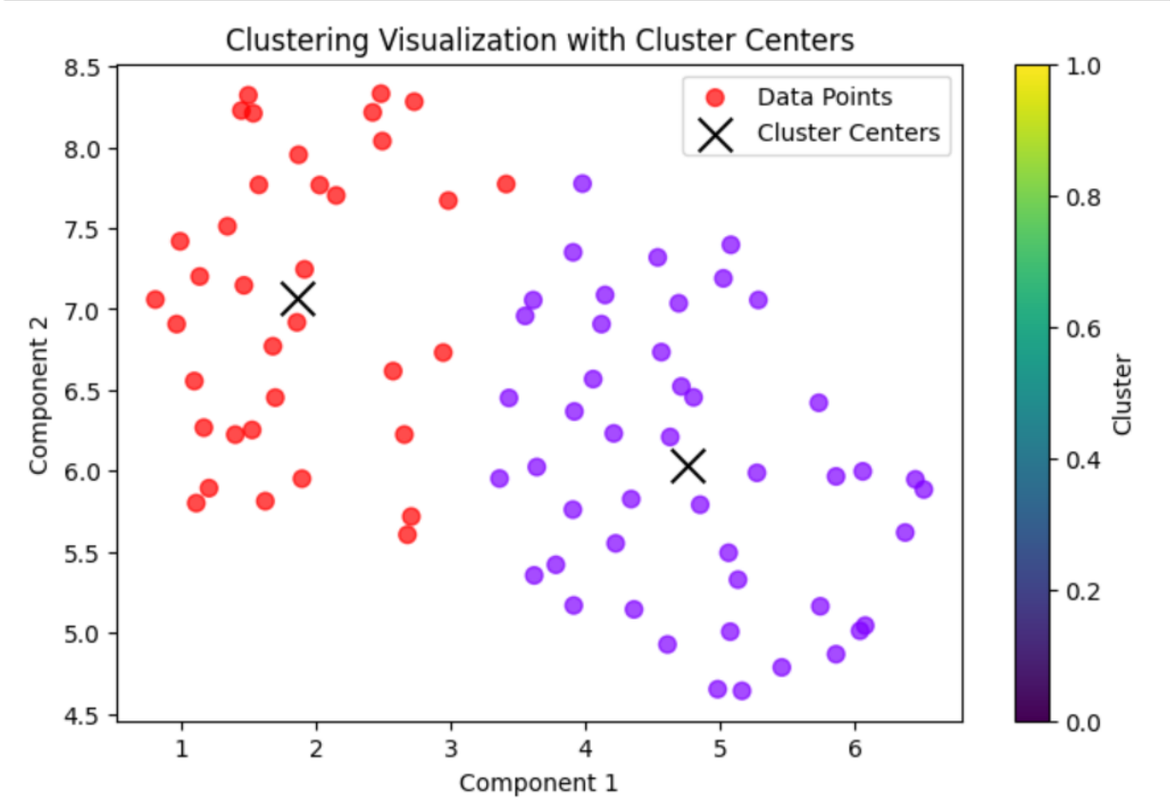
MATA KULIAH : DATA MINING II

Menjalankan Clustering

```
n_clusters = 2 # You can change the number of clusters as needed
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
kmeans.fit(mfcc_umap)

# Get the labels for each segment of the audio
cluster_labels = kmeans.labels_
cluster_centers = kmeans.cluster_centers_
```

Visualisasi

```
# Plot the blobs (clusters) with cluster centers
plt.figure(figsize=(8, 5))
plt.scatter(mfcc_umap[:, 0], mfcc_umap[:, 1], c=cluster_labels, cmap='rainbow', s=50, alpha=0.7, label="Data Points")

# Mark the cluster centers
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], c='black', marker='x', s=200, label="Cluster Centers")

plt.title('Clustering Visualization with Cluster Centers')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar(label='Cluster')
plt.legend(loc='best')
plt.show()
```



Dari K-Means terbentuk 2 cluster dengan cluster centers seperti di atas.

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II

Silhouette Score

```
from sklearn.metrics import silhouette_score

sil_kmeans = silhouette_score(mfcc_umap, cluster_labels)
print(sil_kmeans)
```

```
0.52034646
```

Nilai silhouette score 0.52 menunjukkan nilai yang bagus dari rentang -1 hingga 1 sehingga menunjukkan bahwa cluster sudah dapat mengklasifikasikan label dengan benar namun masih ada hal yang dapat diimprovisasi.

DBScan

DBSCAN ditentukan berdasarkan jarak antar titik. Metode ini tidak memerlukan nilai jumlah cluster di awal karena menggunakan epsilon (threshold jarak) dan minimum sampel. DBSCAN cocok digunakan untuk data yang memang memiliki outliers atau noise dan bentuknya yang tidak normal.

Penentuan Parameter

Fungsi berikut digunakan untuk menentukan jumlah epsilon dan minimum samples dalam DBSCAN.

```
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

# Define ranges for eps and min_samples
eps_range = np.arange(0.1, 3.0, 0.1)  # You can adjust this range
min_samples_range = range(2, 20)  # Adjust this range based on your data

best_eps = None
best_min_samples = None
best_silhouette = -1  # Initialize to a low value
best_labels = None
```

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II

```
# Iterate over combinations of eps and min_samples
for eps in eps_range:
    for min_samples in min_samples_range:
        # Apply DBSCAN with the current eps and min_samples
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        labels = dbscan.fit_predict(mfcc_umap)

        # Only compute silhouette score if there is more than 1 cluster (and no noise points only)
        if len(set(labels)) > 1 and len(set(labels)) != 1 + (labels == -1).sum():
            sil_score = silhouette_score(mfcc_umap, labels)

            # Keep track of the best parameters based on silhouette score
            if sil_score > best_silhouette:
                best_silhouette = sil_score
                best_eps = eps
                best_min_samples = min_samples
                best_labels = labels

print(f'Best eps: {best_eps}, Best min_samples: {best_min_samples}, Best Silhouette Score: {best_silhouette}')

# If you want to visualize the best clustering result:
# Visualize DBSCAN clustering with the best parameters
plt.scatter(mfcc_umap[:, 0], mfcc_umap[:, 1], c=best_labels, cmap='rainbow', s=50, alpha=0.7)
plt.title(f'DBSCAN Clustering with Best Params (eps={best_eps}, min_samples={best_min_samples})')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

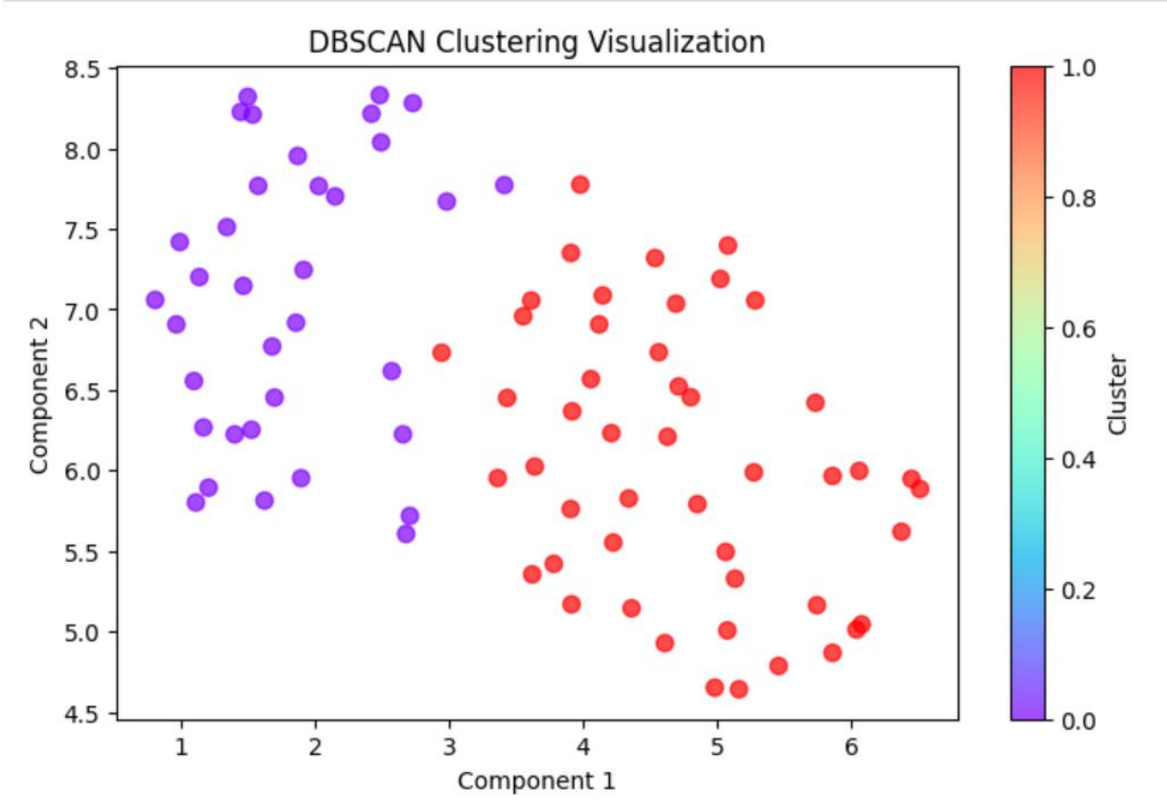
```
Best eps: 1.1, Best min_samples: 14, Best Silhouette Score: 0.5138709545135498
```

Fungsi menghasilan nilai epsilon terbaik adalah 1.1 dan minimum sample terbaiknya 14 dengan silhouette score sebesar 0.51.

Clustering

```
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
import numpy as np

# Apply DBSCAN clustering
dbscan = DBSCAN(eps=1.1, min_samples=14)
dbscan_labels = dbscan.fit_predict(mfcc_umap)

# Visualize the DBSCAN Clusters
plt.figure(figsize=(8, 5))
plt.scatter(mfcc_umap[:, 0], mfcc_umap[:, 1], c=dbscan_labels, cmap='rainbow', s=50, alpha=0.7)
plt.title('DBSCAN Clustering Visualization')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II



Dari DBSCAN terbentuk 2 cluster.

Silhouette Score

```
sil_dbscan = silhouette_score(mfcc_umap, dbscan_labels)
print(sil_dbscan)
```

```
0.51387095
```

Nilai silhouette untuk DBSCAN Clustering adalah 0.51. Hal ini sudah termasuk bagus dalam rentang -1 hingga 1. Artinya cluster terpisahkan dengan cukup baik, namun dapat ditingkatkan lagi.

Hierarchical Clustering

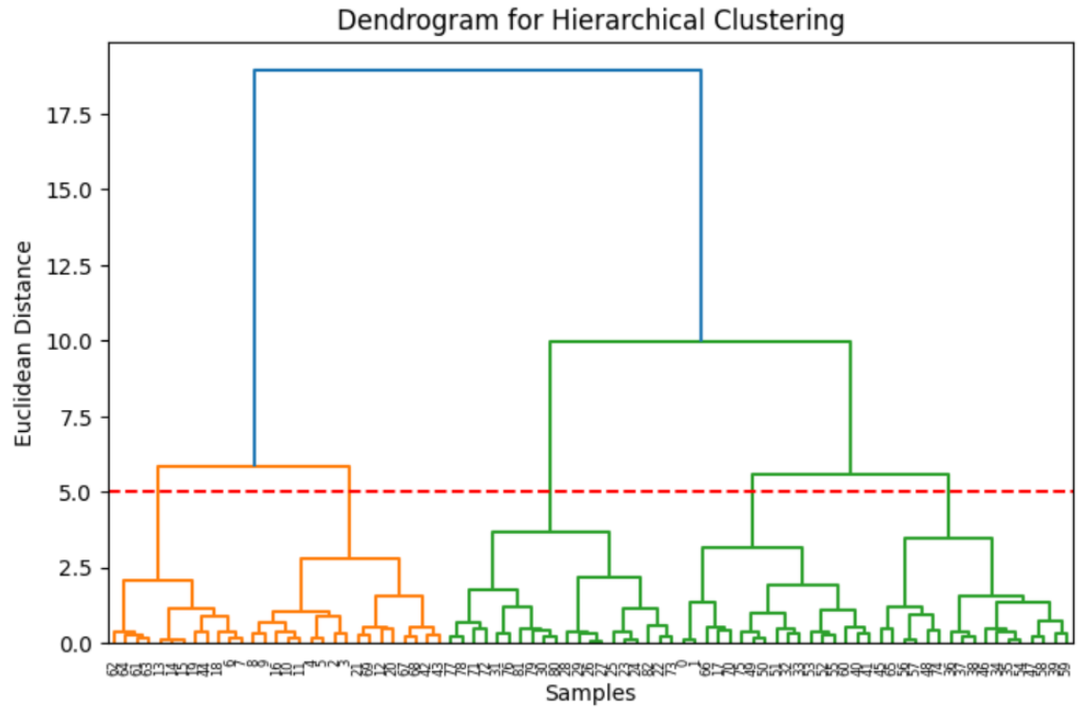
Hierarchical Clustering bekerja dengan menyatukan cluster-cluster kecil sehingga terbentuk tingkatan yang memisahkan data. Hierarchical tidak memerlukan penentuan jumlah cluster, namun jumlah cluster dapat ditentukan dengan mudah dengan menggunakan visualiasasi Dendogram. Kekurangan dari metode ini adalah metode ini tidak cocok untuk dataset yang besar karena memerlukan waktu yang lama dan memori yang besar untuk menjalankannya.

PRAKTEK SIGNAL PROCESSING

NAMA	: AMALIKA ARI ANINDYA
NIM	: 164221029
MATA KULIAH	: DATA MINING II

Penentuan Cluster (Dendogram)

```
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
import numpy as np

# Compute the distance matrix and perform linkage
plt.figure(figsize=(8, 5))
dendrogram = sch.dendrogram(sch.linkage(mfcc_umap, method='ward'))

# Add a horizontal line to mark the cutoff at Euclidean distance = 5
plt.axhline(y=5, color='red', linestyle='--')

plt.title('Dendrogram for Hierarchical Clustering')
plt.xlabel('Samples')
plt.ylabel('Euclidean Distance')
plt.show()
```



Dengan dendogram di atas, saya memutuskan untuk menentukan cutoff pada distance 5. Dari situ dapat dilihat terdapat 5 cluster.

Clustering

```
from sklearn.cluster import AgglomerativeClustering

n_clusters = 5
hier_clustering = AgglomerativeClustering(n_clusters=n_clusters, metric='euclidean', linkage='ward')
cluster_labels = hier_clustering.fit_predict(mfcc_umap)
```

Visualisasi

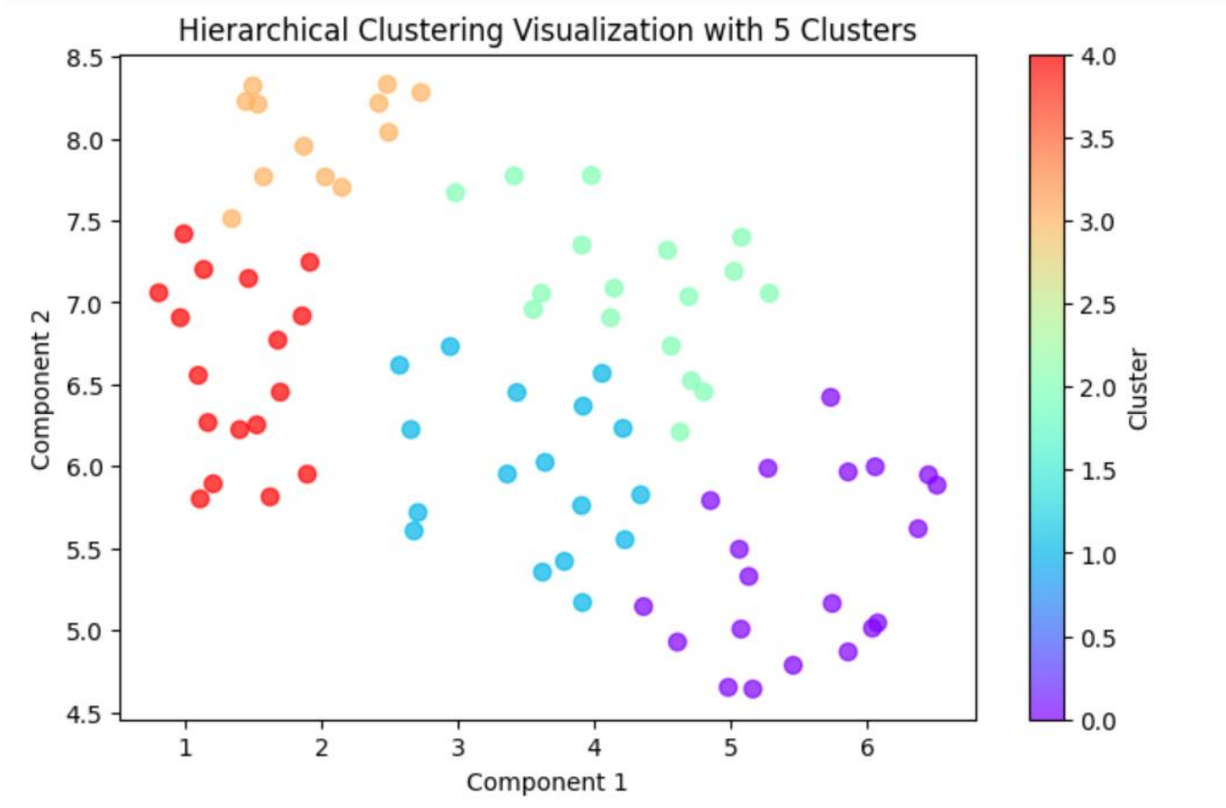
PRAKTEK SIGNAL PROCESSING

NAMA : AMALIKA ARI ANINDYA

NIM : 164221029

MATA KULIAH : DATA MINING II

```
# Visualize the clusters
plt.figure(figsize=(8, 5))
plt.scatter(mfcc_umap[:, 0], mfcc_umap[:, 1], c=cluster_labels, cmap='rainbow', s=50, alpha=0.7)
plt.title(f'Hierarchical Clustering Visualization with {n_clusters} Clusters')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar(label='Cluster')
plt.show()
```



Silhouette Score

```
sil_hc = silhouette_score(mfcc_umap, cluster_labels)
print(sil_hc)
```

```
0.40656933
```

Dengan menggunakan Hierarchical Clustering didapatkan silhouette score sebesar 0.406. Nilai masih termasuk cukup baik dalam menentukan pembagian cluster namun jika dibandingkan dengan metode lain masih kurang.

PRAKTEK SIGNAL PROCESSING

NAMA : AMALIKA ARI ANINDYA

NIM : 164221029

MATA KULIAH : DATA MINING II

Partitioning Around Medoids (PAM) Clustering

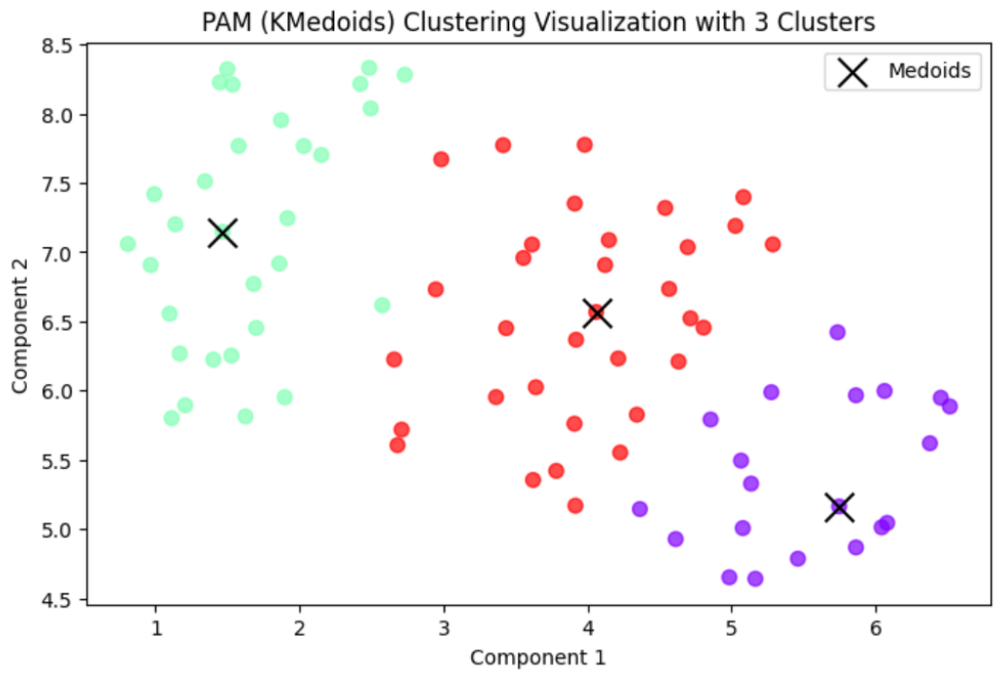
Partitioning Around Medoids Clustering sesuai dengan namanya adalah berpusat pada medoids. Medoids sendiri mirip dengan centroids pada KMeans yaitu titik tengah pada clusternya. Metode ini dapat digunakan pada metrik jarak selain Euclidean. Namun, metode ini tidak cocok untuk dataset yang besar karena komputasinya yang berat.

Clustering

```
from sklearn_extra.cluster import KMedoids

# Apply PAM (KMedoids) Clustering
n_clusters = 3  # Set the number of clusters
pam = KMedoids(n_clusters=n_clusters, metric='euclidean', method='pam', random_state=42)
cluster_labels = pam.fit_predict(mfcc_umap)
```

Visualisasi

```
# Visualize the clusters
plt.figure(figsize=(8, 5))
plt.scatter(mfcc_umap[:, 0], mfcc_umap[:, 1], c=cluster_labels, cmap='rainbow', s=50, alpha=0.7)
plt.scatter(pam.cluster_centers_[:, 0], pam.cluster_centers_[:, 1], c='black', marker='x', s=200, label="Medoids")
plt.title(f'PAM (KMedoids) Clustering Visualization with {n_clusters} Clusters')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.legend(loc='best')
plt.show()
```



Silhouette Score

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II

```
sil_pam = silhouette_score(mfcc_umap, cluster_labels)
print(sil_pam)
```

```
0.42894396
```

Nilai silhouette score sebesar 0.4289 menunjukkan pembagian cluster yang cukup baik namun masih perlu peningkatan. Jika dibandingkan dengan metode lain, metode ini kurang cocok untuk digunakan.

Means Shift Clustering

Means Shift Clustering tidak memerlukan jumlah cluster di awal. Namun, terdapat pengaturan bandwidth yang mengatur estimasi density. Penentuan bandwidth akan sangat berpengaruh pada penghitungannya.

Clustering

```
from sklearn.cluster import MeanShift

# Apply Mean Shift Clustering
mean_shift = MeanShift(bandwidth=None)  # You can also specify 'bandwidth' if known
cluster_labels = mean_shift.fit_predict(mfcc_umap)

# Get the cluster centers
cluster_centers = mean_shift.cluster_centers_

# Print the number of clusters and cluster centers
n_clusters = len(set(cluster_labels))
print(f'Number of Clusters: {n_clusters}')
print(f'Cluster Centers: \n{cluster_centers}')
```

```
Number of Clusters: 2
Cluster Centers:
[[4.4313593 6.2287736]
 [2.7307038 7.033101 ]]
```

Visualisasi

```
# Visualize the clusters and cluster centers
plt.figure(figsize=(8, 5))
plt.scatter(mfcc_umap[:, 0], mfcc_umap[:, 1], c=cluster_labels, cmap='rainbow', s=50, alpha=0.7)
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], c='black', marker='x', s=200, label="Cluster Centers")
plt.title(f'Mean Shift Clustering with {n_clusters} Clusters')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.legend(loc='best')
plt.show()
```

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II

Silhouette Score

```
sil_means = silhouette_score(mfcc_umap, cluster_labels)
print(f'Silhouette Score: {sil_means}')
```

```
Silhouette Score: 0.5019857883453369
```

Nilai silhouette score yang pas pada 0.5 menunjukkan cluster sudah cukup baik untuk ditentukan.

Affinity Propagation

Metode affinity propagation tidak memerlukan jumlah cluster di awal. Sebagai gantinya, pengguna dapat menentukan damping dan preference. Parameter preference mengatur berapa banyak exemplars yang digunakan. Metode ini rawan mengerluarkan cluster yang sedikit.

Clustering

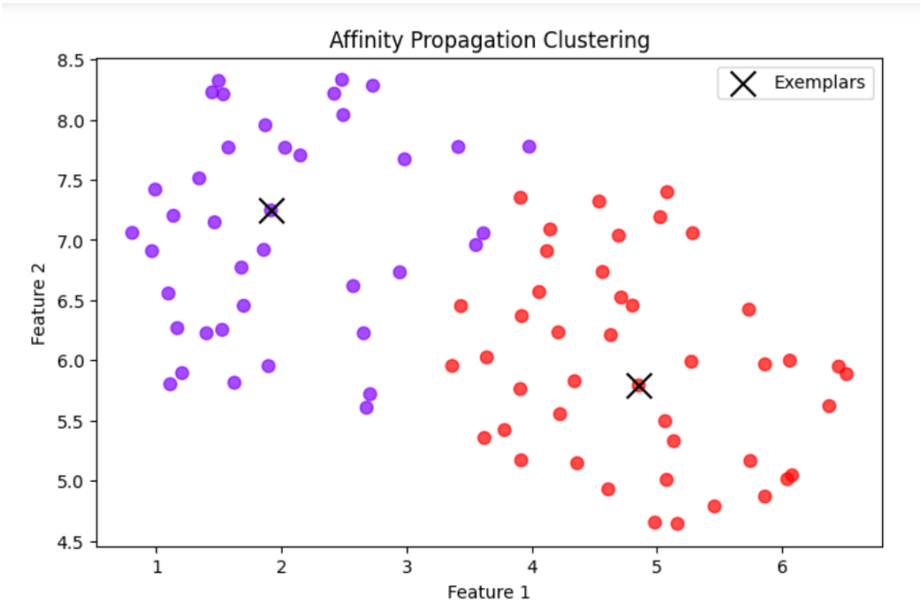
```
from sklearn.cluster import AffinityPropagation

# Apply Affinity Propagation
affinity_propagation = AffinityPropagation(damping=0.5, preference=-50, random_state=42)
cluster_labels = affinity_propagation.fit_predict(mfcc_umap)

# Get the number of clusters and exemplars
n_clusters = len(np.unique(cluster_labels))
exemplars_indices = affinity_propagation.cluster_centers_indices_

# Print the number of clusters
print(f'Number of clusters: {n_clusters}')
print(f'Exemplar indices: {exemplars_indices}')
```

```
Number of clusters: 2
Exemplar indices: [12 71]
```

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II

Visualisasi

```
# Visualization
plt.figure(figsize=(8, 5))
plt.scatter(mfcc_umap[:, 0], mfcc_umap[:, 1], c=cluster_labels, cmap='rainbow', s=50, alpha=0.7)
plt.scatter(mfcc_umap[exemplars_indices, 0], mfcc_umap[exemplars_indices, 1], c='black', marker='x', s=200, label='Exemplars'
plt.title('Affinity Propagation Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend(loc='best')
plt.show()
```



Silhouette Score

```
# Calculate and print the silhouette score
if n_clusters > 1:  # Silhouette score is not defined for a single cluster
    sil_affi = silhouette_score(mfcc_umap, cluster_labels)
    print(f'Silhouette Score: {sil_affi:.4f}')
```

```
Silhouette Score: 0.5072
```

Nilai silhouette score sebesar 0.5072 sudah lumayan baik untuk menentukan cluster.

Spectral Clustering

Spectral Clustering menggunakan nilai eigenvalues dalam matriks yang mirip untuk mereduksi dimensi. Metode ini memerlukan jumlah cluster di awal dan dapat mendeteksi cluster yang non-convex. Metode ini tidak cocok untuk dataset yang besar karena pengerjaannya berat.

Menentukan Jumlah Cluster

PRAKTEK SIGNAL PROCESSING

NAMA : AMALIKA ARI ANINDYA

NIM : 164221029

MATA KULIAH : DATA MINING II


```
from sklearn.cluster import SpectralClustering

silhouette_scores = []
cluster_range = range(2, 10)  # Adjust this range as needed

for n_clusters in cluster_range:
    spectral_clustering = SpectralClustering(n_clusters=n_clusters, affinity='nearest_neighbors', random_state=42)
    cluster_labels = spectral_clustering.fit_predict(mfcc_umap)

    if n_clusters > 1:  # Silhouette score is not defined for a single cluster
        sil_score = silhouette_score(mfcc_umap, cluster_labels)
        silhouette_scores.append(sil_score)
        print(f'Number of clusters: {n_clusters}, Silhouette Score: {sil_score:.4f}')
```

```
Number of clusters: 2, Silhouette Score: 0.5108
Number of clusters: 3, Silhouette Score: 0.4266
Number of clusters: 4, Silhouette Score: 0.3979
Number of clusters: 5, Silhouette Score: 0.4083
Number of clusters: 6, Silhouette Score: 0.3905
Number of clusters: 7, Silhouette Score: 0.3732
Number of clusters: 8, Silhouette Score: 0.3497
Number of clusters: 9, Silhouette Score: 0.3716
```

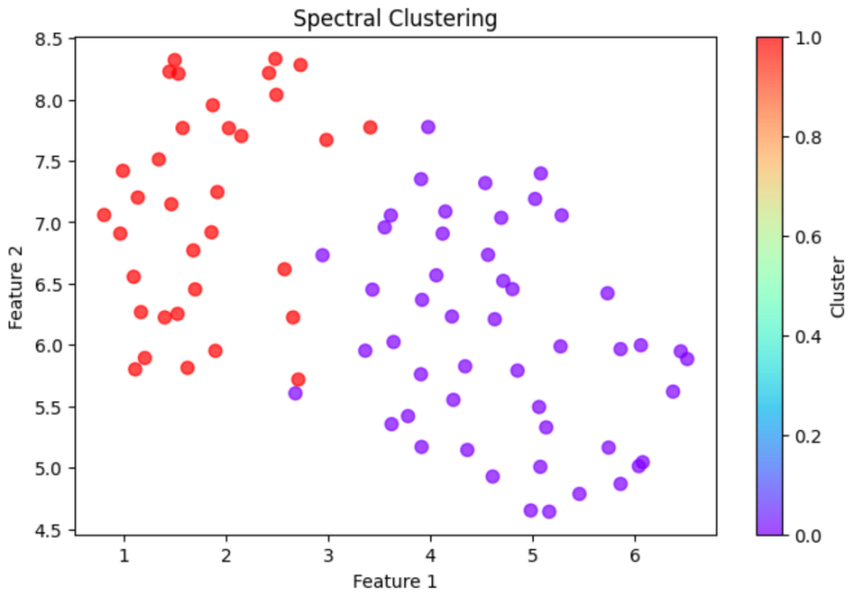
Clustering

```
from sklearn.cluster import SpectralClustering

# Apply Spectral Clustering
n_clusters = 2  # Set the number of clusters you want
spectral_clustering = SpectralClustering(n_clusters=n_clusters, affinity='nearest_neighbors', random_state=42)
cluster_labels = spectral_clustering.fit_predict(mfcc_umap)
```

Visualisasi

```
# Visualization
plt.figure(figsize=(8, 5))
plt.scatter(mfcc_umap[:, 0], mfcc_umap[:, 1], c=cluster_labels, cmap='rainbow', s=50, alpha=0.7)
plt.title('Spectral Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.colorbar(label='Cluster')
plt.show()
```

NAMA : AMALIKA ARI ANINDYA
NIM : 164221029
MATA KULIAH : DATA MINING II



Silhouette Score

```
# Calculate and print the silhouette score
if n_clusters > 1:  # Silhouette score is not defined for a single cluster
    sil_spec = silhouette_score(mfcc_umap, cluster_labels)
    print(f'Silhouette Score: {sil_spec:.4f}')
```

```
Silhouette Score: 0.5108
```

Nilai silhouette score 0.5108 berarti metode ini sudah cukup baik untuk memisahkan cluster yang sesuai.

Perbandingan Metode Clustering

Metode	Jumlah Cluster	Silhouette Score
KMeans	2	0.5203
DBScan	2	0.5139
Hierarchical	5	0.4066
PAM	3	0.4289
Means Shift	2	0.5020
Affinity Propagation	2	0.5072
Spectral Clustering	2	0.5108

Dari tabel di atas dapat dilihat bahwa nilai silhouette score tertinggi ada pada metode KMeans, DBScan, dan Spectral Clustering.