BAB 4
HASIL PENELITIAN

4.1 Tahap Persiapan Dataset

Tahap persiapan dataset merupakan langkah awal dalam proses penelitian yang penting dalam proses penelitian ini karena kualitas dan struktur data yang digunakan akan sangat memengaruhi dataset yang akan digunakan sebagai dasar analisis dalam mendeteksi anomali jaringan. Dataset yang digunakan adalah UNSW-NB15, yang dikembangkan oleh *Australian Centre for Cyber Security* (ACCS) sebagai penelitian keamanan jaringan. Dataset ini dirancang untuk merepresentasikan lalu lintas jaringan modern, baik yang bersifat normal maupun mengandung serangan siber. Dalam penelitian ini, dipilih pendekatan *unsupervised learning*, sehingga keseluruhan data akan digunakan tanpa mempertimbangkan label kelas saat proses klasterisasi.

Dalam penelitian ini, dilakukan proses penggabungan data pelatihan (*training*) dan data pengujian (*testing*) dari dataset UNSW-NB15. Proses ini bertujuan untuk menghasilkan data utuh yang merepresentasikan berbagai kondisi lalu lintas jaringan secara menyeluruh. Berikut kode penggabungan data train dan data test.

```
# Load dataset training dan testing
      data_train = pd.read_csv('/content/drive/MyDrive/Penelitian
Binta/training.csv')
      data_test = pd.read_csv('/content/drive/MyDrive/Penelitian
Binta/testing.csv')
      # Gabungkan dataset training dan testing
      data_full = pd.concat([data_train,data_test],
ignore_index=True)
```

Proses penggabungan dilakukan dalam format *Comma-Separated Values* (CSV) dan diolah menggunakan bahasa pemrograman Python pada Google Collab. Dengan pendekatan *unsupervised learning*, penggabungan data menjadi penting agar tidak ada pemisahan berdasarkan label yang bisa mempengaruhi proses

klasterisasi. Data hasil gabungan ini nantinya akan digunakan pada seluruh tahapan *preprocessing* dan pemodelan.

Hasil dari proses penggabungan menghasilkan total 257.673 baris data dengan 45 atribut, yang terdiri dari fitur numerik dan kategorikal. Fitur-fitur ini mencakup informasi seperti ukuran paket, beban lalu lintas, waktu koneksi, dan tipe protokol jaringan. Berikut kode mengenai jumlah data full dari UNSW NB15.

```
# Menampilkan jumlah baris dan kolom dari data_full
data_full.shape
```

```
(257673, 45)
```

Fokus dari penelitian ini pada deteksi serangan *Denial of Service* (DoS), yaitu jenis serangan yang bertujuan untuk membuat layanan tidak tersedia bagi pengguna yang sah dengan cara membanjiri sistem target menggunakan lalu lintas data yang berlebihan atau tidak valid. Serangan DoS dipilih sebagai fokus karena merupakan salah satu jenis serangan yang paling umum dan berbahaya dalam konteks keamanan jaringan, serta dapat menyebabkan gangguan layanan yang signifikan. Dalam dataset UNSW-NB15, serangan DoS terdapat 16.353 baris data, jumlah yang cukup representatif untuk dianalisis lebih lanjut menggunakan metode deteksi anomali berbasis klasterisasi. Banyaknya data ini memungkinkan identifikasi pola lalu lintas jaringan yang mencerminkan aktivitas anomali secara lebih akurat. Berikut kode menampilkan jumlah data dalam serangan DoS.

```
# Data full dan kolom kategori serangan adalah 'attack_cat'
data_dos = data_full[data_full['attack_cat'] == 'DoS']
# Menapilkan jumlah baris data DoS
print("Jumlah data DoS:", data_dos.shape[0])
```

```
Jumlah data DoS: 16353
```

4.2 Preprocessing Data

Tahapan preprocessing pada data serangan *Denial of Service* (DoS) menggunakan file CSV merupakan langkah awal yang sangat penting dalam menyiapkan data agar dapat digunakan secara maksimal dalam proses analisis dan pemodelan. Tahapan ini bertujuan untuk membersihkan dan menyusun data secara sistematis, sehingga dapat diproses lebih lanjut dengan metode analisis yang sesuai.

Proses ini mencakup penanganan data kosong (*missing values*), penghapusan data duplikat, serta seleksi terhadap fitur-fitur yang relevan. Tanpa tahapan *preprocessing* yang baik, kualitas data yang masih mengandung *noise* atau struktur tidak konsisten dapat mempengaruhi akurasi deteksi, terutama dalam konteks identifikasi anomali seperti serangan *Denial of Service* (DoS). Dalam dataset DoS, terdapat berbagai tipe data, yaitu data numerik (angka), data kategorikal (berupa teks), dan data biner. Namun, karena algoritma klasterisasi yang digunakan dalam penelitian ini dengan data numerik, maka seluruh fitur non-numerik perlu dikonversi terlebih dahulu. Oleh karena itu, proses *preprocessing* juga mencakup transformasi tipe data non numerik menjadi numerik agar seluruh data dapat diolah secara efisien dan menghasilkan model deteksi anomali yang lebih akurat.

4.2.1 Cleaning Data

Pada tahap ini, preprocessing diterapkan pada dataset UNSW-NB15 yang sebelumnya telah digabung antara data pelatihan dan data pengujian. Tahapan pertama adalah pembersihan data (*cleaning* data), yang mencakup penanganan nilai hilang (*missing values*) dan penghapusan data duplikat. Hasil pemeriksaan terhadap data yang berkaitan dengan serangan DoS menunjukkan bahwa tidak ditemukan nilai yang hilang pada fitur-fitur yang digunakan. Oleh karena itu, tidak diperlukan proses imputasi atau penghapusan data pada tahap ini. Berikut kode untuk mengetahui total *missing value*.

```
# Cek total missing value
total_missing = data_dos.isnull().sum().sum()
print("Total missing value dalam dataset:", total_missing)
```

```
Total missing value dalam dataset: 0
```

Selain itu, dilakukan juga pemeriksaan terhadap data duplikat. Duplikat data dapat menyebabkan ketidakseimbangan pembobotan pada proses klasterisasi dan mempengaruhi hasil pembelajaran model. Oleh karena itu, sistem melakukan identifikasi baris yang identik untuk memastikan validitas data. Hasil pemeriksaan menunjukkan bahwa tidak terdapat baris duplikat dalam subset data serangan DoS. Hal ini memperkuat kualitas data yang digunakan. Berikut kode untuk cek duplikat.

```
# Cek total duplikat
total_duplikat = data_dos.isnull().sum().sum()
print("Total duplikat dalam dataset:", total_duplikat)
```

```
Total duplikat dalam dataset: 0
```

Langkah berikutnya dalam tahap *preprocessing* adalah memastikan bahwa seluruh fitur yang digunakan dalam bersifat numerik, karena algoritma *unsupervised learning* seperti K-Means dan DBSCAN hanya dapat memproses data dalam format numerik. Berdasarkan hasil pengecekan tipe data pada subset data serangan DoS, ditemukan bahwa masih terdapat kombinasi antara fitur numerik dan non-numerik. Berikut kode untuk melakukan cek data non numerik.

```
# Cek tipe data per kolom
dtypes = data_dos.dtypes
# Menghitung jumlah kolom numerik dan non-numerik
num_cols = dtypes[dtypes.apply(lambda x: x in ['int64',
'float64'])].index
non_num_cols = dtypes[dtypes.apply(lambda x: x not in ['int64',
'float64'])].index
# Menampilkan hasilnya
print("Jumlah kolom numerik     :", len(num_cols))
print("Jumlah kolom non-numerik :", len(non_num_cols))
# Mampilkan nama-nama kolom non-numerik
print("\nKolom non-numerik:")
print(non_num_cols.tolist())
```

```
Jumlah kolom numerik     : 41
Jumlah kolom non-numerik : 4
Kolom non-numerik:
['proto', 'service', 'state', 'attack_cat']
```

Untuk menghindari kesalahan atau gangguan dalam proses klasterisasi, seluruh fitur non-numerik kemudian dikonversi secara eksplisit menjadi tipe numerik, baik dalam bentuk *float* maupun *integer*, sesuai dengan karakteristik

masing-masing data. Proses konversi ini bertujuan untuk menyelaraskan dengan menggunakan kode berikut.

```
# Mengubah kolom non-numerik menjadi numerik
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in data_dos.columns:
    if data_dos[col].dtype == 'object':
        data_dos[col] = le.fit_transform(data_dos[col])
```

```
data_dos.info()
```

```
Data columns (total 45 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 16353 non-null  int64
 1   dur                16353 non-null  float64
 2   proto              16353 non-null  int64
 3   service            16353 non-null  int64
 4   state              16353 non-null  int64
 5   spkts              16353 non-null  int64
 6   dpkts              16353 non-null  int64
 7   sbytes             16353 non-null  int64
 8   dbytes             16353 non-null  int64
 9   rate               16353 non-null  float64
 10  sttl               16353 non-null  int64
 11  dttl               16353 non-null  int64
 12  sload              16353 non-null  float64
 13  dload              16353 non-null  float64
 14  sloss              16353 non-null  int64
 15  dloss              16353 non-null  int64
 16  sinpkt             16353 non-null  float64
 17  dinpkt             16353 non-null  float64
 18  sjit               16353 non-null  float64
```

```
 19  djit               16353 non-null  float64
 20  swin               16353 non-null  int64
 21  stcpb              16353 non-null  int64
 22  dtcpb              16353 non-null  int64
 23  dwin               16353 non-null  int64
 24  tcprtt             16353 non-null  float64
 25  synack             16353 non-null  float64
 26  ackdat             16353 non-null  float64
 27  smean              16353 non-null  int64
 28  dmean              16353 non-null  int64
 29  trans_depth        16353 non-null  int64
 30  response_body_len  16353 non-null  int64
 31  ct_srv_src         16353 non-null  int64
 32  ct_state_ttl       16353 non-null  int64
 33  ct_dst_ltm         16353 non-null  int64
 34  ct_src_dport_ltm   16353 non-null  int64
 35  ct_dst_sport_ltm   16353 non-null  int64
 36  ct_dst_src_ltm     16353 non-null  int64
 37  is_ftp_login       16353 non-null  int64
 38  ct_ftp_cmd         16353 non-null  int64
 39  ct_flw_http_mthd   16353 non-null  int64
 40  ct_src_ltm         16353 non-null  int64
 41  ct_srv_dst         16353 non-null  int64
 42  is_sm_ips_ports    16353 non-null  int64
 43  attack_cat         16353 non-null  int64
 44  label              16353 non-null  int64
dtypes: float64(11), int64(34)
```

4.2.2 Normalisasi Data

Setelah memastikan bahwa seluruh fitur bersifat numerik, langkah selanjutnya adalah normalisasi data. Normalisasi bertujuan untuk menyetarakan skala nilai antar fitur, karena perbedaan skala dapat menyebabkan bias terhadap

fitur yang memiliki nilai lebih besar. Dalam penelitian ini digunakan metode *Min-Max Scaling*. Berikut kode untuk normalisasi data.

```
# Inisialisasi objek MinMaxScaler
scaler = MinMaxScaler()
# Normalisasi seluruh fitur dalam data_dos
data_dos_scaled = scaler.fit_transform(data_dos)
# Konversi kembali ke DataFrame agar mudah digunakan
data_dos_scaled = pd.DataFrame(data_dos_scaled,
columns=data_dos.columns)
# Tampilkan beberapa baris pertama hasil normalisasi
print(data_dos_scaled.head())
```

Dengan metode ini, nilai yang mendekati 0 kemiripan atau kesamaan antar data dalam satu *cluster* rendah/terkecil, sedangkan yang mendekati 1 menunjukkan bahwa kemiripan atau kesamaan antar data dalam satu *cluster* sangat tinggi. Hasil normalisasi ini penting agar semua fitur memiliki kontribusi yang seimbang dalam proses klasterisasi. Hasil normalisasi data ditunjukkan pada gambar 4.1.

```
         id       dur     proto  service  state     spkts     dpkts    sbytes  \
0  0.000000  0.015367  0.585938      0.0   0.50  0.002192  0.000000  0.000108
1  0.000234  0.019555  0.851562      0.0   0.25  0.001731  0.001089  0.000177
2  0.000514  0.023971  0.851562      0.5   0.25  0.001038  0.000908  0.000080
3  0.000526  0.024424  0.851562      0.5   0.25  0.001038  0.000908  0.000067
4  0.000646  0.470231  0.585938      0.0   0.50  0.002192  0.000000  0.000108

     dbytes         rate  ...  ct_dst_sport_ltm  ct_dst_src_ltm  \
0  0.000000  2.060767e-05  ...               0.0        0.016129
1  0.000119  2.301276e-05  ...               0.0        0.000000
2  0.000186  1.321062e-05  ...               0.0        0.000000
3  0.000069  1.296571e-05  ...               0.0        0.000000
4  0.000000  6.734450e-07  ...               0.0        0.000000

   is_ftp_login  ct_ftp_cmd  ct_flw_http_mthd  ct_src_ltm  ct_srv_dst  \
0           0.0         0.0          0.000000         0.0         0.0
1           0.0         0.0          0.000000         0.0         0.0
2           0.0         0.0          0.166667         0.0         0.0
3           0.0         0.0          0.166667         0.0         0.0
4           0.0         0.0          0.000000         0.0         0.0

   is_sm_ips_ports  attack_cat  label
0              0.0         0.0    0.0
1              0.0         0.0    0.0
2              0.0         0.0    0.0
3              0.0         0.0    0.0
4              0.0         0.0    0.0

[5 rows x 45 columns]
```

Gambar 4.1 Normalisasi Data

4.2.3 Pemilihan Fitur

Pemilihan fitur merupakan tahap penting dalam *preprocessing* data yang berperan secara langsung terhadap efektivitas model dalam mengidentifikasi pola anomali pada data. Dalam penelitian ini, proses pemilihan fitur bertujuan untuk menentukan atribut-atribut yang paling relevan dalam membedakan lalu lintas normal dan lalu lintas yang mengidentifikasi serangan *Denial of Service* (DoS). Fitur yang dipilih diharapkan dapat merepresentasikan karakteristik utama dari aktivitas jaringan yang mencurigakan atau menyimpang dari pola normal.

Pemilihan hanya lima fitur dalam penelitian ini didasarkan pada efektivitas dan efisiensi dalam proses deteksi anomali, khususnya untuk mendeteksi serangan *Denial of Service* (DoS). Berdasarkan analisis korelasi antar fitur dalam dataset UNSW-NB15, kelima fitur numerik yang dipilih dinilai memiliki kontribusi signifikan dalam merepresentasikan karakteristik lalu lintas jaringan yang berkaitan dengan serangan DoS. Penggunaan terlalu banyak fitur justru dapat menimbulkan risiko *overfitting*, yaitu ketika model terlalu kompleks sehingga menangkap *noise* atau informasi yang tidak relevan. Selain itu, jumlah fitur yang berlebihan juga dapat memperlambat proses komputasi dan mengurangi efisiensi sistem. Oleh karena itu, pemilihan lima fitur dianggap sebagai langkah untuk menjaga akurasi deteksi, mengurangi kompleksitas model, serta memastikan proses analisis tetap efisien. Kelima fitur tersebut dijelaskan lebih lanjut pada tabel 4.1.

Tabel 4.1 Fitur yang Terpilih

No	Nama Fitur	Deskripsi
1.	stcpb	Source TCP base sequence number, menunjukkan nomor urut awal sesi TCP dari sisi sumber.
2.	dtcpb	Destination TCP base sequence number, menunjukkan nomor urut awal sesi TCP dari sisi tujuan.
3.	sload	Source bits per second, mengindikasikan kecepatan lalu lintas bit yang dikirim dari sumber.
4.	dload	Destination bits per second, menunjukkan beban lalu lintas yang diterima oleh tujuan dalam satuan bit per detik.
5.	sbytes	Bytes from source to destination, menggambarkan total jumlah byte yang dikirim dari sumber ke tujuan.

Kelima fitur tersebut dipilih karena memiliki keterkaitan langsung dengan pola lalu lintas data yang terjadi dalam serangan DoS. Serangan jenis ini ditandai dengan lalu lintas tinggi, seragam, dan berlangsung secara terus-menerus dalam waktu singkat.

Selain disajikan dalam bentuk tabel penjelasan, hasil dari proses pemilihan fitur divisualisasikan. Pada tahap ini terpilih kolom *stcpb, dtcpb, sload, dload,* dan *sbytes* yang diambil dari dataset DoS yang telah melalui proses dinormalisasi. Berikut kode seleksi fitur.

```
# Pilih lima fitur numerik yang relevan untuk deteksi serangan
DoS
selected_features = ['stcpb', 'dtcpb', 'sload', 'dload',
'sbytes']

# Ambil subset dari data hasil normalisasi berdasarkan fitur yang
dipilih
data_selected = data_dos_scaled[selected_features]

# Tampilkan 5 baris pertama hasil feature selection
print(data_selected.head())
```

```
['stcpb', 'dtcpb', 'sload', 'dload', 'sbytes']
```


4.3 Tahap Pengujian Algoritma K-Means

K-Means digunakan terlebih dahulu karena lebih cepat membentuk *cluster* awal yang bisa divisualisasikan dan dianalisis. Hasil ini digunakan sebagai dasar untuk deteksi anomali lebih lanjut dengan DBSCAN yang lebih sensitif terhadap *noise* dan *outlier* dalam penelitian ini, menggunakan algoritma K-Means sebagai metode *unsupervised learning* untuk melakukan proses klasterisasi terhadap data jaringan. Algoritma ini bekerja dengan cara mengelompokkan data ke dalam beberapa *cluster* berdasarkan kemiripan nilai dari fitur-fitur yang telah dipilih sebelumnya. Setiap data akan dimasukkan ke dalam *cluster* yang memiliki pusat (*centroid*) terdekat, sehingga data dalam satu *cluster* memiliki karakteristik yang serupa. Proses ini dilakukan tanpa memerlukan label atau kategori yang telah ditentukan sebelumnya, sehingga sangat sesuai digunakan dalam mendeteksi pola-pola anomali seperti serangan DoS.

Langkah awal dalam klasterisasi menggunakan K-Means adalah menentukan jumlah *cluster*, yang disebut dengan nilai k. Untuk menentukan nilai k yang terbaik, digunakan matrik evaluasi *Silhouette Score*, yaitu ukuran yang menunjukkan seberapa sesuai suatu data berada di dalam *cluster* tempat berada dibandingkan dengan *cluster* lainnya. Nilai *Silhouette Score* berada dalam rentang -1 hingga 1. Semakin mendekati 1, semakin baik pemisahan antar *cluster* dan semakin jelas data dalam masing-masing *cluster*. Dalam penelitian ini, proses evaluasi dilakukan terhadap nilai k, mulai dari 2 hingga 8. Nilai k yang menghasilkan *Silhouette Score* tertinggi dipilih sebagai jumlah *cluster* yang akan digunakan dalam proses klasterisasi akhir. Pendekatan ini bertujuan untuk menghasilkan pengelompokan data yang paling *representatif* terhadap karakteristik serangan DoS dalam dataset. Berikut kode untuk nilai *Silhouette Score* untuk masing-masing nilai k.

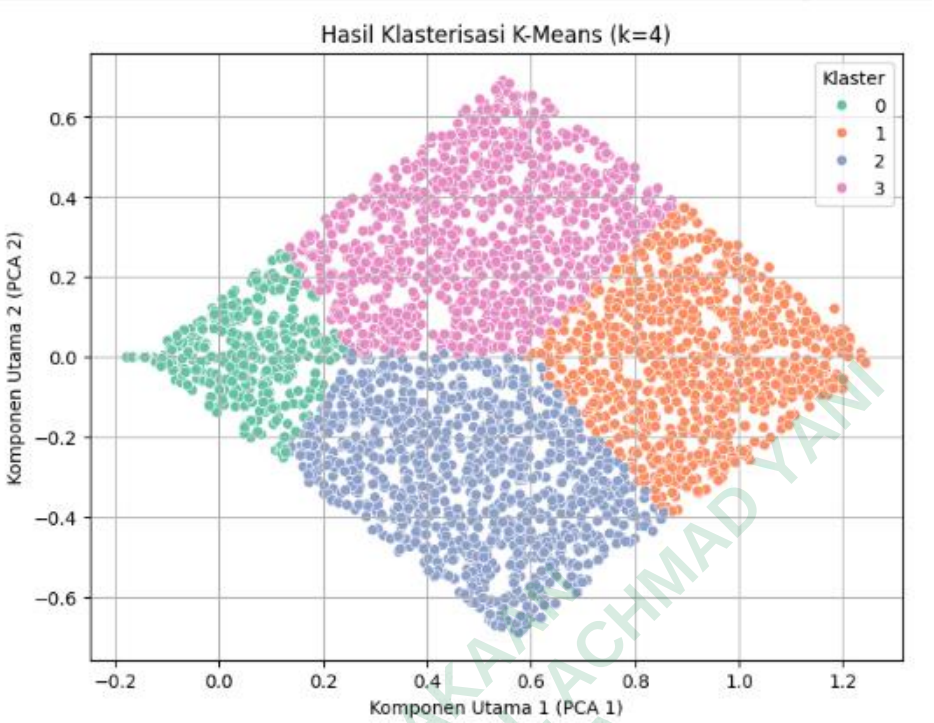
```
# Menentukan jumlah klaster (k) terbaik untuk algoritma K-Means
# Dengan cara mengevaluasi Silhouette Score dari setiap jumlah
klaster
silhouette_scores = []
for k in range(2, 8):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    score = silhouette_score(X_scaled, kmeans.labels_)
    silhouette_scores.append(score)
    print(f'Cluster: {k}, Silhouette Score: {score:.4f}')
```

```
Cluster: 2, Silhouette Score: 0.8370
Cluster: 3, Silhouette Score: 0.8323
Cluster: 4, Silhouette Score: 0.8377
Cluster: 5, Silhouette Score: 0.8191
Cluster: 6, Silhouette Score: 0.8206
Cluster: 7, Silhouette Score: 0.8215
```

Berdasarkan hasil perhitungan *Silhouette Score* untuk setiap nilai k, terlihat bahwa kualitas pemisahan data antar *cluster* mengalami perubahan seiring bertambahnya jumlah klaster. Ditemukan bahwa nilai k = 4 merupakan *cluster* yang

paling tepat, karena menghasilkan *Silhouette Score* tertinggi sebesar 0,8377. Nilai ini menunjukkan bahwa pemisahan data kedalam empat *cluster* memberikan struktur *cluster* yang paling baik, dengan pemisahan antar *cluster* yang jelas dan tingkat kemiripan yang tinggi di dalam masing-masing *cluster*. Data dalam satu *cluster* memiliki karakteristik yang signifikan, sementara perbedaannya cukup jauh dengan data dari *cluster* lain. Hal ini mencerminkan bahwa pemilihan empat *cluster* mampu merepresentasikan pola-pola serangan DoS dengan baik, sehingga sangat sesuai untuk digunakan dalam proses deteksi anomali berbasis klasterisasi.

Hasil klasterisasi dengan nilai k = 4 yang merupakan *cluster* yang paling tepat karena menghasilkan nilai *Silhouette Score* tertinggi sebesar 0,8377 yang menunjukkan pemisahan *cluster* yang sangat baik. Untuk memvisualisasikan hasil *cluster* menggunakan *Principal Component Analysis* (PCA). Teknik ini digunakan untuk mereduksi lima fitur *stcpb, dtcpb, sload, dload,* dan *sbytes* menjadi dua komponen utama agar dapat divisualisasikan dalam bentuk grafik dua dimensi. Sumbu X (PCA 1) merepresentasikan kombinasi linier dari kelima fitur yang memiliki variasi atau informasi paling besar dan menjadi dimensi yang paling membedakan pola antar *cluster*. Sementara itu, sumbu Y (PCA 2) menunjukkan kombinasi fitur dengan variasi terbesar kedua dan bersifat ortogonal (tegak lurus) terhadap PCA 1, sehingga mewakili aspek penting lainnya dalam pemisahan data. Setiap titik dalam grafik menggambarkan satu sesi komunikasi jaringan, dan warna yang berbeda menunjukkan *cluster* hasil pengelompokan oleh algoritma K-Means. Keempat *cluster* diberi label 0 hingga 3, dan divisualisasikan pada gambar 4.2 untuk mempermudah interpretasi pola anomali jaringan.



Gambar 4.2 Visualisasi Menggunakan *PCA*

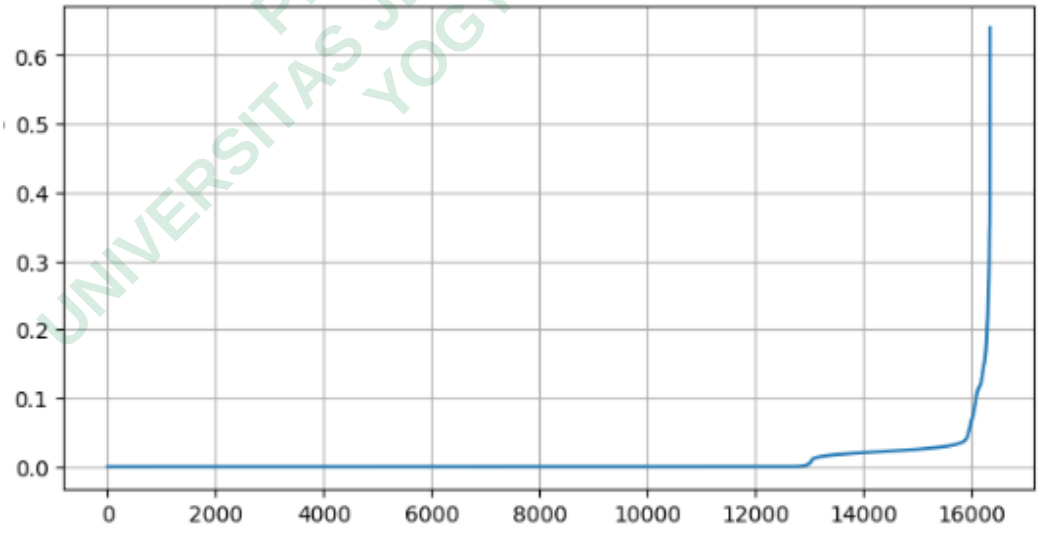
Hasil dari distribusi data dalam visualisasi ini menunjukkan bahwa keempat *cluster* terbentuk memiliki batas pemisah yang relatif jelas dan tidak saling tumpang tindih, yang menunjukkan keberhasilan algoritma K-Means menggunakan nilai k = 4 dalam memisahkan data berdasarkan karakteristik perilaku lalu lintas jaringan. Setiap *cluster* merepresentasikan jenis lalu lintas tertentu yang memiliki pola tersendiri, baik dari sisi volume data, kecepatan transfer dan karakteristik protokol TCP semuanya dalam fitur-fitur yang dipilih. Struktur *cluster* semacam ini tidak hanya berguna untuk menggambarkan pola umum lalu lintas jaringan, tetapi juga sangat berpotensi dalam mendeteksi anomali yang tidak sesuai dengan pola *cluster* yang terbentuk.

Dengan struktur seperti ini, klasterisasi menggunakan algoritma K-Means berguna untuk memetakan pola umum dalam lalu lintas jaringan, serta berguna sebagai dasar awal untuk mendeteksi anomali yang kemudian akan divalidasi lebih lanjut menggunakan algoritma DBSCAN. Oleh karena itu, hasil klasterisasi ini menunjukkan keberhasilan teknik K-Means dalam pembentukan kelompok.

4.4 Tahap Pengujian DBSCAN

Pengujian algoritma DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) dilakukan setelah proses klasterisasi awal menggunakan algoritma K-Means. Dalam penelitian ini, K-Means dan DBSCAN tidak digunakan secara terpisah, melainkan saling melengkapi dalam satu rangkaian proses klasterisasi. Algoritma K-Means digunakan terlebih dahulu untuk membentuk struktur *cluster* awal berdasarkan pola distribusi umum dalam data, sedangkan DBSCAN berperan dalam menyaring data yang menyimpang atau tidak sesuai dengan kepadatan *cluster*, sehingga dapat diidentifikasi sebagai anomali.

Kombinasi metode ini digunakan untuk mendeteksi anomali secara lebih efektif karena mampu mengenali baik pola data normal maupun *outlier*. Algoritma DBSCAN mengelompokkan data berdasarkan dua parameter utama, yaitu *epsilon* (ε) dan minimum points (*minPts*). Parameter *epsilon* (ε) menentukan jarak maksimum antar titik dalam ruang fitur (yang sebelumnya telah dinormalisasi), agar dua titik dapat dianggap sebagai tetangga. Penentuan nilai epsilon dilakukan dengan menggunakan metode *elbow* (titik siku) pada grafik jarak tetangga terdekat (*k-distance graph*), yang divisualisasikan pada gambar 4.3.



Gambar 4.3 Visualisasi Penentuan Nilai Epsilon

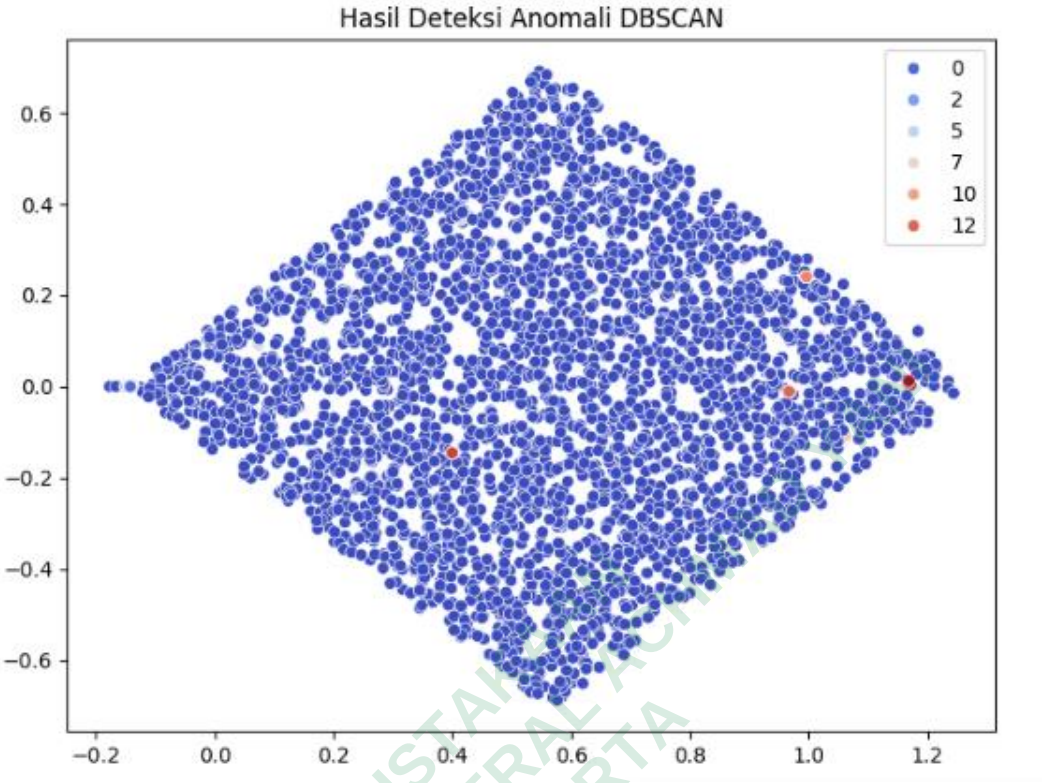
Pada penelitian ini, nilai epsilon ditetapkan sebesar 0.01, yang diperoleh dari rata-rata dari k=4 untuk mendapatkan titik yang sesuai dengan skala data hasil normalisasi, sehingga titik-titik yang memiliki jarak dekat yang dapat membentuk

cluster. Sementara itu, parameter *minPts* ditetapkan sebesar 5, yang merupakan nilai umum yang sering digunakan dalam DBSCAN untuk menentukan jumlah minimal tetangga yang diperlukan agar suatu titik dianggap sebagai bagian dari *cluster* yang padat. Dengan konfigurasi parameter ini, DBSCAN dapat secara efektif memisahkan *cluster* padat sebagai anomali.

Setelah DBSCAN dijalankan dengan parameter, dilanjutkan dengan identifikasi antara data normal dan data anomali. DBSCAN secara otomatis memberikan label -1 untuk data yang dianggap sebagai anomali. Berikut kode untuk identifikasi nilai normal dan anomali.

```
# Label -1 adalah anomali
anomali = np.sum(dbscan_labels == -1)
# Menghitung jumlah data yang dianggap normal
normal = np.sum(dbscan_labels != -1)
print(f'Jumlah Anomali: {anomali}, Normal: {normal}')
Jumlah Anomali: 3281, Normal: 13072
```

Berdasarkan hasil deteksi terhadap serangan DoS sebanyak 16.353 data, ditemukan bahwa 3.281 data termasuk kategori anomali, sedangkan 13.072 data termasuk kategori normal. Sedangkan data yang berhasil dikelompokkan ke dalam *cluster* akan diberi label 0, 2, 5 dan seterusnya, sesuai jumlah *cluster* yang terbentuk yang merepresentasikan lalu lintas jaringan yang dianggap normal. Visualisasi hasil deteksi tersebut ditampilkan pada gambar 4.4.



Gambar 4.4 Visualisasi Normal dan Anomali

Dalam visualisasi, titik-titik berwarna biru tua mendominasi, yang merepresentasikan data normal yang berhasil dikelompokkan ke dalam *cluster* mayoritas. Sementara itu, titik-titik berwarna oranye hingga merah menggambarkan anomali atau data yang masuk dalam klaster minoritas dengan jumlah anggota yang sangat sedikit. Label 0, 2, 5, 7, 10, dan 12 menunjukkan *cluster* yang terbentuk berdasarkan hasil pemrosesan DBSCAN.

4.5 Tahap Evaluasi Model

Tahap evaluasi model dilakukan untuk mengukur efektivitas kombinasi algoritma K-Means dan DBSCAN dalam mendeteksi anomali pada sistem deteksi intrusi (IDS). Evaluasi ini bertujuan untuk memastikan bahwa hasil klasterisasi dan deteksi anomali yang diperoleh Dalam penelitian ini, evaluasi model menggunakan *Silhouette Score* dan menggunakan *False Positive Rate* (FPR).

4.5.1 Silhouette Score

Silhouette Score digunakan sebagai matrik evaluasi, tanpa memerlukan label asli karena perhitungan mengandalkan jarak antar data dan kohesi dalam *cluster*. Nilai *Silhouette Score* berada pada rentang antara -1 hingga 1, dengan nilai yang lebih mendekati 1 menunjukkan bahwa data berada pada *cluster* yang tepat dan *cluster* yang terbentuk saling terpisah dengan baik. Dalam penelitian ini, *Silhouette Score* pada DBSCAN dihitung hanya untuk data yang berhasil dikelompokkan ke dalam *cluster*, sedangkan data yang dianggap noise atau anomali tidak ikut dihitung karena tidak memiliki *cluster*.

Hasil perhitungan *Silhouette Score* menunjukkan nilai 0.3116, bahwa struktur *cluster* yang terbentuk memiliki kualitas pemisahan yang cukup baik. Meskipun nilai ini tidak tergolong tinggi, namun masih menunjukkan bahwa *cluster* yang terbentuk cukup representatif dalam menggambarkan distribusi data DoS secara umum.

4.5.2 False Positive Rate

False Positive Rate (FPR) merupakan matrik evaluasi terhadap data normal sebagai anomali (*false alarm*). Nilai FPR dihitung menggunakan perbandingan antara data normal yang salah sebagai anomali (*false positive*). FPR merupakan indikator krusial dalam sistem deteksi intrusi, karena tingkat kesalahan dalam mendeteksi lalu lintas normal dapat menyebabkan alarm palsu dan mengganggu operasi sistem.

Hasil evaluasi menunjukkan bahwa nilai FPR = 0, yang berarti tidak terdapat data normal yang salah diklasifikasikan sebagai anomali. Nilai ini menunjukkan bahwa kombinasi algoritma K-Means dan DBSCAN efektif dalam mengenali lalu lintas jaringan yang normal, sehingga mampu menekan risiko terjadinya *false alarm*.