Resources FileNotFoundError FileN •••

Please explain the error:

```
ValueError: Length of values (2000) does not
```

UTS Machine Learning - Tugas Clustering

Nama: M Faishal Abdurrahman

NIM: 1103213015

Import Library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score, davies_bouldin_score
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings('ignore')
```

Mount Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call dı
```

Load Dataset

```
path_clustering = '/content/drive/My Drive/UTS/UTSClustering.csv'
df_clustering = pd.read_csv(path_clustering, encoding='latin1')
```

EKSPLORASI DATA AWAL

```
print("Informasi Dataset:")
print(df_clustering.info())

print("\nStatistik Deskriptif:")
print(df_clustering.describe())

print("\nJumlah data missing:")
print(df_clustering.isnull().sum())

print("\nSampel Data:")
df_clustering.head()

# Identifikasi tipe data kolom
numerical_cols = df_clustering.select_dtypes(include=['float64', 'int64']).columns.t
categorical_cols = df_clustering.select_dtypes(include=['object', 'category']).colur

print(f"\nKolom Numerik ({len(numerical_cols)}): {numerical_cols}")
print(f"\nKolom Kategorikal ({len(categorical_cols)}): {categorical_cols}")
```

```
Informasi Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
```

```
memory usage: 33.1+ MB
None

Statistik Deskriptif:
            Quantity      UnitPrice     CustomerID
count  541909.000000  541909.000000  406829.000000
mean        9.552250       4.611114   15287.690570
std       218.081158      96.759853    1713.600303
min    -80995.000000  -11062.060000   12346.000000
25%         1.000000       1.250000   13953.000000
50%         3.000000       2.080000   15152.000000
75%        10.000000       4.130000   16791.000000
max     80995.000000   38970.000000   18287.000000

Jumlah data missing:
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64

Sampel Data:

Kolom Numerik (3): ['Quantity', 'UnitPrice', 'CustomerID']

Kolom Kategorikal (5): ['InvoiceNo', 'StockCode', 'Description', 'InvoiceDate',
```

VISUALISASI DATA

```
#  Berapa banyak fitur yang ingin diplot
n_plot = 6
sel_cols = numerical_cols[:n_plot]

# Sampling data
df_sample = df_clustering.sample(n= min(len(df_clustering), 1000), random_state=42)

# 3. Histogram
n_cols = 3
n_rows = math.ceil(n_plot / n_cols)
fig, axes = plt.subplots(n_rows, n_cols, figsize=(4*n_cols, 3*n_rows))
for ax, col in zip(axes.flatten(), sel_cols):
    sns.histplot(df_sample[col], kde=False, ax=ax, bins=30)
    ax.set_title(col)
# kosongkan sisa subplot jika ada
for ax in axes.flatten()[n_plot:]:
    ax.axis('off')
plt.tight_layout()
plt.show()

# 4. Scatter matrix (ambil 4 fitur pertama saja atau sesuai kebutuhan)
sm_cols = sel_cols[:4]
scatter_matrix(df_sample[sm_cols], diagonal='hist', alpha=0.5,
               figsize=(3*len(sm_cols), 3*len(sm_cols)))
plt.suptitle('Scatter Matrix', y=1.02)
plt.show()

# 5. Heatmap korelasi
plt.figure(figsize=(4*len(sel_cols), 4))
sns.heatmap(df_sample[sel_cols].corr(), annot=False, cmap='coolwarm')
plt.title('Matriks Korelasi')
plt.show()
```

Hypothesis

The error message "ValueError: Length of values (2000) does not match length of index (541909)" indicates that you're trying to assign a NumPy array (`best_clusters`) to a new column in your DataFrame (`df_clustering`), but the array's size (2000) doesn't match the number of rows in your DataFrame (541909).

This mismatch likely occurred because during preprocessing, you created a subsampled DataFrame `df_clean` to handle memory constraints and potentially improve performance:

```
max_samples = 2000
if len(df_clean) > max_samples:
    df_clean = df_clean.sample(max_samples, r
```

Use code with caution

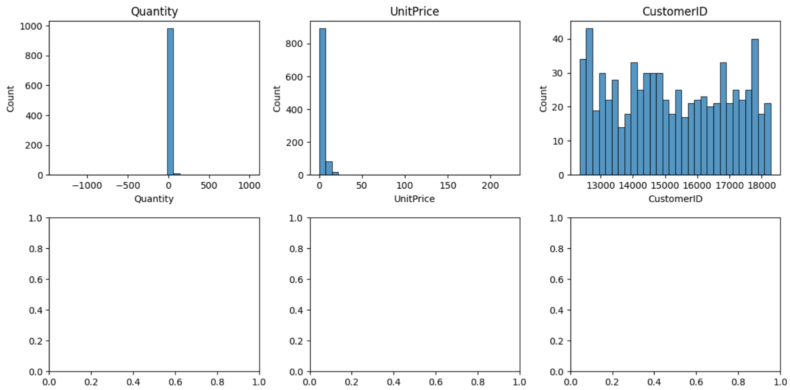
You then performed clustering on this subsampled data (`df_clean`), resulting in cluster labels (`best_clusters`) with a length of 2000. However, in the final step, you're attempting to add these labels to the original, full DataFrame `df_clustering` , which has 541909 rows. This size difference leads to the ValueError.

Suggested Changes

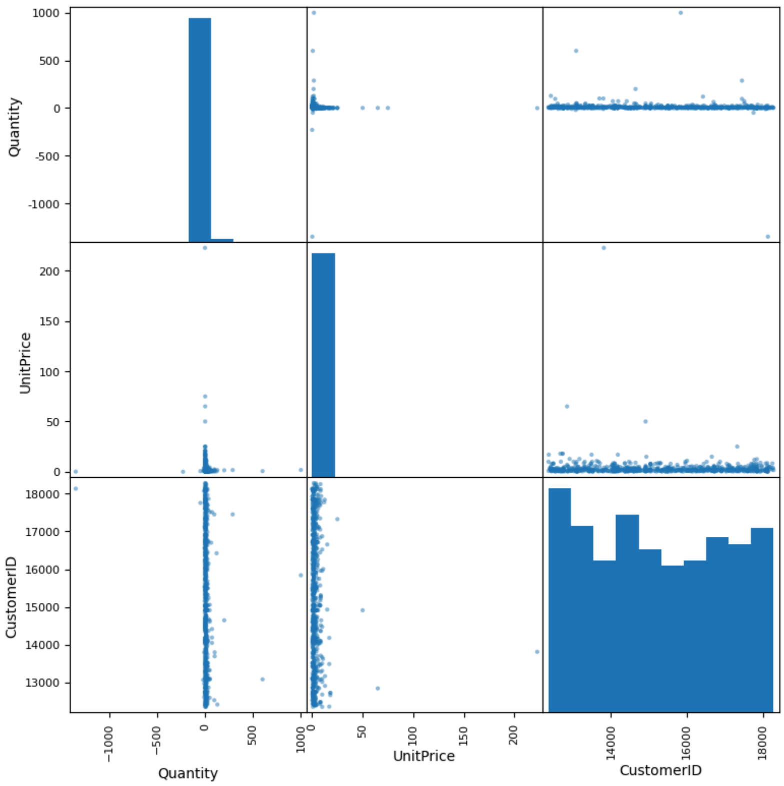
```
# Simpan model terbaik
import joblib
import os

# Buat direktori jika belum ada
os.makedirs('/content/drive/My Drive/clusteri

if best_model == 'KMeans':
    joblib.dump(kmeans, '/content/drive/My Dr
elif best_model == 'Agglomerative':
    joblib.dump(agg_clustering, '/content/dri
elif best_model == 'GMM':
    joblib.dump(gmm, '/content/drive/My Drive

# Fit the best model to the entire dataset to
if best_model == 'KMeans':
    best_clusters_full = kmeans.predict(prepr
elif best_model == 'Agglomerative':
    best_clusters_full = agg_clustering.fit_p
elif best_model == 'GMM':
    best_clusters_full = gmm.predict(preproce

# Tambahkan cluster ke data asli
df_clustering['Cluster'] = best_clusters_full

# Simpan hasil ke CSV
df_clustering.to_csv('/content/drive/My Drive

print("\nModel dan hasil clustering telah dis
```

Use code with caution

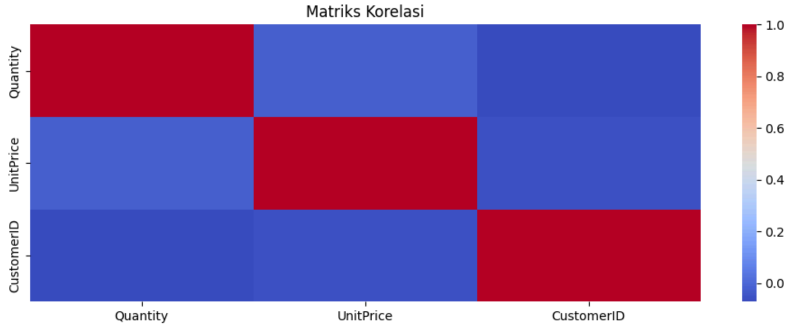
Rate this answer



Scatter Matrix



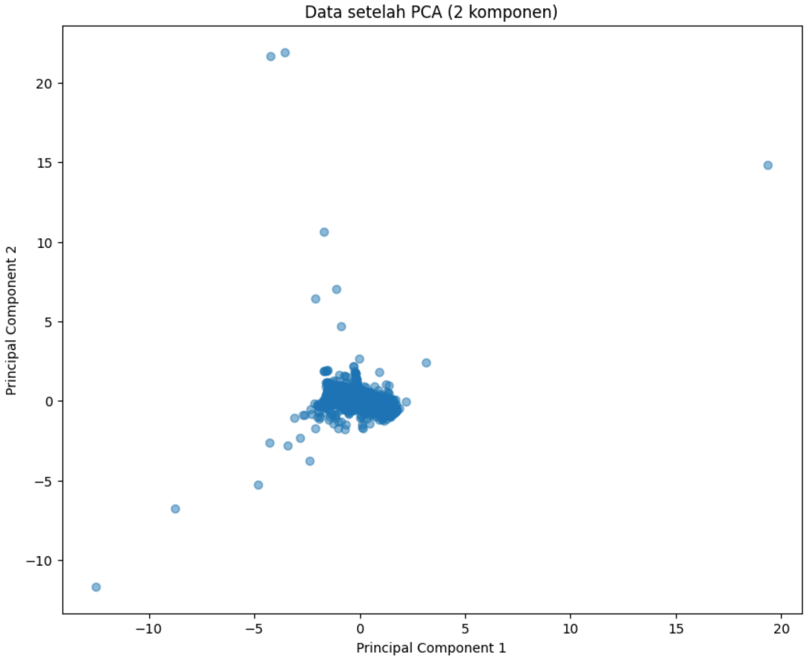
Matriks Korelasi



DATA PREPROCESSING

```
# Handling missing values
df_clean = df_clustering.copy()
max_samples = 2000
if len(df_clean) > max_samples:
    df_clean = df_clean.sample(max_samples, random_state=42)

# Ubah numerik ke float32
df_clean[numerical_cols] = df_clean[numerical_cols].astype('float32')

# Untuk kolom numerik, isi dengan median
for col in numerical_cols:
    if df_clean[col].isnull().sum() > 0:
        df_clean[col].fillna(df_clean[col].median(), inplace=True)
# Untuk kolom kategorikal, isi dengan mode
for col in categorical_cols:
    if df_clean[col].isnull().sum() > 0:
        df_clean[col].fillna(df_clean[col].mode()[0], inplace=True)

# Preprocessing Pipeline
# Buat preprocessor untuk kolom numerik dan kategorikal
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

# Gabungkan preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols) if categorical_cols else
    ],
    remainder='drop'
)

# Terapkan preprocessing
X = preprocessor.fit_transform(df_clean)
print(f"\nBentuk data setelah preprocessing: {X.shape}")

# Dimensionality reduction dengan PCA untuk visualisasi
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
print(f"Variance explained by PCA components: {pca.explained_variance_ratio_}")

# Visualisasi data setelah PCA
plt.figure(figsize=(10, 8))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5)
plt.title('Data setelah PCA (2 komponen)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

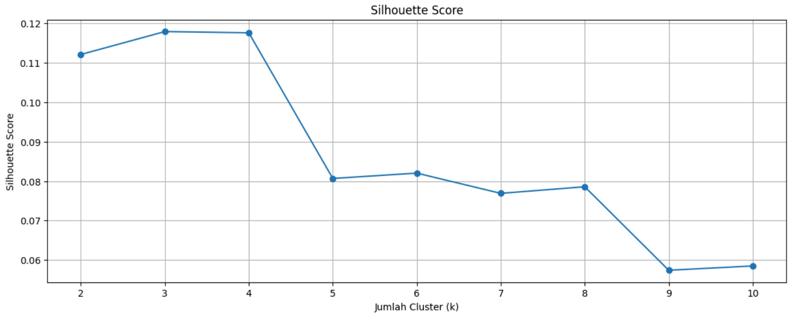
```
Bentuk data setelah preprocessing: (2000, 5779)
Variance explained by PCA components: [0.14888933 0.14480405]
```

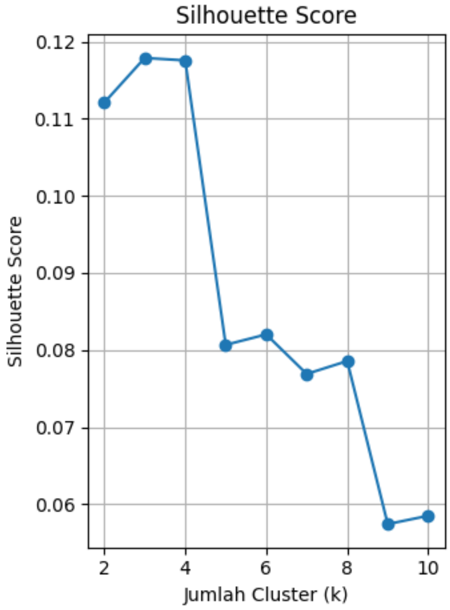


CLUSTERING DENGAN KMEANS

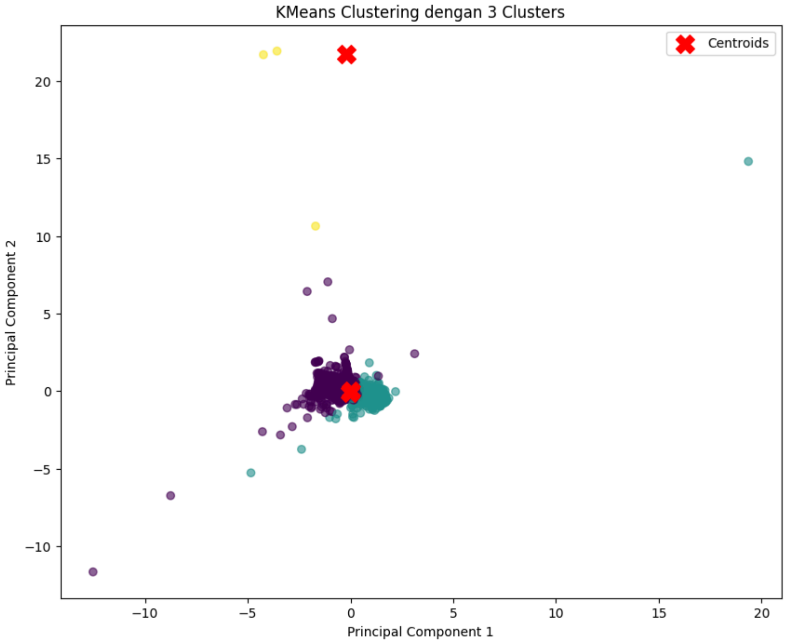
```
# Subsampling X untuk Elbow & Silhouette
max_samp = 2000
if X.shape[0] > max_samp:
    idx = np.random.choice(X.shape[0], size=max_samp, replace=False)
    X_samp = X[idx]
else:
    X_samp = X

# Menentukan jumlah cluster optimal dengan Elbow Method
inertia = []
silhouette_scores = []
k_range = range(2, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)
    if k > 1:  # Silhouette score memerlukan minimal 2 cluster
        silhouette_scores.append(silhouette_score(X, kmeans.labels_))

# Plot Elbow Method
plt.figure(figsize=(14, 5))

plt.plot(list(k_range), silhouette_scores, 'o-')
plt.title('Silhouette Score')
plt.xlabel('Jumlah Cluster (k)')
plt.ylabel('Silhouette Score')
plt.grid(True)
plt.show()

plt.subplot(1, 2, 2)
plt.plot(list(k_range), silhouette_scores, 'o-')
plt.title('Silhouette Score')
plt.xlabel('Jumlah Cluster (k)')
plt.ylabel('Silhouette Score')
plt.grid(True)

plt.tight_layout()
```

```
plt.show()

# Tentukan k optimal berdasarkan Elbow Method dan Silhouette
k_optimal = 3  # Ganti dengan nilai sesuai analisis grafik

# Fit KMeans dengan k optimal
kmeans = KMeans(n_clusters=k_optimal, random_state=42, n_init=10)
kmeans.fit(X)
clusters_kmeans = kmeans.labels_

# Evaluasi model KMeans
silhouette_kmeans = silhouette_score(X, clusters_kmeans)
davies_bouldin_kmeans = davies_bouldin_score(X, clusters_kmeans)
print(f"\nEvaluasi KMeans (k={k_optimal}):")
print(f"Silhouette Score: {silhouette_kmeans:.4f}")
print(f"Davies-Bouldin Index: {davies_bouldin_kmeans:.4f}")

# Visualisasi hasil KMeans pada data yang telah di-PCA
plt.figure(figsize=(10, 8))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters_kmeans, cmap='viridis', alpha=0.6)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            c='red', marker='X', s=200, label='Centroids')
plt.title(f'KMeans Clustering dengan {k_optimal} Clusters')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```



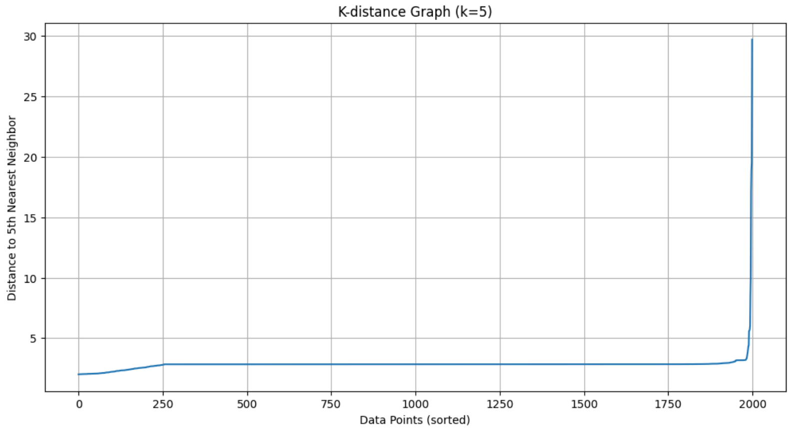



```
Evaluasi KMeans (k=3):
Silhouette Score: 0.1179
Davies-Bouldin Index: 1.7719
```

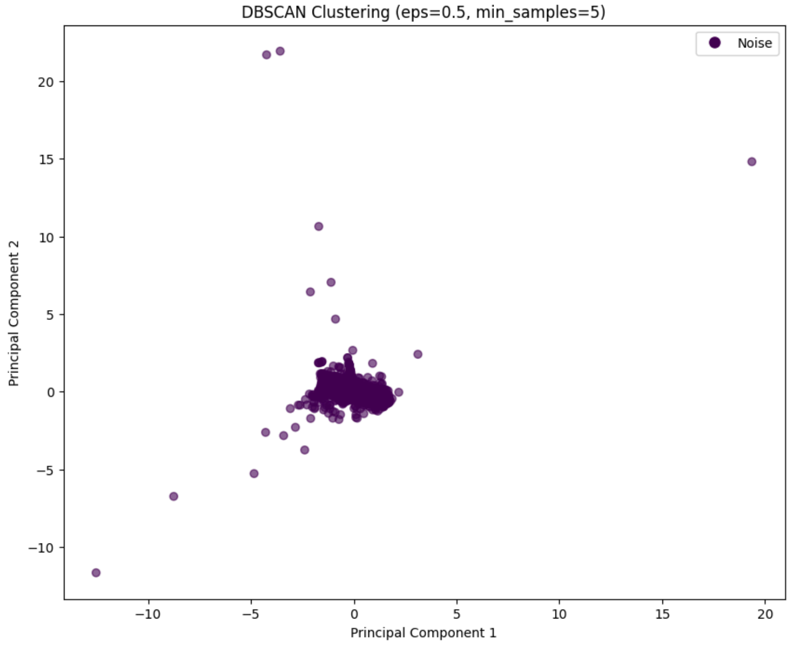


CLUSTERING DENGAN DBSCAN

```
# Mencari epsilon optimal dengan k-distance graph
from sklearn.neighbors import NearestNeighbors

# Ambil sampel jika data terlalu besar
X_sample = X if X.shape[0] < 5000 else X[np.random.choice(X.shape[0], 5000, replace=F

# Compute k-distances
k = 5  # jumlah tetangga
neigh = NearestNeighbors(n_neighbors=k)
neigh.fit(X_sample)
distances, indices = neigh.kneighbors(X_sample)
distances = np.sort(distances[:, k-1])

# Plot k-distance graph
plt.figure(figsize=(12, 6))
plt.plot(distances)
plt.title('K-distance Graph (k=5)')
plt.xlabel('Data Points (sorted)')
plt.ylabel('Distance to 5th Nearest Neighbor')
plt.grid(True)
plt.show()

# Berdasarkan k-distance graph, pilih epsilon yang sesuai
epsilon = 0.5  # Ganti dengan nilai berdasarkan analisis k-distance graph
min_samples = 5  # Minimal points untuk membentuk cluster

# Fit DBSCAN
dbscan = DBSCAN(eps=epsilon, min_samples=min_samples)
clusters_dbscan = dbscan.fit_predict(X)

# Evaluasi DBSCAN
n_clusters = len(set(clusters_dbscan)) - (1 if -1 in clusters_dbscan else 0)
n_noise = list(clusters_dbscan).count(-1)
print(f"\nEvaluasi DBSCAN (eps={epsilon}, min_samples={min_samples}):")
print(f"Jumlah Cluster: {n_clusters}")
print(f"Jumlah Noise Points: {n_noise} ({n_noise/len(clusters_dbscan)*100:.2f}%)")

if n_clusters > 1:  # Silhouette score hanya valid jika ada lebih dari 1 cluster
    # Hitung silhouette score tanpa noise points
    mask = clusters_dbscan != -1
    if sum(mask) > 1:  # Pastikan ada lebih dari 1 data point setelah menghapus noise
        silhouette_dbscan = silhouette_score(X[mask], clusters_dbscan[mask])
        print(f"Silhouette Score (tanpa noise): {silhouette_dbscan:.4f}")

# Visualisasi hasil DBSCAN pada data yang telah di-PCA
plt.figure(figsize=(10, 8))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters_dbscan, cmap='viridis', al
plt.title(f'DBSCAN Clustering (eps={epsilon}, min_samples={min_samples})')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')

# Tambahkan legend manual untuk noise points
unique_clusters = set(clusters_dbscan)
if -1 in unique_clusters:
    unique_clusters.remove(-1)
    unique_clusters = list(unique_clusters)
    unique_clusters = [-1] + sorted(unique_clusters)
    legend_elements = [plt.Line2D([0], [0], marker='o', color='w',
                                  markerfacecolor=scatter.cmap(scatter.norm(c)),
                                  markersize=10, label=f'Cluster {c}' if c != -1 else
                      for c in unique_clusters]
    plt.legend(handles=legend_elements)
plt.show()
```




```
Evaluasi DBSCAN (eps=0.5, min_samples=5):
Jumlah Cluster: 0
Jumlah Noise Points: 2000 (100.00%)
```

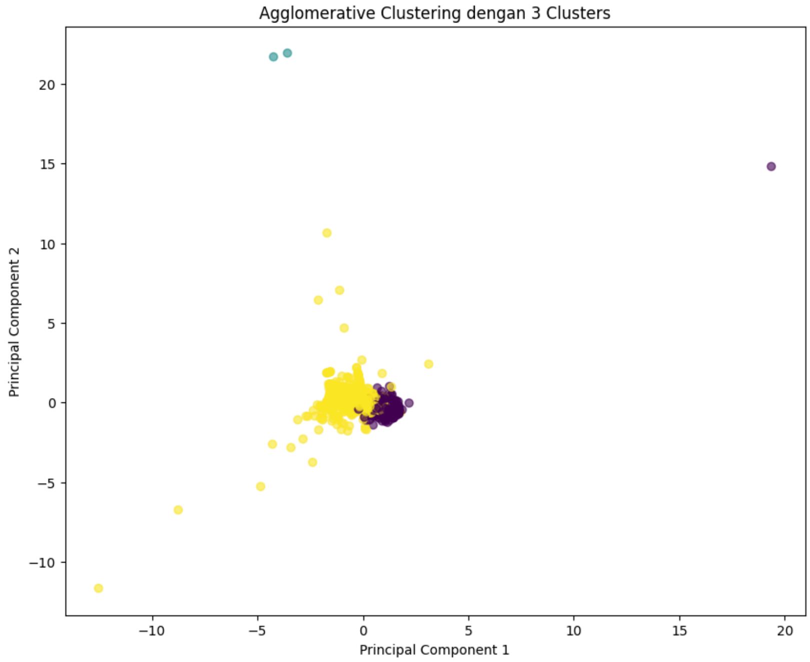


HIERARCHICAL CLUSTERING (AgglomerativeClustering)

```
# Fit Agglomerative Clustering
agg_clustering = AgglomerativeClustering(n_clusters=k_optimal)
clusters_agg = agg_clustering.fit_predict(X)

# Evaluasi Agglomerative Clustering
silhouette_agg = silhouette_score(X, clusters_agg)
davies_bouldin_agg = davies_bouldin_score(X, clusters_agg)
print(f"\nEvaluasi Agglomerative Clustering (n_clusters={k_optimal}):")
```

```
print(f"Silhouette Score: {silhouette_agg:.4f}")
print(f"Davies-Bouldin Index: {davies_bouldin_agg:.4f}")

# Visualisasi hasil Agglomerative Clustering pada data yang telah di-PCA
plt.figure(figsize=(10, 8))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters_agg, cmap='viridis', alpha=0.6)
plt.title(f'Agglomerative Clustering dengan {k_optimal} Clusters')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()

# Plot dendrogram (untuk dataset kecil)
if X.shape[0] <= 100:  # Hanya tampilkan jika data cukup kecil
    from scipy.cluster.hierarchy import dendrogram, linkage

    linkage_matrix = linkage(X[:100], method='ward')
    plt.figure(figsize=(15, 8))
    dendrogram(linkage_matrix)
    plt.title('Hierarchical Clustering Dendrogram (100 sampel pertama)')
    plt.xlabel('Sample index')
    plt.ylabel('Distance')
    plt.show()
```

```
Evaluasi Agglomerative Clustering (n_clusters=3):
Silhouette Score: 0.1046
Davies-Bouldin Index: 1.7359
```

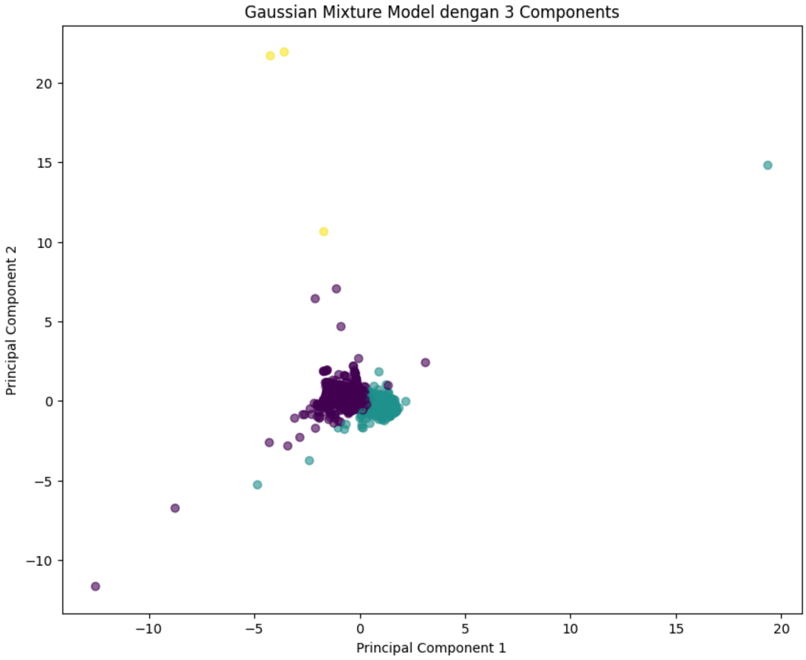


GAUSSIAN MIXTURE MODEL

```
# Fit Gaussian Mixture Model
gmm = GaussianMixture(n_components=k_optimal, random_state=42)
clusters_gmm = gmm.fit_predict(X)

# Evaluasi GMM
silhouette_gmm = silhouette_score(X, clusters_gmm)
print(f"\nEvaluasi Gaussian Mixture Model (n_components={k_optimal}):")
print(f"Silhouette Score: {silhouette_gmm:.4f}")
print(f"BIC: {gmm.bic(X)}")
print(f"AIC: {gmm.aic(X)}")

# Visualisasi hasil GMM pada data yang telah di-PCA
plt.figure(figsize=(10, 8))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters_gmm, cmap='viridis', alpha=0.6)
plt.title(f'Gaussian Mixture Model dengan {k_optimal} Components')
```

```
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

```
Evaluasi Gaussian Mixture Model (n_components=3):
Silhouette Score: 0.1179
BIC: 263318951.312971
AIC: -17405387.504499316
```



PERBANDINGAN SEMUA MODEL

```
# Tambahkan cluster labels ke dataframe asli
df_results = pd.DataFrame({
    'KMeans': clusters_kmeans,
    'DBSCAN': clusters_dbscan,
    'Agglomerative': clusters_agg,
    'GMM': clusters_gmm
})

# Summary hasil clustering
print("\nPerbandingan Hasil Clustering:")
print(f"{'Model':<20} {'Jumlah Cluster':<15} {'Silhouette Score':<20}")
print("-" * 55)

models = {
    'KMeans': {'clusters': len(set(clusters_kmeans)), 'silhouette': silhouette_kmear
    'DBSCAN': {'clusters': n_clusters, 'silhouette': silhouette_dbscan if n_clusters
    'Agglomerative': {'clusters': len(set(clusters_agg)), 'silhouette': silhouette_a
    'GMM': {'clusters': len(set(clusters_gmm)), 'silhouette': silhouette_gmm}
}

for model_name, metrics in models.items():
    print(f"{model_name:<20} {metrics['clusters']:<15} {metrics['silhouette'] if is:
```

```
Perbandingan Hasil Clustering:
Model                Jumlah Cluster  Silhouette Score
-------------------------------------------------------
KMeans               3               0.11788439704342094
DBSCAN               0               N/A
Agglomerative        3               0.1045709495632277
GMM                  3               0.11788439704342094
```