BAB 4
HASIL PENELITIAN

4.1 Ringkasan Hasil Penelitian

Penelitian ini mengembangkan dan mengevaluasi metode deteksi anomali pada lalu lintas jaringan, khususnya serangan DDoS menggunakan pendekatan Threshold-Based Detection berbasis algoritma DBSCAN. Setelah melalui tahap pengkajian permasalahan, tahap selanjutnya diawali dengan *data collection & extraction* menggunakan dataset CIC-IDS2017, yang kemudian diproses melalui *data cleaning*, *normalization* menggunakan *Min-Max Scaling*, *selection feature* berdasarkan karakteristik volume serangan DDoS, yaitu *Total Fwd Packets*, *Average Packet Size*, *Flow Bytes/s*, dan *Min Packet Length*. Data benign hasil *pre-processing* diklasterisasi menggunakan DBSCAN dengan parameter epsilon dan *MinPts* yang ditentukan melalui analisis K-Distance Plot serta divisualisasikan menggunakan PCA 2 dimensi.

Selanjutnya, untuk menghitung threshold dengan parameter rata-rata dan standar deviasi dengan koefisien *k* menggunakan *grid search*. Evaluasi dilakukan pada data uji seimbang antara benign dan DDoS menggunakan metrik *Accuracy*, *Precision*, *Recall*, *F1-Score*, dan *False Postive Rate* (FPR). Hasil penelitian menunjukkan bahwa metode yang diusulkan mampu mendeteksi serangan DDoS secara efektif dengan tingkat *false alarm* yang terkendali.

4.2 Tahap Awal

Pada tahap awal, dilakukan pengkajian terhadap permasalahan tingginya *false alarm* pada sistem IDS. Salah satu permasalahan yang diidentifikasi adalah keterbatasan IDS dalam membedakan lalu lintas normal dan serangan DDoS, terutama ketika karakteristik serangan menyerupai pola lalu lintas normal. Berdasarkan analisis tersebut, penelitian ini mengusulkan pendekatan Threshold-Based Detection menggunakan algoritma DBSCAN untuk mendeteksi serangan DDoS pada dataset CIC-IDS2017.

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

4.3 *Data Collection & Extraction*

Tahapan ini bertujuan untuk mengumpulkan dan menyiapkan data lalu lintas jaringan sebagai bahan pengujian dari metode yang diusulkan. Dataset yang digunakan dalam penelitian ini adalah CIC-IDS2017, yang dikembangkan oleh Canadian Institute for Cybersecurity (CIC), dimana dataset ini merepresentasikan berbagai jenis serangan dan lalu lintas normal (benign). Dalam konteks penelitian ini, difokuskan pada subset data yang merepresentasikan serangan DDoS (Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv. Distribusi aktivitas harian dan jenis serangan pada dataset CIC-IDS2017 ditunjukkan pada Tabel 4.1 berikut.

Tabel 4. 1 Distribusi dataset CIC-IDS2017

Nama File	Aktivitas	Jenis Serangan
Monday-WorkingHours.pcap_ISCX.csv	Monday	Benign (Normal human activites)
Tuesday-WorkingHours.pcap_ISCX.csv	Tuesday	Benign, FTP-Patator, SSH-Patator
Wednesday-workingHours.pcap_ISCX.csv	Wednesday	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Thursday	Benign, Web Attack – Brute Force, Web Attack – Sql Injection, Web Attack – XSS
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Thursday	Benign, Infiltration
Friday-WorkingHours-Morning.pcap_ISCX.csv	Friday	Benign, Bot
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Friday	Benign, PortScan
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Friday	Benign, DDoS

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Pada tahap ekstraksi, file dataset disimpan dalam format *csv* dan diunggah ke Google Drive agar dapat diakses secara langsung menggunakan Google Colab. Berikut adalah kode program dalam proses Mount Google Drive dan pembacaan file dataset.

```
# Mount Google Drive untuk akses file dataset
drive.mount('/content/drive')

# Load file csv DDoS dari CIC-IDS2017
data_ddos= pd.read_csv('/content/drive/MyDrive/Tugas
Akhir/Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv')
```

Setelah data berhasil dimuat, dilakukan analisis awal terhadap struktur dan kualitas data dengan kode program sebagai berikut.

```
# Informasi awal dataset
print(f"Ukuran data DDoS: {data_ddos.shape[0]} baris dan
{data_ddos.shape[1]} kolom")
print(f"Jumlah duplikat: {data_ddos.duplicated().sum()} baris")
print(f"Jumlah NaN: {data_ddos.isna().sum().sum()} nilai")
file_path = '/content/drive/MyDrive/Tugas Akhir/Friday-
WorkingHours-Afternoon-DDos.pcap_ISCX.csv'
size_mb= os.path.getsize(file_path) / (1024 * 1024)
print(f'Ukuran file: {size_mb :.2f} MB')
```

```
Ukuran data DDoS: 225745 baris dan 79 kolom
Jumlah duplikat: 2633 baris
Jumlah NaN: 4 nilai
Ukuran file: 73.55 MB
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

4.4 *Data Pre-Processing*

Tahapan *pre-processing* merupakan langkah penting dalam penelitian ini untuk memastikan bahwa data yang digunakan dalam proses *clustering* dan deteksi serangan DDoS memiliki kualitas yang baik. Data mentah dari lalu lintas jaringan umumnya memiliki duplikat, nilai kosong (*missing values*), serta fitur yang tidak

relevan untuk proses analisis. Hal tersebut dapat mengganggu performa algoritma *unsupervised learning* seperti DBSCAN dan dapat menurunkan efektivitas metode yang digunakan. Tahapan ini dilakukan melalui tiga langkah utama, yaitu: *data cleaning*, *normalization*, dan *feature selection*.

4.4.1 *Data Cleaning* (Pembersihan Data)

Pada tahap ini, dilakukan proses pembersihan dataset dari baris-baris dan fitur yang tidak memberikan kontribusi signifikan terhadap proses deteksi serangan DDoS. Langkah awal dilakukan untuk memastikan tidak ada kesalahan teknis dalam pemanggilan kolom, seperti spasi tersembunyi. Hal tersebut umum terjadi pada dataset yang berasal dari file *csv* yang memiliki format tidak konsisten. Berikut kode program untuk *striping* karakter *space* pada setiap nama kolom serta menghasilkan seluruh nama kolom yang bersih dan konsisten.

```
# Membersihkan nama kolom
data_ddos.columns = data_ddos.columns.str.strip()

# Menampilkan nama semua kolom
data_ddos.columns.tolist()
```

Data duplikat dapat menyebabkan bias pada proses analisis, terutama dalam algoritma DBSCAN yang sensitif terhadap kepadatan data. Nilai inf dan -inf dapat terjadi akibat pembagian angka nol dalam fitur tertentu, serta NaN (*Not a Number*) muncul akibat nilai yang hilang. Oleh karena itu, semua baris yang terduplikasi, inf, dan NaN dihapus dari dataset. Dengan ini, hanya data valid yang dipertahankan, untuk memastikan tidak ada eror saat proses normalisasi. Berikut kode program yang digunakan dalam tahap ini.

```
# Menghapus data duplikat
data_ddos.drop_duplicates(inplace=True)

# Membersihkan data (inf dan NaN)
data_ddos.replace([np.inf, -np.inf], np.nan, inplace=True)
data_ddos.dropna(inplace=True)
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Selain itu, fitur konstan (nilai tetap) dan *sparse* (mayoritas bernilai nol) tidak memberikan informasi berarti dalam penelitian ini. Proses ini membantu menyaring fitur yang tidak memiliki variasi statistik maupun distribusi signifikan terhadap klasifikasi lalu lintas jaringan, khususnya serangan DDoS menggunakan ambang batas sparsitas sebesar 95%.

Setelah fitur yang tidak informatif berhasil diidentifikasi, semua kolom tersebut dihapus dari dataset, kemudian dilakukan reset indeks agar struktur data tetap rapi dan konsisten menggunakan kode program berikut.

```
# Menghapus kolom yang tidak relevan (kontan dan sparse)
data_ddos.drop(columns=columns_to_drop, inplace=True)

# Reset indeks setelah drop
data_ddos.reset_index(drop=True, inplace=True)

# Ukuran data setelah drop (baris, kolom)
print(f"Ukuran dataset DDoS saat ini: {data_ddos.shape[0]} baris
dan {data_ddos.shape[1]} kolom.")
```

```
Ukuran dataset DDoS saat ini: 223082 baris dan 63 kolom.
```

Langkah berikutnya dalam tahap *pre-processing* yaitu analisis distribusi Label untuk mengetahui proporsi antar lalu lintas benign dan DDoS. Berikut output dari kode program yang digunakan dalam langkah analisis distribusi ini.

```
Jumlah berdasarkan Label:
Label
DDoS   128014
BENIGN   95068
Name: count, dtype: int64
Persentase berdasarkan Label:
Label
DDoS   57.38%
BENIGN  42.62%
Name: proportion, dtype: object
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Langkah terakhir dalam tahap *pre-processing* adalah memisahkan label dari fitur, bertujuan agar proses penelitian hanya dilakukan pada fitur numerik, sedangkan label disimpan untuk proses evaluasi setelahnya.

4.4.2 *Normalization* (Normalisasi)

Setelah proses *data cleaning*, langkah selanjutnya adalah normalization (normalisasi) terhadap data numerik agar dalam skala yang sama. Jika skala fitur tidak sama, maka fitur dengan nilai yang lebih besar dapat mendominasi dan menyebabkan hasil klasterisasi menjadi tidak representatif. Penelitian ini menggunakan normalisasi dengan metode *Min-Max Scaling*, yang mengubah nilai setiap fitur ke dalam rentang 0 dan 1. Metode ini tetap mempertahankan distribusi relatif antar nilai, sehingga tidak mengubah bentuk distribusi asli data. Berikut kode program yang digunakan dalam proses normalisasi data.

```
scaler = MinMaxScaler()

# .fit_transform() akan mempelajari min dan max dari setiap
kolom, lalu menerapkannya
scaled_features_array = scaler.fit_transform(data_features)

# Ubah kembali array yang sudah di-scaling menjadi DataFrame
Pandas
data_scaled = pd.DataFrame(scaled_features_array,
columns=data_features.columns)
```

4.4.3 *Feature Selection* (Seleksi Fitur)

Feature selection merupakan langkah penting dalam pre-processing untuk memastikan bahwa hanya fitur-fitur yang paling relevan dan berkontribusi terhadap deteksi serangan DDoS yang digunakan dalam analisis. Dalam penelitian ini, pemilihan fitur dilakukan secara manual berdasarkan analisis karakteristik serangan DDoS yang dikenal sebagai serangan volumetrik, yaitu serangan yang mengeksploitasi kapasitas bandwidth dan sumber daya server dengan cara mengirimkan lalu lintas dalam jumlah besar secara terus-menerus. Karakteristik

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

serangan DDoS meliputi jumlah paket yang sangat tinggi dalam waktu singkat, rata-rata ukuran paket yang besar, laju transfer data yang ekstrem, dan variasi panjang paket. Berdasarkan analisis tersebut, dipilih empat fitur utama sebagai representasi karakteristik serangan DDoS dari aspek volume yang ditunjukkan pada Tabel 4.2.

Tabel 4. 2 Daftar Fitur yang Dipilih

No	Nama Fitur	Keterangan
1	*Total Fwd Packets*	Jumlah total paket yang dikirim dari sumber ke tujuan dalam satu sesi aliran
2	*Average Packet Size*	Ukuran rata-rata dari semua paket yang dikirim
3	*Flow Bytes/s*	Jumlah *byte* yang ditransmisikan per detik
4	*Min Packet Length*	Variasi ukuran paket

Fitur-fitur tersebut dipilih karena dianggap mampu membedakan antara lalu lintas normal dan yang mengindikasikan serangan DDoS. Selain itu, jumlah fitur yang terbatas juga membantu dalam meningkatkan efisiensi proses klasterisasi dan deteksi serta mengurangi kompleksitas perhitungan. Berikut adalah kode program yang digunakan untuk mengambil subset fitur dari data yang telah dinormalisasi.

```
# Daftar fitur terpilih

selected_features = [
  'Total Fwd Packets',
  'Average Packet Size',
  'Flow Bytes/s',
  'Min Packet Length',
]

# Mengambil subset fitur dari data hasil normalisasi
data_dbscan = data_scaled[selected_features]
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Hasil dari proses seleksi fitur ini adalah sebuah data yang hanya terdiri dari empat fitur terpilih, yang selanjutnya akan digunakan dalam proses klasterisasi menggunakan algoritma DBSCAN.

Dalam penelitian ini, dilakukan *data splitting* berdasarkan pada label asli sebagai acuan untuk proses validasi threshold. Pembagian ini bertujuan untuk memisahkan antara data yang dianggap normal dan data yang mengandung serangan DDoS, guna membentuk threshold yang representatif serta menguji keefektifannya. Langkah pertama, data hasil normalisasi dan seleksi fitur digabung kembali dengan label asli agar dapat dipisahkan secara eksplisit antara lalu lintas benign dan DDoS. Pemisahan data dan benign berdasarkan nilai pada kolom Label. Lalu lintas benign digunakan untuk membentuk threshold, sementara DDoS digunakan untuk pengujian. Selanjutnya, untuk membentuk threshold yang merepresentasikan kondisi normal secara akurat, data benign dibagi menjadi dua bagian: 70% data benign digunakan untuk membentuk threshold (data training) dan 30% data benign digabungkan dengan seluruh data DDoS sebagai data testing. Berikut kode program yang digunakan dalam proses *data splitting*.

```
# Membagi data benign menjadi data training dan data testing (70%
training dan 30% testing)
benign_train, benign_test = train_test_split(
  benign_data,
  test_size=0.3,
  random_state=42,
  shuffle=True
)

test_data = pd.concat([benign_test, ddos_data], ignore_index=True)
```

Data benign yang digunakan untuk threshold akan dihilangkan labelnya, karena DBSCAN adalah metode *unsupervised learning*, sedangkan data testing dipisahkan dari label untuk proses deteksi, namun label aslinya disimpan untuk keperluan evaluasi. Berikut kode program yang digunakan dalam proses ini.

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

```
# Untuk DBSCAN & threshold
data_train = benign_train.drop(columns=['Label'])

# Untuk evaluasi threshold
data_test = test_data.drop(columns=['Label'])

# Label asli untuk evaluasi
label_test = test_data['Label']
```

4.5 DBSCAN

DBSCAN merupakan algoritma *unsupervised learning* berbasis kepadatan yang mampu membentuk klaster tanpa menentukan jumlah klaster di awal, serta secara otomatis mengenali titik-titik yang tidak sesuai dengan pola kepadatan mayoritas sebagai *noise*. Dalam penelitian ini, parameter *MinPts* ditentukan berdasarkan jumlah minimal. Untuk menentukan nilai *epsilon*, digunakan pendekatan visual grafik K-Distance Plot, yaitu memploting jarak ke tetangga ke-*MinPts* dari setiap data serta titik perubahan tajam (*elbow point*) pada grafik dianggap sebagai estimasi. Berdasarkan gambar 4.1, nilai *epsilon* yang diperoleh berada di angka 0.02, berikut kode program yang digunakan untuk menentukan nilai *epsilon*.

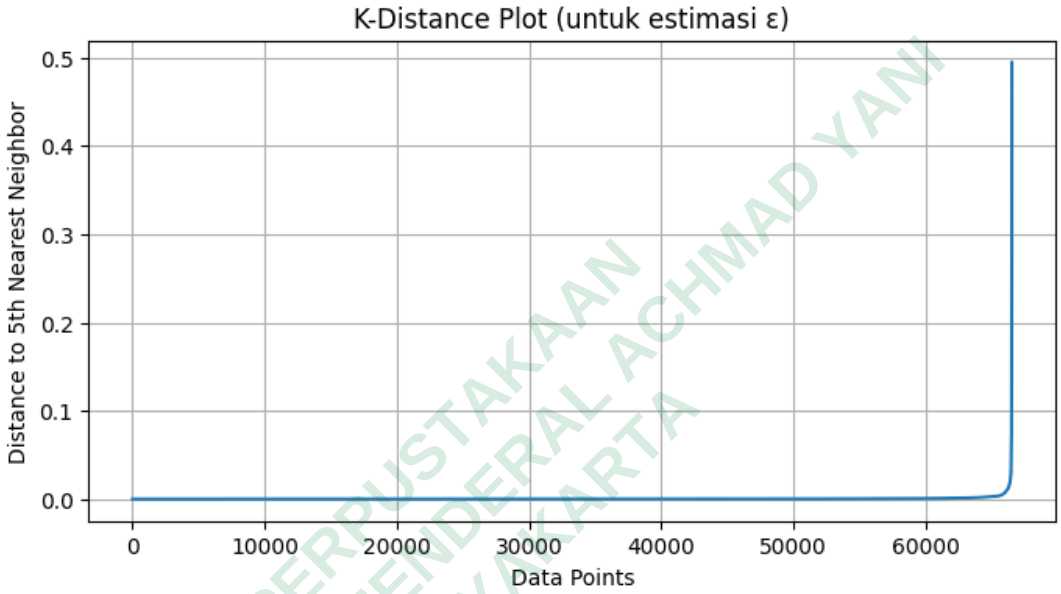
```
# Estimasi MinPts =
min_pts = 5

# Estimasi epsilon
nn = NearestNeighbors(n_neighbors=min_pts)
nn.fit(data_train)
distances, indices = nn.kneighbors(data_train)

# Urutkan jarak untuk plot
distances = np.sort(distances[:, min_pts-1])
plt.figure(figsize=(8, 4))
plt.plot(distances)
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

```
plt.title("K-Distance Plot (untuk estimasi ε)")
plt.xlabel("Data Points")
plt.ylabel(f"Distance to {min_pts}th Nearest Neighbor")
plt.grid(True)
plt.show()
```



PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

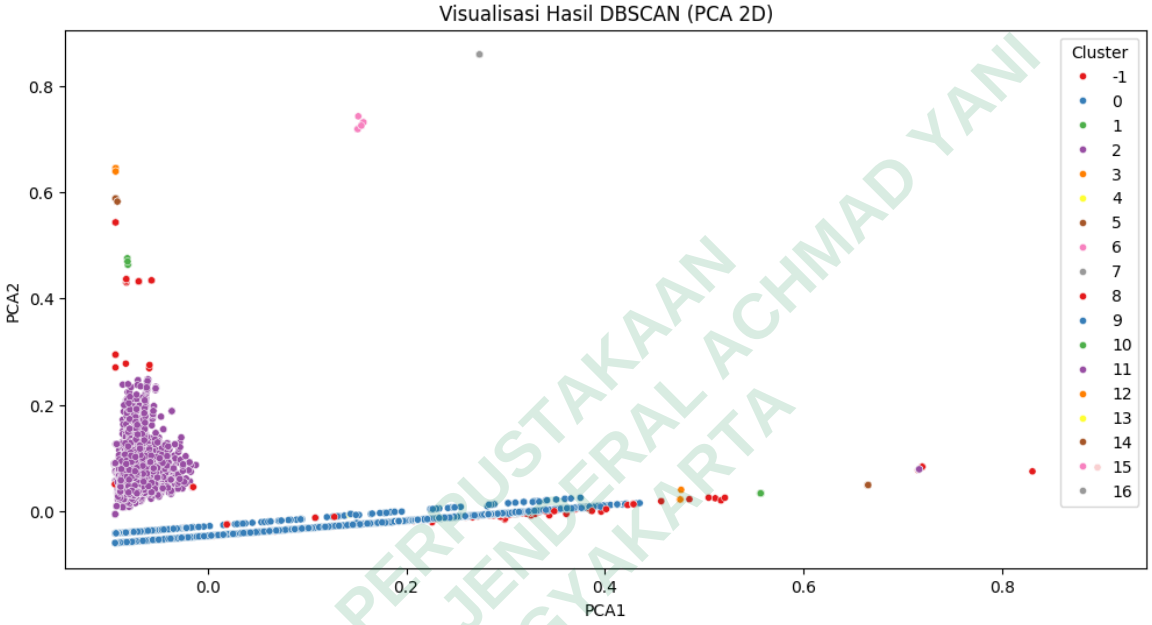
Gambar 4. 1 K-Distance Plot

Setelah parameter ditentukan, dilakukan proses klasterisasi menggunakan DBSCAN terhadap data benign, yaitu data lalu lintas normal yang telah dinormalisasi dan dipilih fiturnya, menggunakan kode program sebagai berikut.

```
# Proses DBSCAN
dbscan = DBSCAN(eps=0.02, min_samples=min_pts)
dbscan.fit(data_train)

# Menyimpan hasil label cluster
cluster_labels = dbscan.labels_
```

Untuk distribusi hasil klaster, menggunakan teknik *Principal Component Analysis* (PCA) untuk mereduksi dimensi data menjadi dua komponen utama dan divisualisasikan dalam bentuk *scatter plot*. Dimensi data direduksi menjadi komponen PCA1 dan PCA2 untuk menggambarkan distribusi data berdasarkan hasil klasterisasi DBSCAN. Visualisasi hasil DBSCAN dalam bentuk PCA 2 dimensi ini dapat dilihat pada Gambar 4.2.



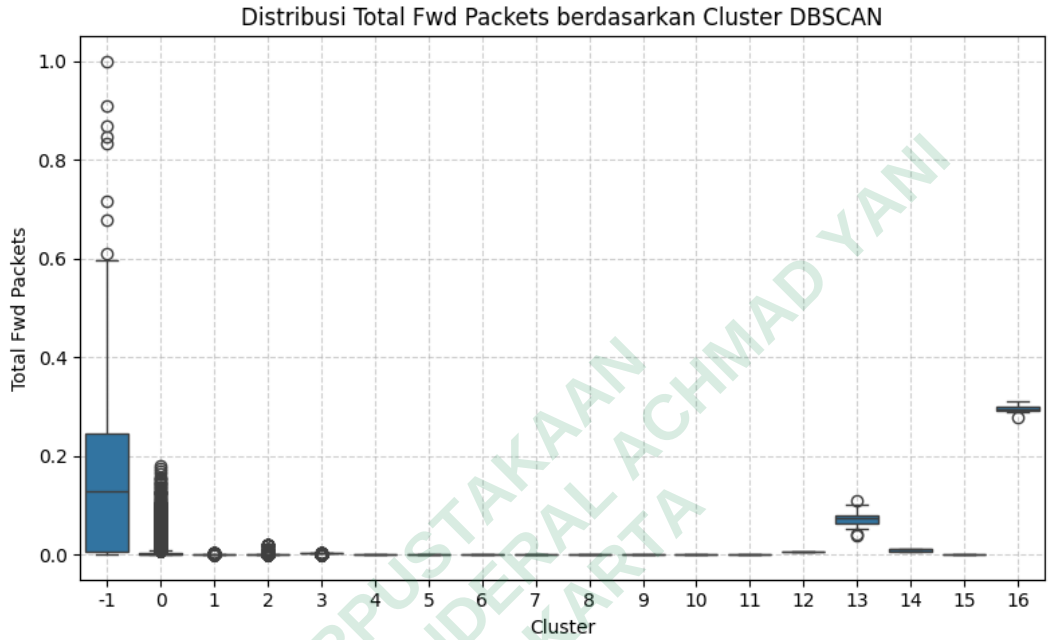
Gambar 4. 2 Visualisasi Hasil DBSCAN (PCA 2D)

Berdasarkan gambar tersebut, dapat dilihat bahwa titik-titik warna yang berjumlah lebih dari 10 klaster berbeda berhasil dibentuk oleh DBSCAN. Titik dengan label -1 merupakan data yang dianggap *noise* oleh DBSCAN, yaitu data yang tidak tergabung dalam klaster manapun karena tidak memenuhi kepadatan minimum (*MinPts*). Mayoritas data terkonsentrasi pada dua klaster utama, menunjukkan bahwa DBSCAN mampu mengidentifikasi pola lalu lintas benign yang dominan. Sebagian kecil data tersebar lebih jauh sepanjang komponen PCA1 dan PCA2, yang kemungkinan mencerminkan variasi alami dalam lalu lintas benign atau adanya *outlier* dalam data benign oleh DBSCAN.

Selain visualisasi PCA 2 dimensi, dilakukan juga visualisasi distribusi masing-masing fitur dalam bentuk boxplot yang digunakan dalam proses threshold profiling terhadap hasil klaster DBSCAN. Visualisasi ini bertujuan untuk

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

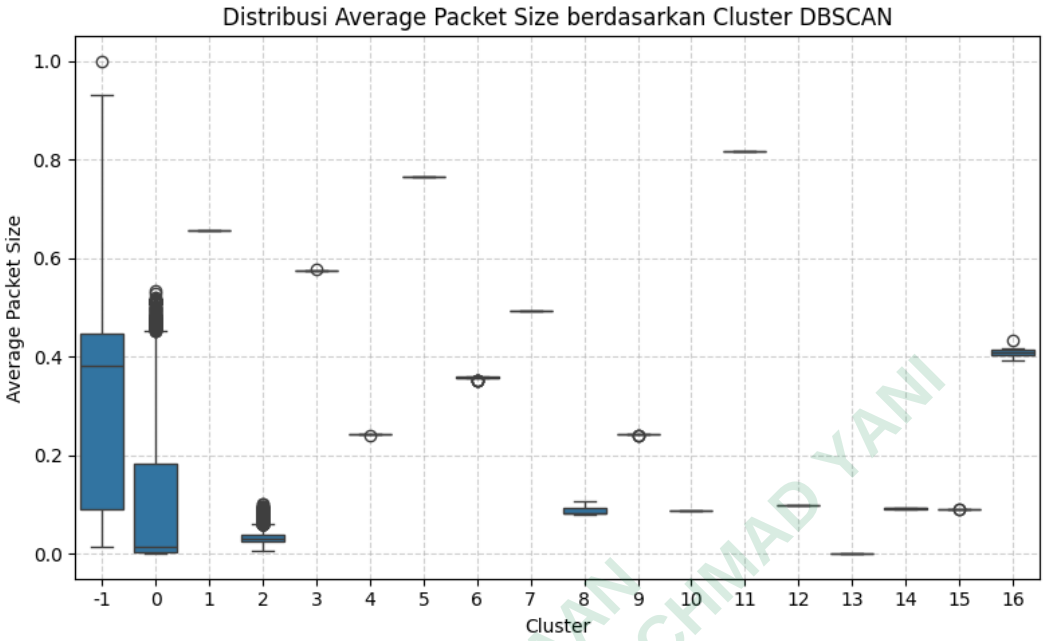
menunjukkan karakteristik masing-masing klaster, menggambarkan persebaran nilai fitur berdasarkan masing-masing label klaster serta mengidentifikasi fitur-fitur yang mampu memisahkan klaster secara visual. Berikut visualisasi boxplot beserta penjelasan untuk masing-masing fitur yang telah dipilih.



Gambar 4. 3 Boxplot Distribusi Total Fwd Packets

Gambar 4.3 Menunjukkan visualisasi boxplot dari fitur *Total Fwd Packets* berdasarkan hasil klasterisasi DBSCAN. Pada gambar tersebut terlihat bahwa klaster -1 memiliki distribusi nilai yang sangat bervariasi dan cenderung tinggi, sehingga mengindikasikan bahwa klaster tersebut berisi data *outlier*. Sementara pada klaster 0 hingga 13 cenderung memiliki nilai yang mendekati nol dengan distribusi yang stabil, dapat diasumsikan lalu lintas jaringan yang normal. Sedangkan pada klaster 16 menampilkan median dan variasi yang lebih tinggi dibandingkan klaster lainnya, yang dapat menjadi indikasi pola aktivitas mencurigakan atau lalu lintas yang dicurigai sebagai serangan DDoS dengan intensitas tertentu.

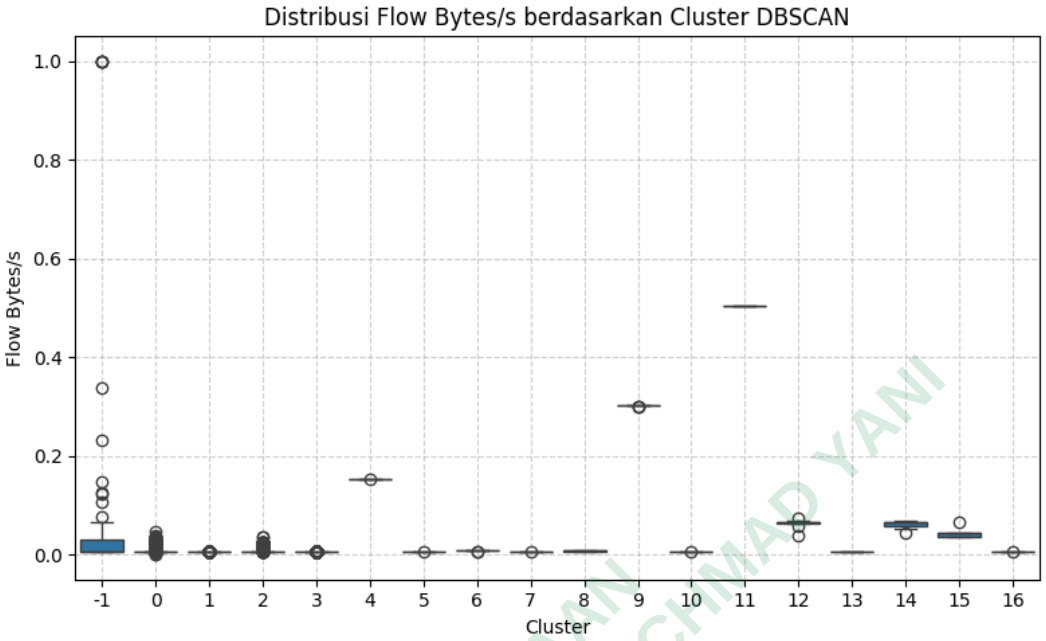
PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA



Gambar 4. 4 Boxplot Distribusi Average Packet Size

Gambar 4.4 menunjukkan visualisasi boxplot dari fitur *Average Packet Size* berdasarkan hasil klasterisasi DBSCAN. Dari gambar tersebut terlihat bahwa klaster -1 dan klaster 0 memiliki rentang distribusi yang cukup lebar, sehingga mengindikasikan adanya keberagaman nilai yang ekstrem. Selain itu, terdapat klaster 4, 6, dan 11 yang memiliki nilai *Average Packet Size* yang relatif tinggi namun dengan rentang distribusi yang sempit. Pola ini dapat mengindikasikan bahwa klaster tersebut merepresentasikan jenis lalu lintas tertentu yang konsisten.

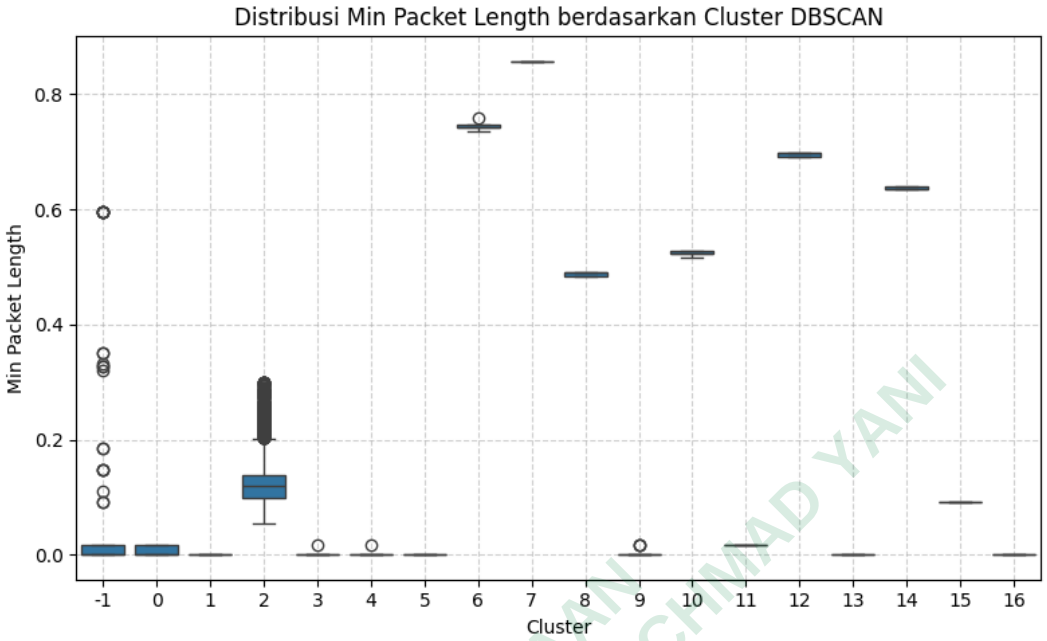
PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA



Gambar 4. 5 Boxplot Distribusi Flow Bytes/s

Gambar 4.5 menampilkan visualisasi boxlot dari fitur *Flow Bytes/s* berdasarkan hasil klasterisasi DBSCAN. Dari gambar, terlihat bahwa klaster -1 memiliki distribusi data yang sangat tinggi dan tersebar luas, mencerminkan ketidakteraturan dalam volume lalu lintas jaringan. Klaster-klaster lainnya menunjukkan sebaran nilai *Flow Bytes/s* yang lebih sempit dan stabil, menandakan bahwa lalu lintas pada klaster tersebut memiliki pola yang lebih teratur dan kemungkinan merepresentasikan lalu lintas normal.

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA



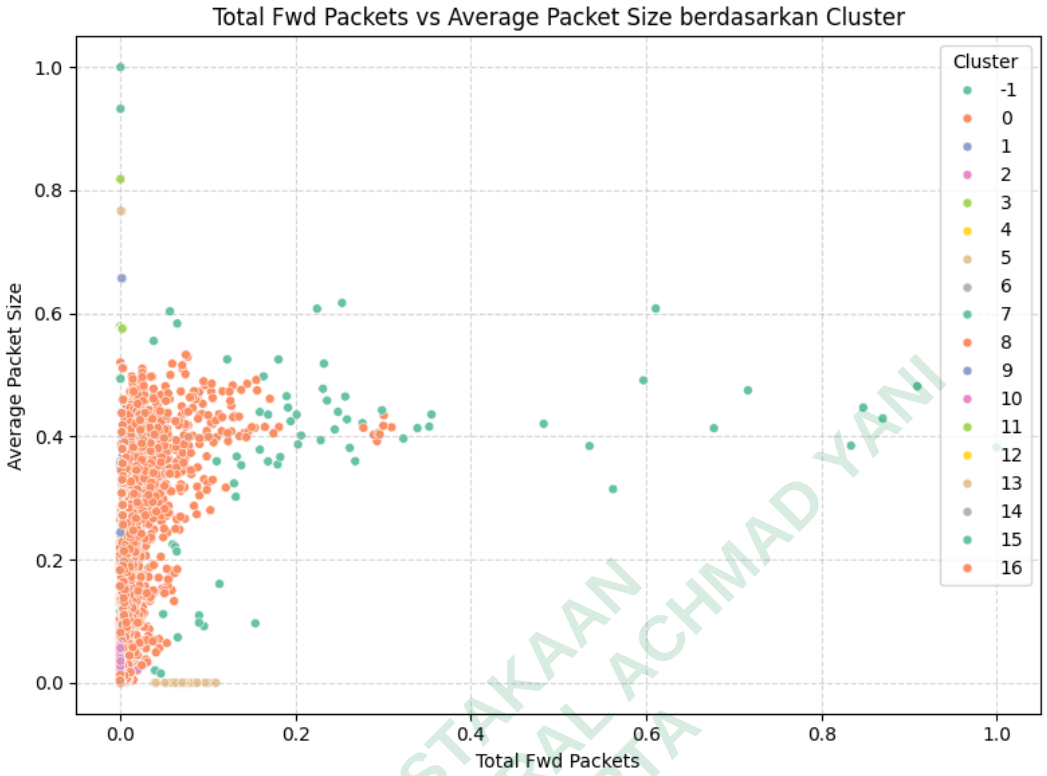
Gambar 4. 6 Boxplot Distribusi Min Packet Length

Gambar 4.6 memperlihatkan distribusi nilai fitur Min Packet Length berdasarka hasil klasterisasi DBSCAN. Terlihat bahwa setiap klaster memiliki rentang nilai yang berbeda satu sama lain, hal ini menunjukkan bahwa fitur ini sensitif terhadap variasi jenis lalu lintas dalam data jaringan.

Hasil visualisasi boxplot pada masing-masing fitur diatas, mendukung asumsi bahwa DBSCAN berhasil memisahkan lalu lintas berdasarkan kepadatan dan pola fitur, dan menunjukkan bahwa fitur-fitur terpilih seperti *Average Packet Size* dan *Min Packet Length* efektif dalam membedakan karakteristik lalu lintas benign dan serangan DDoS.

Selain menggunakan boxplot untuk memvisualisasikan distribusi nilai antar fitur, dilakukan juga visualisasi hasil klasterisasi DBSCAN dalam bentuk *scatter plot* berdasarkan kombinasi fitur. Hal ini bertujuan untuk menampilkan pola pemisahan antar klaster secara lebih eksplisit. Kombinasi fitur yang pertama yaitu *Total Fwd Packets* dan *Average Packet Size* yang ditunjukkan pada Gambar 4.7 berikut.

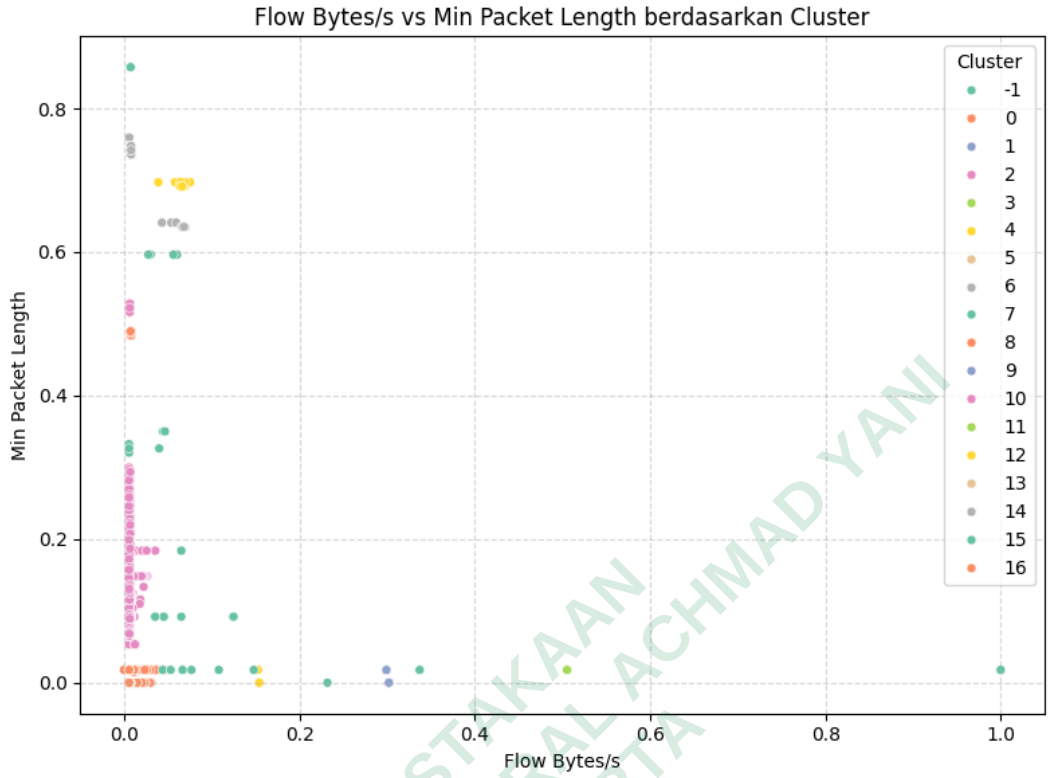
PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA



Gambar 4. 7 Scatter Plot Fitur Total Fwd Packets dan Average Packet Size

Pada gambar tersebut, klaster 0 mendominasi area padat yang menunjukkan bahwa DBSCAN berhasil mengelompokkan data benign secara konsisten ke dalam satu klaster besar. Sementara itu, klaster -1 tersebar luas dan menjangkau hingga kanan atas menunjukkan bahwa terdapat sejumlah data dengan jumlah paket dan ukuran paket yang tinggi. Persebaran ini menunjukkan bahwa DBSCAN berhasil mendeteksi pola lalu lintas ekstrem sebagai anomali dan mengelompokkannya sebagai *outlier*.

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA



Gambar 4. 8 Scatter Plot Fitur Flow Bytes/s dan Min Packet Length

Gambar 4.8 menampilkan *scatter plot* kombinasi fitur *Flow Bytes/s* dan *Min Packet Length*. Pada gambar, terlihat bahwa sebagian besar titik data berkumpul di area kiri bawah, yang merepresentasikan aliran data dengan laju rendah dan ukuran paket minimum yang kecil. Klaster 0 dan 2 mendominasi dan terkonsentrasi cukup padat, menandakan bahwa DBSCAN berhasil mengelompokkan sebagian besar lalu lintas ke dalam klaster tertentu secara stabil.

Visualisasi *scatter plot* antara *Total Fwd Packets* dan *Average Packet Size* menunjukkan bahwa sebagian besar data terkonsentrasi dalam satu area padat dan didominasi oleh klaster 0, yang diasumsikan sebagai lalu lintas normal. *Scatter plot* kedua, antara *Flow Bytes/s* dan *Min Packet Length*, mengungkapkan pola serupa. Klaster *noise* tersebar luas dengan laju data tinggi, sedangkan beberapa klaster kecil muncul di posisi dengan panjang paket minimum yang besar. Hal ini menegaskan bahwa fitur-fitur ini berhasil memisahkan pola lalu lintas berdasarkan kepadatan dan karakteristik spesifiknya. Hasil ini mendukung keberhasilan DBSCAN dalam mengidentifikasi baik lalu lintas normal maupun anomali.

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Untuk mengukur kualitas hasil klasterisasi oleh DBSCAN dalam mengelompokkan data benign, menggunakan evaluasi *Silhouette Index* (SI) atau *Silhouette Score*. Metrik ini mengevaluasi kualitas melalui dua aspek, yaitu kedekatan intra-klaster dan jarak antar-klaster. Dalam implementasinya, perhitungan *Silhouette Score* berdasarkan jika hasil DBSCAN menghasilkan lebih dari satu klaster valid dan terdapat *noise* (label -1). Berikut adalah kode program yang digunakan dalam menghitung *Silhouette Score*.

```
# Menghitung silhouette score, hanya jika ada lebih dari 1
cluster (bukan hanya noise)
if len(set(cluster_labels)) > 1 and -1 in cluster_labels:
      score = silhouette_score(data_train, cluster_labels)
      print(f"Silhouette Score: {score:.4f}")
```

```
Silhouette Score: 0.0387
```

Hasil dari perhitungan *Silhouette Score* pada penelitian ini adalah 0,0387. Nilai ini menunjukkan bahwa klaster-klaster yang terbentuk memiliki struktur yang cukup baik dengan pemisahan yang jelas.

Distribusi label pada data testing menunjukkan adanya *imbalanced data*, dimana data serangan DDoS mendominasi dengan jumlah 128.014 entri (81.77%), sedangkan data benign sebanyak 28.251 entri (18.22%). Berikut kode program yang digunakan untuk mengetahui distribusi label pada data testing.

```
# Distribusi Label di Data Testing
print("Distribusi label di data testing:")
print(label_test.value_counts())
print("\nPersentase:")
print(label_test.value_counts(normalize=True) * 100)
```

```
Distribusi label di data testing:
Label
DDoS   128014
BENIGN   28521
Name: count, dtype: int64

Persentase:
Label
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

```
DDoS    81.779794
BENIGN  18.220206
Name: proportion, dtype: float64
```

Imbalanced data ini dapat menyebabkan bias dalam proses evaluasi performa sistem deteksi, sehingga perlu dilakukan *data balancing* (penyeimbangan data). Metode *data balancing* yang digunakan adalah *undersampling* terhadap kelas mayoritas (DDoS), dengan mengambil sampel sebanyak jumlah data benign. Dengan langkah ini, diperoleh data testing seimbang yang terdiri dari dua kelas dengan jumlah sama. Berikut kode program yang digunakan dalam proses *data balancing*.

```
# Undersampling jumlah DDoS agar jumlahnya seimbang dengan Benign
ddos_sample = ddos_test_data.sample(n=len(benign_test_data),
random_state=42)
```

4.6 Threshold Profiling

Setelah proses klasterisasi menggunakan algoritma DBSCAN, setiap data benign telah memperoleh label klaster berdasarkan kepadatan distribusinya. Untuk melanjutkan ke tahap threshold profiling, hasil klasterisasi ini digabungkan kembali ke data benign menggunakan indeks yang sesuai. Proses ini bertujuan untuk mengidentifikasi klaster mana saja yang dapat dianggap sebagai representasi dari lalu lintas normal. Dalam DBSCAN, klaster dengan label selain -1 dianggap valid, sedangkan label -1 menandakan *noise*. Berikut kode program dan distribusi jumlah data per klaster hasil DBSCAN.

```
# Gabungkan data training dengan hasil cluster
data_train_clustered = data_train.copy()
data_train_clustered['Cluster'] = cluster_labels

# Tampilkan beberapa baris pertama untuk verifikasi
print("Data training dengan label cluster:")
display(data_train_clustered.head())

print("\nDistribusi jumlah data per cluster:")
print(data_train_clustered['Cluster'].value_counts())
```

```
Distribusi jumlah data per cluster:
Cluster
 0   42078
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

```
 2    21664
 3    1501
 1     941
-1     97
 8     39
 5     38
 13     32
 9     26
 6     25
 10     25
 11     18
 12     18
 7     11
 15     11
 14     9
 16     8
 4      6
Name: count, dtype: int64
```

Threshold dibentuk dari lalu lintas jaringan yang benar-benar representatif sebagai data normal, hanya data benign yang termasuk dalam klaster valid yang akan digunakan. Data dengan label -1 dianggap sebagai *noise* atau potensi anomali, sehingga tidak digunakan dalam proses perhitungan threshold. Setelah proses penyaringan, kolom cluster dihapus karena tidak diperlukan dalam analisis statistik. Dataset hasil penyaringan tersebut yang menjadi dasar perhitungan threshold berdasarkan nilai rata-rata dan standar deviasi dari masing-masing fitur. Berikut kode program beserta output jumlah data benign yang valid dan akan digunakan dalam pembentukan threshold.

```
# Memfilter data hanya cluster valid (tidak termasuk noise/-1)
valid_cluster_data =
data_train_clustered[data_train_clustered['Cluster'] != -1]

# Drop kolom 'Cluster', karena hanya fitur yang dibutuhkan
valid_features = valid_cluster_data.drop(columns=['Cluster'])

# Menampilkan jumlah data dari cluster valid
print(f"Jumlah data benign (valid cluster):
{valid_features.shape[0]}")
```

```
Jumlah data benign (valid cluster): 66450
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Threshold ditentukan berdasarkan rata-rata dan standar deviasi dari setiap fitur dalam data benign yang dianggap mewakili lalu lintas normal. Untuk menentukan nilai koefisien (*k*) terbaik untuk masing-masing fitur menggunakan metode *grid search*. Dalam penelitian ini, nilai *k* yang diuji berada dalam rentang 0.1 hingga 3.0 dengan langkah 0.5. Tujuannya adalah mengeksplorasi kombinasi threshold untuk mendapatkan hasil deteksi dengan evaluasi terbaik. Berikut kode program yang digunakan untuk inisialisasi *grid search.*

```
# Daftar nilai k yang akan diuji (0.1 sampai 3.0 dengan langkah
0.5)
k_values_range = np.arange(0.1, 3.0, 0.5)

# Fitur yang digunakan
selected_features = ['Total Fwd Packets', 'Average Packet Size',
'Min Packet Length', 'Flow Bytes/s']

# Data benign dari DBSCAN
valid_features = valid_cluster_data[selected_features]

# Grid search untuk kombinasi k
results = []

for k_total, k_avgsize, k_minlen, k_flowbytes, k_packetlengthmean
in product(k_values_range, repeat=5):
  k_dict = {
    'Total Fwd Packets': k_total,
    'Average Packet Size': k_avgsize,
    'Min Packet Length': k_minlen,
    'Flow Bytes/s': k_flowbytes
  }
```

UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA PERPUSTAKAAN

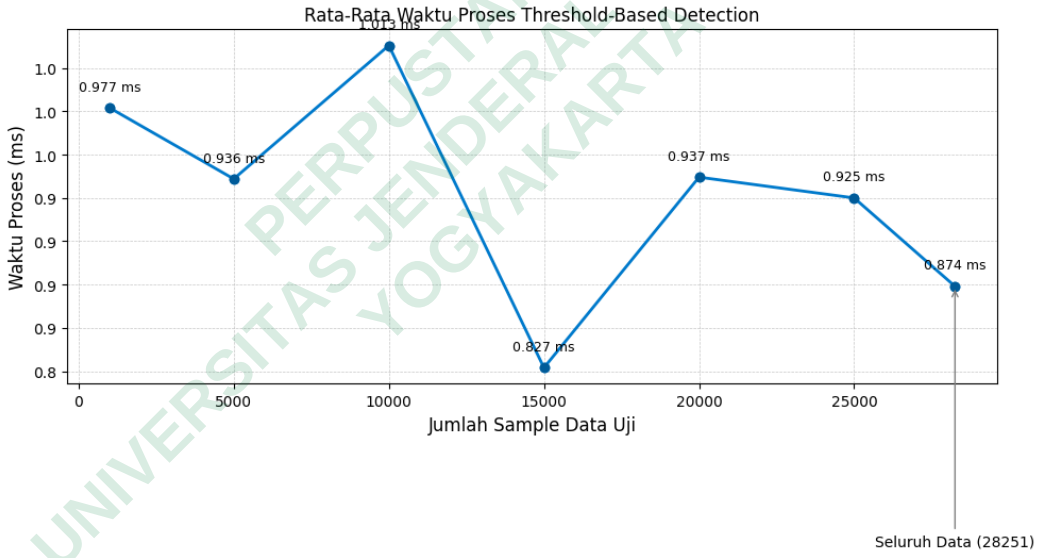
4.7 Threshold-Based Detection

Proses deteksi anomali dalam penelitian ini menggunakan pendekatan Threshold-Based Detection, yaitu mengelompokkan suatu aliran jaringan sebagai anomali apabila nilai satu atau lebih fiturnya melebihi threshold tertentu. Perhitungan threshold dilakukan dengan pendekatan dinamis struktur dictionary k_dict, yang berisi nilai *k* untuk masing-masing fitur. Berikut kode program yang digunakan dalam menghitung threshold pada setiap fitur.

```
# Threshold Based-Detection
exceeds = balanced_test_data[selected_features] > thresholds
predicted_anomaly = exceeds.any(axis=1).astype(int)
```

4.8 Evaluasi

Untuk mengetahui seberapa cepat metode Threshold-Based Detection dalam memproses berbagai jumlah sampel data testing secara bertahap, mulai dari 1.000 data sampel hingga seluruh data testing sejumlah 28.251 baris. Berdasarkan Gambar 4.9, bahwa rata-rata waktu proses untuk mendeteksi serangan DDoS berada dibawah 1.1 milidetik (ms) untuk setiap ukuran jumlah data sampel. Waktu proses tercepat terjadi pada 15.000 data uji, yaitu sekitar 0.827 ms, sedangkan waktu terlama terjadi pada 10.000 data sampel, sekitar 1.013 ms. Dalam penelitian ini, waktu proses cenderung stabil dan efisien, bahkan saat seluruh data diuji sekaligus.



Gambar 4. 9 Rata-rata Waktu Proses Threshold-Based Detection

Setelah dilakukan *grid search* terhadap kombinasi nilai threshold untuk empat fitur yang telah dipilih, sistem mengevaluasi performa deteksi anomali untuk setiap kombinasi menggunakan data uji yang sudah seimbang (*data balancing*). Berikut kode yang digunakan dalam memilih 10 kombinasi terbaik berdasarkan *F-1 Score* sebagai metrik utama.

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

```
# Menampilkan 10 kombinasi terbaik berdasarkan F1-Score
top_results = results_df.sort_values(by='F1-Score',
ascending=False).head(10)
print("Top 10 kombinasi threshold:")
print(top_results.round(4))
```

Dari kombinasi tersebut, kemudian dipilih kombinasi paling baik (Top-1) yang memiliki performa tertinggi secara keseluruhan, terutama dalam mendeteksi serangan DDoS dengan tetap menjaga jumlah *false alarm* seminimal mungkin. Berikut output dari kode program yang digunakan dalam tahap ini.

```
Evaluasi Akhir (Top-1 threshold terbaik):
Threshold k:
 Total Fwd Packets   : 2.6
 Average Packet Size  : 1.1
 Min Packet Length   : 2.6
 Flow Bytes/s     : 2.6

Evaluasi performa:
 Accuracy : 0.7253
 Precision : 0.7751
 Recall  : 0.6347
 F1-Score : 0.6979
 FPR    : 0.1841
```

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Berdasarkan kombinasi threshold terbaik tersebut, hasil evaluasi sistem dijelaskan melalui metrik evaluasi berikut.

4.8.1 *Accuracy*

Accuracy mencerminkan bahwa sistem mampu mengklasifikasikan data benign dan DDoS secara benar dalam proporsi yang tinggi dari seluruh data testing. Dalam penelitian ini, sistem mencerminkan performa yang baik dan berhasil mengklasifikasikan 72.53% dari seluruh data testing dengan benar, baik benign maupun DDoS.

4.8.2 *Precision*

Precision mencerminkan bahwa sebagian besar deteksi terhadap DDoS memang benar-benar serangan. Nilai *precision* tinggi mengindikasikan *false alarm* yang rendah, sehingga sistem tidak keliru dalam mengganggap lalu lintas normal

sebagai serangan. Dari seluruh DDoS, menghaslkan sekitar 77.51% deteksi yang benar. Dalam hal ini, menunjukkan bahwa sistem presisi dan tidak banyak menghasilkan *false alarm*.

4.8.3 *Recall*

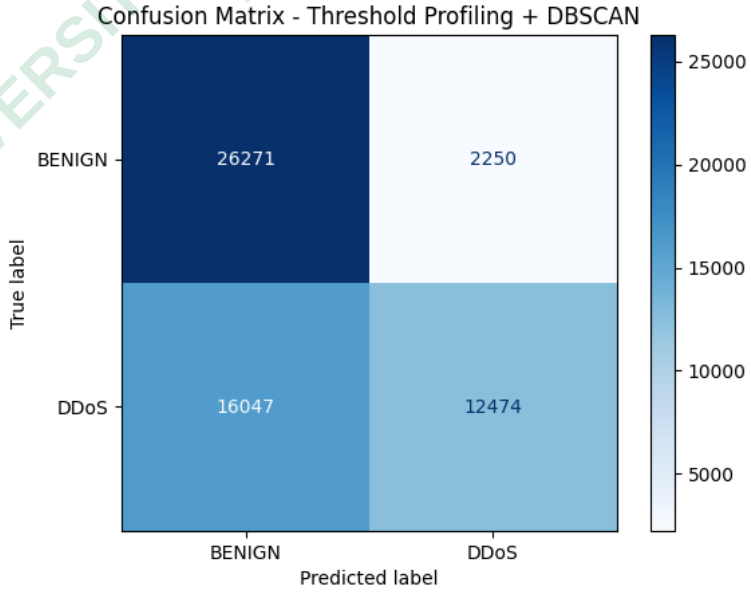
Recall mengukur seberapa banyak serangan DDoS yang berhasil dideteksi oleh sistem dan menandakan bahwa sistem sangat sensitif dalam mendeteksi serangan. Dalam penelitian ini, sistem berhasil mendeteksi sekitar 63.47% dari seluruh serangan DDoS yang ada dalam data testing.

4.8.4 *F1-Score*

F1-Score menunjukkan bahwa sistem memiliki kemampuan deteksi yang stabil dan andal. Dalam penelitian ini, sistem berhasil mendeteksi 69.79% dari serangan DDoS.

4.8.5 *False Positive Rate* (FPR)

FPR mencerminkan proporsi lalu lintas benign yang keliru diklasifikasikan sebagai DDoS, sekitar 18.41% dari data benign keliru diklasifikasikan sebagai serangan DDoS. Untuk melengkapi pemahaman terhadap performa sistem, berikut visualisasi *confusion matrix* seperti terlihat pada Gambar 4.10.



Gambar 4. 10 Confusion Matrix

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Confusion matrix memperlihatkan rincian jumlah data yang berhasil diklasifikasikan dengan benar dan salah oleh sistem. Dari total data benign, sebanyak 26.271 data berhasil diklasifikasikan dengan benar (*True Negative*), sementara 2.250 data lainnya salah diklasifikasikan sebagai DDoS (*False Positive*). Sedangkan dari total data DDoS, sistem berhasil mendeteksi 12.474 data (*True Positive*) sementara 16.047 lainnya sebagai *false negative*. Secara keseluruhan, visualisasi *confusion matrix* mendukung hasil evaluasi dan menunjukkan bahwa sistem memiliki performa yang cukup baik dalam membedakan lalu lintas normal dan serangan DDoS.

Seluruh proses threshold profiling dalam penelitian ini dirancang untuk bersifat iteratif. Jika nilai *recall* terlalu rendah atau FPR terlalu tinggi, sistem memungkinkan penyesuaian parameter fitur, nilai *epsilon*, *MinPts*, maupun koefisien *k* untuk threshold setiap fitur. Dengan demikian, performa dapat ditingkatkan secara bertahap dan melalui proses *grid search* hingga diperoleh konfigurasi yang paling optimal.

PERPUSTAKAAN UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA