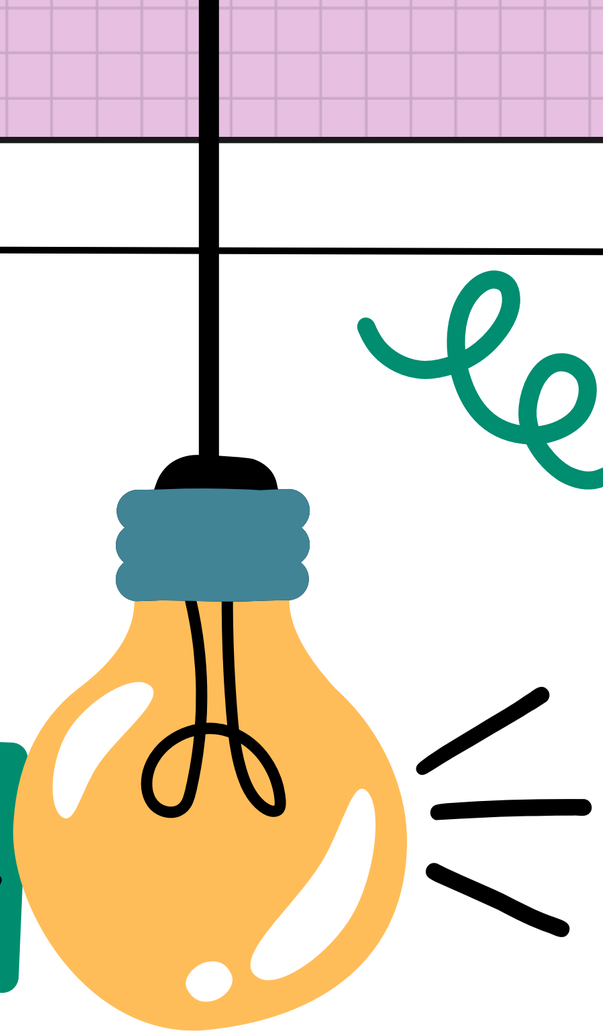
LAPORAN PRAKTIKUM MACHINE LEARNING

AUTOMATIC CLUSTERING



Dosen Pengampu : Nur Rosyid Mubtadai S.Kom., M.T.

Bayu Kurniawan / 3322600019

Praktikum Automatic Clustering

Penjelasan:

- Variance Within Clusters (VWC) adalah ukuran sebaran atau variasi dari setiap cluster dalam suatu metode pengelompokan data, seperti k-means clustering. VWC mengestimasi sebaran observasi di dalam setiap cluster, dengan nilai yang lebih kecil menunjukkan cluster dengan sebaran rendah atau lebih padat. (Semakin kecil nilai varians ini, semakin rapat atau homogen data dalam klaster tersebut)

- Variance Between Clusters (VBC) adalah ukuran sebaran atau variasi antara cluster dalam suatu metode pengelompokan data. VBC mengukur seberapa jauh kelompok-kelompok tersebut terpisah satu sama lain. (Semakin besar nilai varians antar klaster, semakin berbeda atau terpisah klaster-klaster tersebut)

- Total varians atau varians keseluruhan dari dataset adalah jumlah dari varians antar klaster dan varians dalam klaster. Ini mencerminkan sejauh mana data dalam keseluruhan dataset tersebar atau berbeda.

Dalam analisis pengelompokan data, tujuan utama adalah untuk mencapai VWC yang rendah dan VBC yang tinggi. Hal ini menunjukkan bahwa observasi di dalam setiap cluster saling berdekatan dan terpisah dengan jelas antara cluster-cluster yang berbeda. Jika VWC tinggi dan VBC rendah, ini dapat menunjukkan bahwa pengelompokan tidak efektif dalam mengelompokkan data dengan baik.

AUTOMATIC CLUSTERING TANPA MENGGUNAKAN LIBRARY

Praktikum Automatic Clustering

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
import random

# Fungsi untuk menghitung jarak antara dua titik
def euclidean_distance(point1, point2):
    return np.linalg.norm(point1 - point2)

# Fungsi untuk menginisialisasi centroid secara acak
def initialize_centroids(data, k):
    centroids = random.sample(list(data), k)
    return centroids

# Fungsi untuk mengelompokkan data ke dalam cluster
def assign_to_clusters(data, centroids):
    clusters = {}
    for point in data:
        closest_centroid = min(centroids, key=lambda centroid: euclidean_distance(point, centroid))
        if closest_centroid in clusters:
            clusters[closest_centroid].append(point)
        else:
            clusters[closest_centroid] = [point]
    return clusters

# Fungsi untuk menghitung centroid baru untuk setiap cluster
def update_centroids(clusters):
    new_centroids = []
    for centroid, cluster_points in clusters.items():
        new_centroid = np.mean(cluster_points, axis=0)
        new_centroids.append(new_centroid)
    return new_centroids

# Fungsi untuk menentukan apakah konvergensi telah tercapai
def has_converged(old_centroids, new_centroids, tolerance=1e-4):
    return all(euclidean_distance(old, new) < tolerance for old, new in zip(old_centroids, new_centroids))

# Fungsi K-Means
def k_means(data, k):
    centroids = initialize_centroids(data, k)
    while True:
        clusters = assign_to_clusters(data, centroids)
        new_centroids = update_centroids(clusters)
        if has_converged(centroids, new_centroids):
            return clusters, centroids
        centroids = new_centroids
```

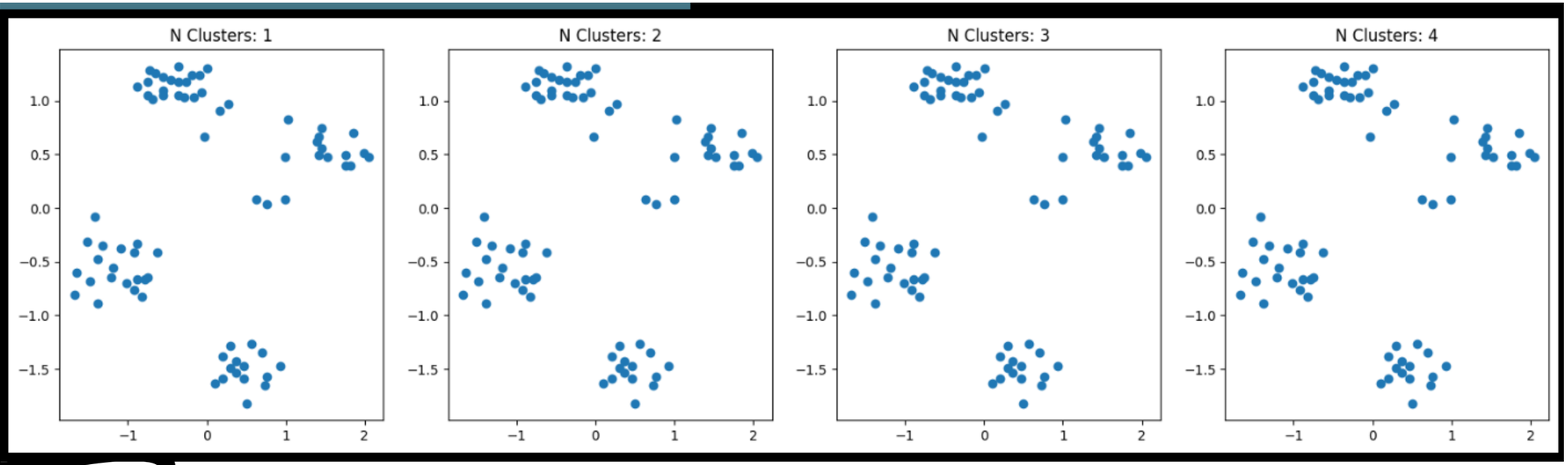
```python
# Fungsi untuk mengelompokkan data ke dalam cluster
def assign_to_clusters(data, centroids):
    clusters = {}
    for point in data:
        closest_centroid = min(centroids, key=lambda centroid: euclidean_distance(point, centroid))
        # Konversi centroid menjadi tuple
        closest_centroid_tuple = tuple(closest_centroid)
        if closest_centroid_tuple in clusters:
            clusters[closest_centroid_tuple].append(point)
        else:
            clusters[closest_centroid_tuple] = [point]
    return clusters

dataset = pd.read_csv("C:/Users/bayuk/OneDrive/Documents/AI/pens/smtr3/Machine Learning/Data/ruspini.csv")
dataset = dataset.fillna(dataset.groupby("CLASS").transform("mean"))

# Memilih atribut yang akan digunakan
selected_features = ['X','Y']

# Memisahkan fitur dan target
X = dataset[selected_features]
y = dataset['CLASS']

# Melakukan normalisasi data
scaler = StandardScaler()
data_norm = scaler.fit_transform(X)

# Membuat dataframe pandas dari data ternormalisasi
df = pd.DataFrame(data_norm, columns=['X', 'Y'])  # Sesuaikan jumlah kolom dengan data Anda
df['CLASS'] = y

data = df.loc[:,['X','Y']]

# Contoh penggunaan K-Means
if __name__ == '__main__':
    # Data
    data = np.array(data)

    # Jumlah cluster
    k = 4  # jumlah cluster

    # Centroid yang telah diinisialisasi (random)
    centroids = random.sample(list(data), k)

    # Memanggil fungsi assign_to_clusters
    clusters = assign_to_clusters(data, centroids)

    # Menjalankan K-Means
    clusters, final_centroids = k_means(data, k)
```

Praktikum Automatic Clustering

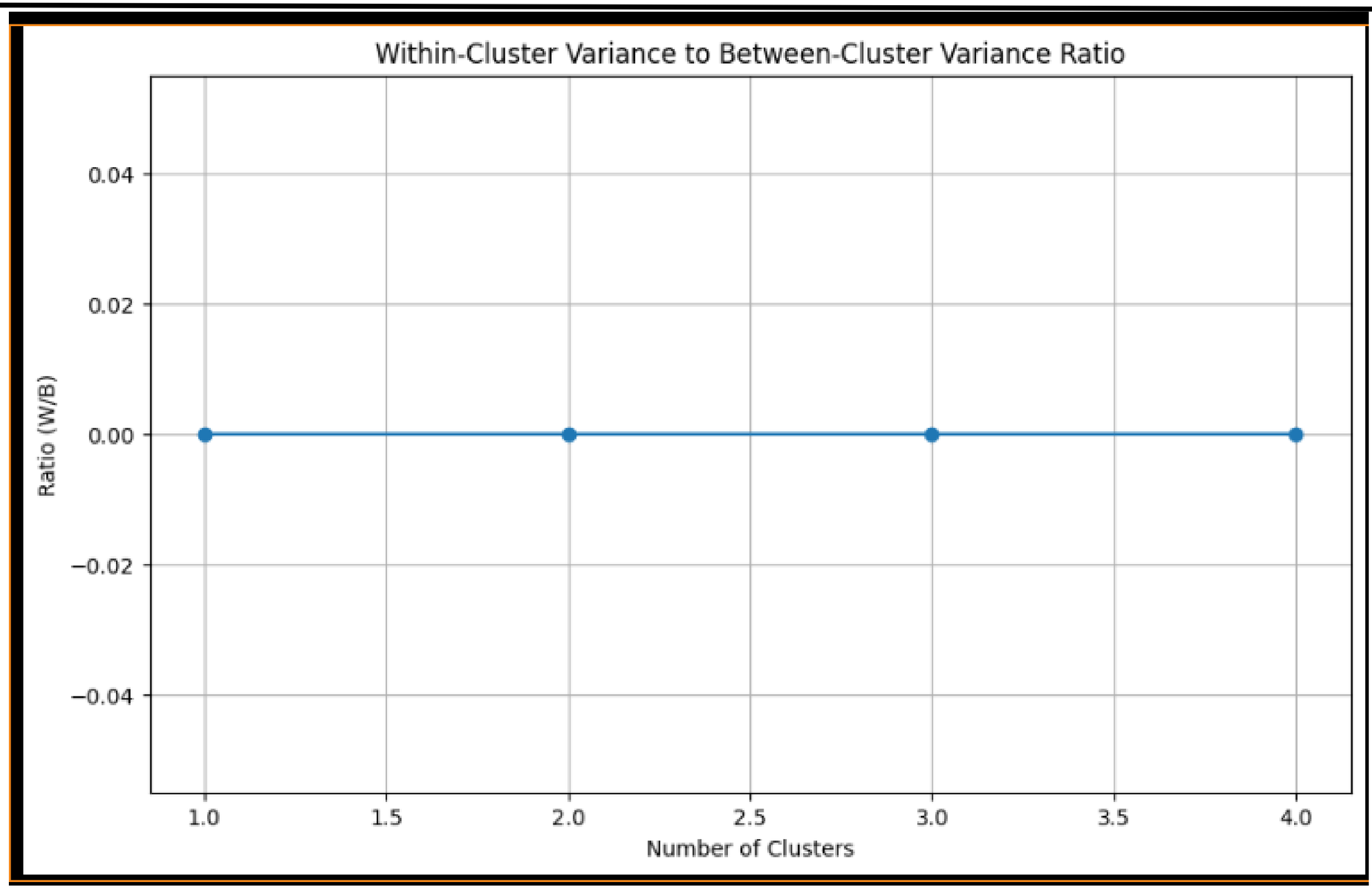
```
    # Menampilkan hasil clustering
    for i, centroid in enumerate(final_centroids, start=1):
        print(f"Centroid Cluster {i}: {centroid}")
    for centroid, cluster_points in clusters.items():
        print(f'Cluster dengan centroid {centroid}:')
        print(cluster_points)

    print('Final Centroids:')
    print(final_centroids)

    satu = "X"
    dua = "Y"

    fig, axs = plt.subplots(nrows=1,ncols=k,figsize=(20,5))
    for i, ax in enumerate(fig.axes,start=1):
        ax.scatter(x=df[satu], y=df[dua])
        ax.set_title(f'N Clusters: {i}')

# Menghitung error, variance between, dan variance within
errors = []
variance_between = []
variance_within = []

for k_test in range(1, 5):  # Menguji dari 1 hingga 4 cluster
    kmeans = k_means(data, k_test)
    clusters, final_centroids = k_means(data, k_test)

    # Mendapatkan centroid akhir dan label cluster
    cluster_centers = final_centroids
    labels = []

    for point in data:
        closest_centroid = min(cluster_centers, key=lambda centroid: euclidean_distance(point, centroid))
        label = np.where(cluster_centers == closest_centroid)[0][0]
        labels.append(label)

    error = 0
    variance_within_clusters = []  # Menyimpan variance within setiap cluster

    for i in range(k_test):
        cluster_points = data[labels == i]
        if len(cluster_points) > 1:
            variance_within_cluster = np.var(cluster_points, axis=0, ddof=1).sum()  # ddof=1 untuk menghitung variance dengan benar
            variance_within_clusters.append(variance_within_cluster)
        else:
            variance_within_clusters.append(0.0)  # Cluster dengan 1 titik memiliki variance 0

    error = sum(variance_within_clusters)  # Total variance within adalah jumlah variance dalam setiap cluster

    # Menghitung variance between clusters
    global_center = data.mean(axis=0)
    squared_distances = [np.linalg.norm(cluster_centers[i] - global_center) ** 2 for i in range(k_test)]
    variance_between_clusters = sum(squared_distances)

    errors.append(error)
    variance_between.append(variance_between_clusters)
    variance_within.append(error)

# Print the errors, variance between, and variance within
print("Number of Clusters  |  Variance Between  |  Variance Within")
for i in range(4):
    print(f"         {i+1:2d}         |  {variance_between[i]:15.2f} | {variance_within[i]:14.2f}")

# Hitung rasio within-cluster variance dibagi between-cluster variance
ratios = [variance_within[i] / variance_between[i] for i in range(4)]

# Jumlah cluster
cluster_numbers = list(range(1, 5))

# Visualisasi rasio
plt.figure(figsize=(10, 6))
plt.plot(cluster_numbers, ratios, marker='o', linestyle='-')
plt.title('Within-Cluster Variance to Between-Cluster Variance Ratio')
plt.xlabel('Number of Clusters')
plt.ylabel('Ratio (W/B)')
plt.grid(True)
plt.show()
```

Praktikum Automatic Clustering

```
Centroid Cluster 1: [-1.14626147 -0.55970301]
Centroid Cluster 2: [1.4289973  0.47245093]
Centroid Cluster 3: [-0.36196371  1.11658397]
Centroid Cluster 4: [ 0.46382939 -1.50126913]
Cluster dengan centroid (-1.1462614715333053, -0.5597030099830441):
[array([-1.67929121, -0.806722  ]), array([-1.64628627, -0.60001155]), array([-1.48126158, -0.68269573]), array([-1.51426652, -0.31061691]), array([-1.38224677, -0.88940619]), array([-1.38224677, -0.47
Cluster dengan centroid (1.4289972959593735, 0.4724509320351327):
[array([1.02711365, 0.82629059]), array([0.99410871, 0.47488282]), array([0.99410871, 0.08213295]), array([0.76307415, 0.04079086]), array([0.6310544 , 0.08213295]), array([1.39016796, 0.61958014]), ar
Cluster dengan centroid (-0.3619637120889858, 1.1165839728780877):
[array([-0.88717271,  1.13635628]), array([-0.75515296,  1.17769837]), array([-0.65613815,  1.26038255]), array([-0.72214803,  1.2810536 ]), array([-0.55712334,  1.21904046]), array([-0.45810853,  1.19
Cluster dengan centroid (0.46382938516022776, -1.5012691347421598):
[array([ 0.49903465, -1.81960324]), array([ 0.73006921, -1.65423487]), array([ 0.92809884, -1.46819546]), array([ 0.20199022, -1.59222174]), array([ 0.46602971, -1.59222174]), array([ 0.76307415, -1.57
Final Centroids:
[array([-1.14626147, -0.55970301]), array([1.4289973 , 0.47245093]), array([-0.36196371,  1.11658397]), array([ 0.46382939, -1.50126913])]
Number of Clusters  |  Variance Between  |  Variance Within
         1       |            0.00 |          0.00
         2       |            2.01 |          0.00
         3       |            4.78 |          0.00
         4       |            7.74 |          0.00
```



Praktikum Automatic Clustering



AUTOMATIC CLUSTERING MENGGUNAKAN LIBRARY

Praktikum Automatic Clustering

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import numpy as np

def kolomn(dataset):
    print("\nKolom yang tersedia dalam dataset:")
    for col in dataset.columns:
        print(col)

def optimise_k_means(data, max_k):
    means = []
    inertias = []

    for k in range(1, max_k + 1):
        kmeans = KMeans(n_clusters=k)
        kmeans.fit(data)

        means.append(k)
        inertias.append(kmeans.inertia_)

    fig, ax = plt.subplots(figsize=(10, 5))
    ax.plot(means, inertias, "o-")
    ax.set_xlabel('Number of Clusters')
    ax.set_ylabel('Inertia')
    ax.grid(True)

print("""<!> INI ADALAH PROGRAM AUTOMATIC CLUSTERING (Kmeans)<!>\n""")

data = str(input("Masukkan nama data yang digunakan: "))
k_cluster = int(input("Masukkan banyak cluster yang diinginkan: "))

dataset = pd.read_csv(data+".csv")
column = []
kolomn(dataset)
while True:
    kolom = input("Masukkan kolom yang ingin di drop: ")
    column.append(kolom)
    pilihan = input("Ingin mengulang?")
    if pilihan == 't':
        break
dataset = dataset.drop(columns=column)
dataset = dataset.fillna(dataset.mean())
dataset = dataset.iloc[:,:]
var_bet = np.array(dataset)
sc = StandardScaler()
var_df = dataset.iloc[:,:]
df = dataset.iloc[:,:]
df_tf = sc.fit_transform(df)
```

```python
kolomn(dataset)

optimise_k_means(df, k_cluster)
kmeans = KMeans(k_cluster)
kmeans.fit(df.iloc[:,:])
df[f'KMeans_{k_cluster}'] = kmeans.labels_

centroids = kmeans.cluster_centers_
for i, centroid in enumerate(centroids, start=1):
    print(f"Centroid Cluster {i}: {centroid}")

for k in range (1,k_cluster+1):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(var_df)
    df[f'KMeans_{k}'] = kmeans.labels_

satu = input("Masukkan Nama kolom yang ingin divisualisasi: ")
dua = input("Masukkan Nama kolom yang ingin divisualisasi: ")
fig, axs = plt.subplots(nrows=1,ncols=k_cluster,figsize=(20,5))
for i, ax in enumerate(fig.axes,start=1):
    ax.scatter(x=df[satu], y=df[dua], c=df[f'KMeans_{i}'])
    ax.set_title(f'N Clusters: {i}')

    # Calculate error, variance between, and variance within
errors = []
variance_between = []
variance_within = []

for k in range(1, k_cluster + 1):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(var_df)

    cluster_centers = kmeans.cluster_centers_
    labels = kmeans.labels_

    error = 0
    for i in range(k):
        global_center = np.mean(var_df, axis=0)
        cluster_points = var_df[labels == i].iloc[:,:]
        error += ((cluster_points - cluster_centers[i]) ** 2).sum().sum()

# Hitung jarak antara pusat cluster dan pusat cluster global
        distances = np.linalg.norm(cluster_centers[i] - global_center, axis=0)
# Kuadratkan jarak
        squared_distances = distances**2
```

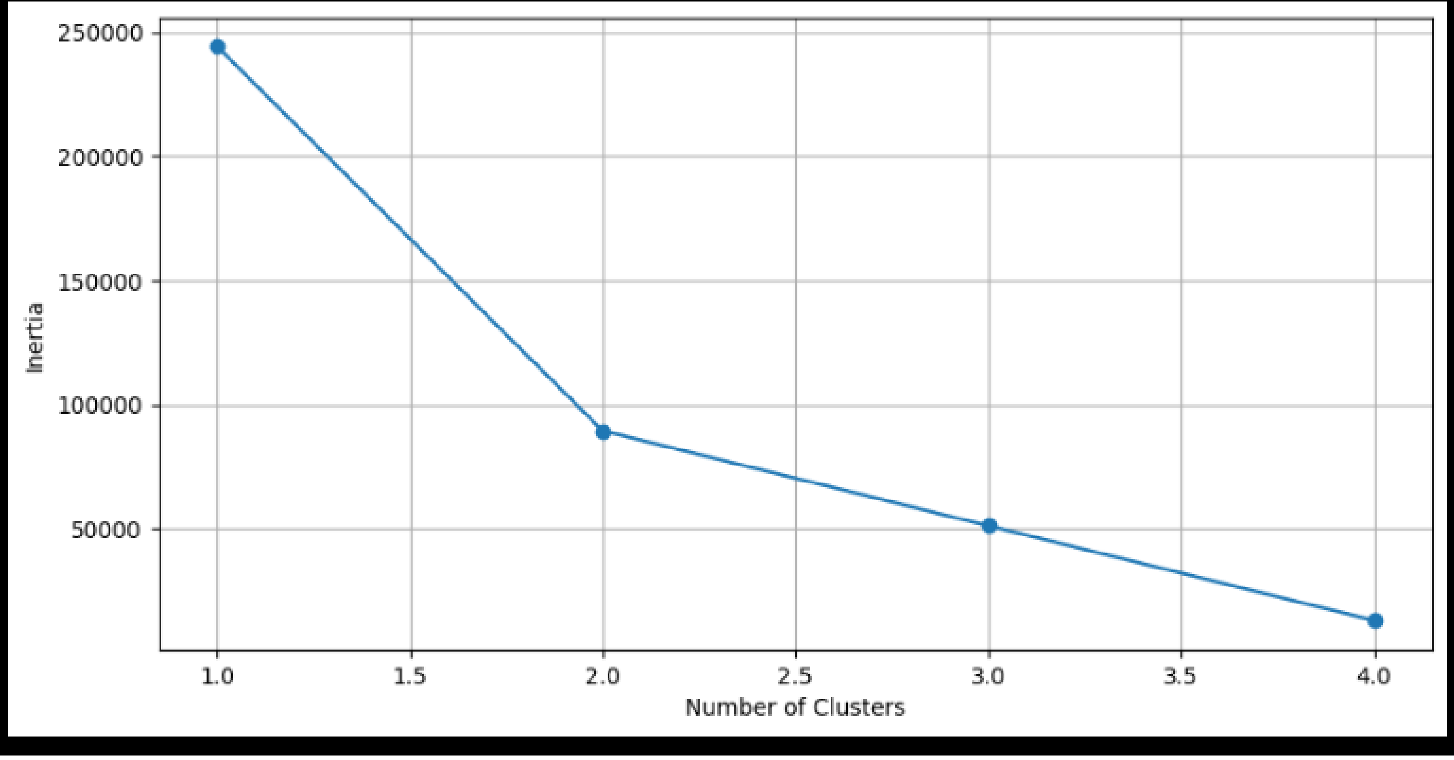
Praktikum Automatic Clustering

```
        total_error = ((var_df - cluster_centers[labels]) ** 2).sum().sum()
        within_cluster_var = error

            # Hitung variance between clusters
        variance_between_clusters = np.sum(squared_distances)

        errors.append(total_error)
        variance_between.append(variance_between_clusters)
        variance_within.append(within_cluster_var)

        # Print the errors, variance between, and variance within
    print("Number of Clusters  |  Variance Between  |  Variance Within")
    for k in range(k_cluster):
        print(f"        {k+1:2d}        |  {variance_between[k]:15.2f} | {variance_within[k]:14.2f}")

    # Hitung rasio within-cluster variance dibagi between-cluster variance
    ratios = [variance_within[i] / variance_between[i] for i in range(k_cluster)]

    # Jumlah cluster
    cluster_numbers = list(range(1, k_cluster + 1))

    # Visualisasi rasio
    plt.figure(figsize=(10, 6))
    plt.plot(cluster_numbers, ratios, marker='o', linestyle='-')
    plt.title('Within-Cluster Variance to Between-Cluster Variance Ratio')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Ratio (W/B)')
    plt.grid(True)
    plt.show()
```

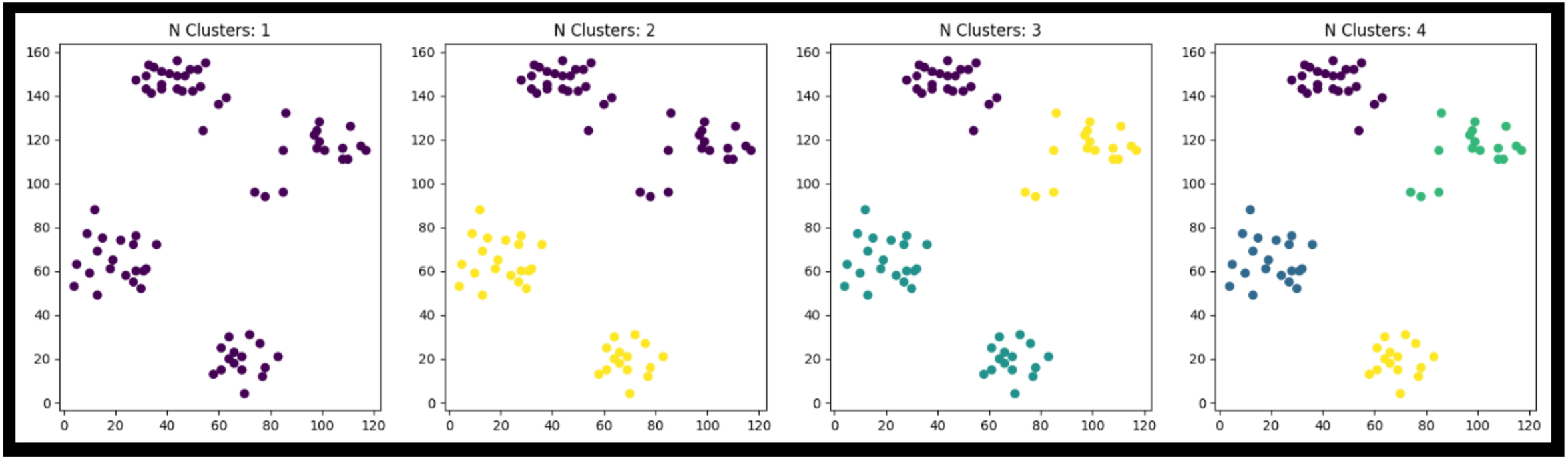
```
<!> INI ADALAH PROGRAM AUTOMATIC CLUSTERING (Kmeans)<!>


Kolom yang tersedia dalam dataset:
#
X
Y
CLASS

Kolom yang tersedia dalam dataset:
X
Y
```

```
Centroid Cluster 1: [ 43.91304348 146.04347826]
Centroid Cluster 2: [68.93333333 19.4       ]
Centroid Cluster 3: [ 98.17647059 114.88235294]
Centroid Cluster 4: [20.15 64.95]
C:\Users\bayuk\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packa
  warnings.warn(
C:\Users\bayuk\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packa
  warnings.warn(
C:\Users\bayuk\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packa
  warnings.warn(
C:\Users\bayuk\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packa
  warnings.warn(
```

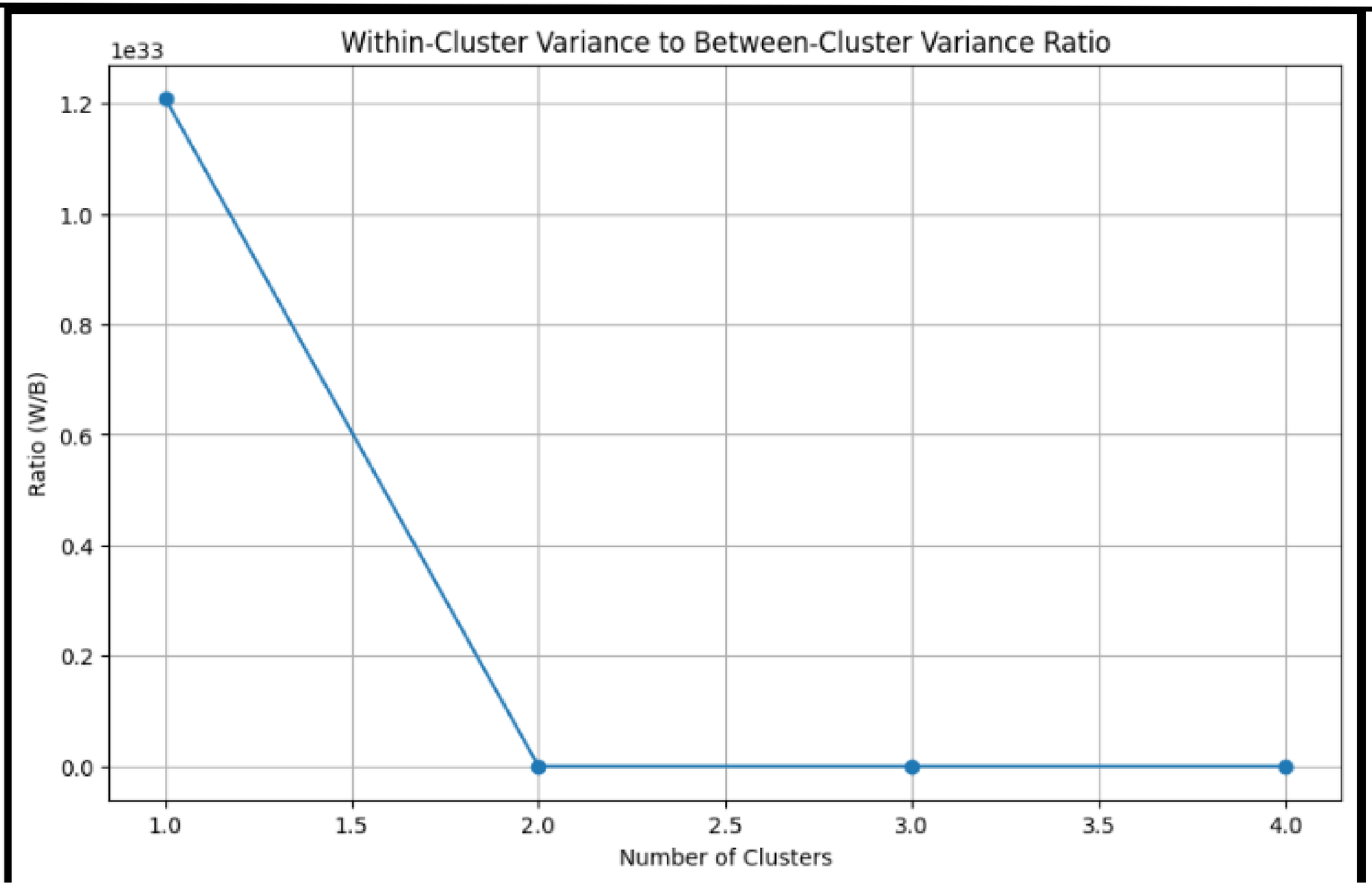
Number of Clusters	Variance Between	Variance Within
1	0.00	244524.37
2	2363.03	89450.32
3	2398.81	51097.76
4	5475.24	12881.05



Praktikum Automatic Clustering



Praktikum Automatic Clustering



THANK YOU