

**Laporan Tugas Besar 2 IF3170 Inteligensi Artifisial  
Semester Ganjil Tahun Ajaran 2024/2025**

# **Implementasi Algoritma Pembelajaran Mesin (KNN, GNB, dan ID3)**

**Oleh:**

**13221011 JAZILA FAZA ALIYYA NURFAUZI**

**13221055 AHMAD HAFIDZ ALIIM**

**13221065 CAITLEEN DEVINA**

**18221130 RAYHAN MAHESWARA PRAMANDA**

**18321008 JASMINE CALLISTA AURELLIE IRFAN**



**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
DESEMBER 2024**

## Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>1. Deskripsi Tugas</b>	<b>3</b>
<b>2. Algoritma K-Nearest Neighbor (KNN)</b>	<b>3</b>
a. Deskripsi Singkat	3
b. Alur Kerja Algoritma	4
c. Penjelasan Implementasi	4
<b>3. Algoritma Gaussian Naive Bayes (GNB)</b>	<b>4</b>
a. Deskripsi Singkat	4
b. Alur Kerja Algoritma	5
c. Penjelasan Implementasi	5
<b>4. Algoritma ID3 (DTL)</b>	<b>7</b>
a. Deskripsi Singkat	7
b. Alur Kerja Algoritma	8
c. Penjelasan Implementasi	8
<b>5. Penjelasan Exploratory Data Analysis (EDA)</b>	<b>9</b>
<b>6. Penjelasan Data Cleaning &amp; Preprocessing</b>	<b>18</b>
a. Data Cleaning	18
1. Handling Missing Data	18
2. Dealing with Outliers	21
3. Remove Duplicates	25
4. Feature Engineering	25
b. Data Preprocessing	26
1. Feature Scaling	26
2. Encoding Categorical Variables	28
3. Handling Imbalanced Classes	30
4. Dimensionality Reduction	32
<b>7. Penjelasan Data Pipeline</b>	<b>33</b>
<b>8. Perbandingan Algoritma Implementasi Mandiri dengan Pustaka SciKit-Learn</b>	<b>36</b>
a. K-Nearest Neighbor	36
b. Gaussian Naive Bayes	36
c. ID3	37
<b>9. Kontribusi Anggota dalam Kelompok</b>	<b>37</b>
<b>10. Referensi</b>	<b>39</b>

## 1. Deskripsi Tugas

Pada tugas besar 2 IF3170 ini, terdapat sebuah *data set* berisi riwayat serangan siber beserta data-data terkait yang relevan dari UNSW bernama UNSW-NB15. Diperlukan sejumlah algoritma *supervised learning* yang diimplementasi dari awal, yakni K-Nearest Neighbour (KNN), Gaussian Naive Bayes (GNB), dan ID3. Model-model tadi akan digunakan untuk memprediksi *data set* UNSW-NB15 sehingga dilakukan pula langkah-langkah EDA, *data cleaning*, dan *data preprocessing* agar data siap digunakan sebagai data latih maupun uji.

## 2. Algoritma K-Nearest Neighbor (KNN)

### a. Deskripsi Singkat

K-Nearest Neighbor (KNN) merupakan salah satu algoritma supervised machine learning untuk penyelesaian masalah klasifikasi dan regresi. Pada implementasinya, KNN dapat diaplikasikan pada masalah pengenalan pola, penambahan data, maupun deteksi intrusi. Algoritma KNN membuat plot koordinat data berdasarkan atribut yang dimilikinya, lalu mengklasifikasikannya ke dalam kelompok-kelompok yang memiliki kedekatan atribut. Saat diberikan koordinat data baru, data tersebut dapat diklasifikasikan ke dalam sebuah kelompok dengan memperhatikan kelompok dengan "neighbor" terdekat. Algoritma KNN bekerja dengan mencari K neighbor terdekat terhadap data target berdasarkan metrik jarak: Euclidean, Manhattan, Minkowski, dll.

Jarak Euclidean merupakan jarak kartesius antara dua titik yang berada pada bidang/hyperplane. Jarak Euclidean juga dapat divisualisasikan sebagai panjang garis lurus yang menghubungkan dua titik. Metrik ini merupakan perhitungan perpindahan total yang terjadi antara dua keadaan suatu benda.

$$EuclideanDistance(x, y) = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$$

Jarak Manhattan digunakan untuk mengetahui jarak total yang ditempuh antara dua keadaan suatu benda. Metrik jarak ini dihitung dengan menjumlahkan selisih mutlak antara koordinat titik-titik pada bidang n-dimensi.

$$ManhattanDistance(x, y) = \sqrt{\sum_{i=1}^n |x_i - y_i|}$$

Jarak Minkowski merupakan bentuk umum dari jarak Euclidean dan Manhattan.

$$MinkowskiDistance(x,y) = \sqrt[p]{\sum_{i=1}^n (x_i - y_i)^p}$$

Dari rumus di atas, dapat disimpulkan bahwa apabila  $p = 2$  maka diperoleh rumus jarak Euclidean dan apabila  $p = 1$  maka diperoleh rumus jarak Manhattan.

**b. Alur Kerja Algoritma**

Algoritma KNN menggunakan prinsip kesamaan dengan memprediksi kategori atau nilai data input berdasarkan kedekatan kategori terhadap K tetangga terdekat pada dataset training. Alur kerja algoritma KNN adalah berikut:

- 1. Menentukan nilai K yang optimal. K merupakan jumlah tetangga terdekat yang perlu dipertimbangkan saat memprediksi.
- 2. Menghitung jarak antara setiap titik data dan titik target.
- 3. Menentukan K tetangga terdekat dari titik target.
- 4. Mengklasifikasi atau mencari rerata untuk regresi.

Dalam mengklasifikasi, kelas dengan kemunculan terbanyak di antara K tetangga terdekat akan menjadi kelas yang diprediksi untuk titik target. Dalam masalah regresi, kelas untuk titik target dihitung dengan mengambil rata-rata nilai dari K tetangga terdekat.

**c. Penjelasan Implementasi**

Pada implementasi algoritma, dibuat sebuah kelas bernama `KNN` yang terdiri dari lima (5) metode, yakni sebagai berikut.

**Tabel 3.1 Penjelasan Kelas Implementasi**

Kelas	Penjelasan
<pre>def __init__(self, k_neighbors=3, metric='euclidean'):     self.n_neighbors = n_neighbors     self.metric = metric     self.x_train = None     self.y_train = None</pre>	Konstruktor kelas yang akan menginisiasi parameter model: <code>k_neighbors</code> yaitu jumlah tetangga terdekat yang ditentukan, metrik jarak yang akan digunakan (euclidean, manhattan, atau minkowski), <code>x_train</code> dan <code>y_train</code> untuk menyimpan data latih yang akan diisi kemudian.

<pre>def fit(self, x, y):     """Store the training data."""     self.x_train = np.array(x)     self.y_train = np.array(y)</pre>	<p>Metode ini menerima x (fitur latih) dan y (label target) sebagai parameter. Lalu, metode mengisi larik x_train dan y_train dengan array NumPy dari x dan y.</p>
<pre>def compute_distances(self, x):     if self.metric == 'euclidean':         distances = np.sqrt(((x[:, np.newaxis] - self.x_train) ** 2).sum(axis=2))     elif self.metric == 'manhattan':         distances = np.abs(x[:, np.newaxis] - self.x_train).sum(axis=2)     elif self.metric == 'minkowski':         distances = (((np.abs(x[:, np.newaxis] - self.x_train)) ** self.p).sum(axis=2)) ** (1 / self.p)     else:         raise ValueError(f"Unsupported metric: {self.metric}")     return distances</pre>	<p>Metode ini menghitung matriks jarak antara setiap data dalam x (data uji) dengan semua data dalam x_train (data latih). Jarak dihitung tergantung metrik yang dipilih: Euclidean, Manhattan, atau Minkowski menggunakan formula yang sesuai. Jika metrik yang diberikan tidak valid, metode akan mengembalikan error. Hasilnya adalah matriks jarak, di mana setiap elemen mewakili jarak antara data uji dan data latih.</p>
<pre>def predict(self, x):     """Predict the class labels for the provided data."""     x = np.array(x)     distances = self.compute_distances(x)      neighbors = np.argsort(distances, axis=1)[: , :self.k_neighbors]     predictions = []      for neighbor_indices in neighbors:         neighbor_labels = self.y_train[neighbor_indices]      predictions.append(np.bincount(neighbor_labels). argmax())      return np.array(predictions)</pre>	<p>Metode ini digunakan untuk memprediksi label kelas dari data input/uji x. Metode memanggil compute_distances untuk menghitung jarak antara data uji dan data latih. Kemudian, indeks dari k_neighbors jarak terkecil diambil untuk setiap data uji. Label kelas dari tetangga terdekat dihitung sesuai frekuensi terbanyak dengan fungsi np.bincount. Hasil berupa array prediksi label kelas untuk setiap data dalam x.</p>
<pre>def save_model(self, filename):     with open(filename, 'wb') as file:         pickle.dump(self, file)</pre>	<p>Metode ini menerima parameter <i>filename</i> kemudian menggunakan fungsi dari pustaka pickle untuk menyimpan model yang telah dilatih ke dalam file dengan format .pkl.</p>

<pre>def load_model(filename):     with open(filename, 'rb') as file:         model = pickle.load(file)     return model</pre>	<p>Metode ini menerima parameter <i>filename</i> kemudian menggunakan fungsi dari pustaka pickle untuk memuat berkas model berformat .pkl dengan nama sesuai parameter <i>filename</i>.</p>
--	---

### 3. Algoritma Gaussian Naive Bayes (GNB)

#### a. Deskripsi Singkat

Naive bayes adalah salah satu algoritma pembelajaran mesin yang tergolong sebagai *supervised learning*. Algoritma ini merupakan jenis khusus dari Bayesian Network yang mengasumsikan independensi antarfitur. Artinya, keberadaan suatu fitur diasumsikan tidak memengaruhi kondisi fitur lainnya. Algoritma ini didasarkan pada teorema Bayes yang berbunyi:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Algoritma Naive Bayes merupakan algoritma probabilistik dan tidak menyimpan informasi data latih, hanya informasi statistik terkaitnya saja. Naive Bayes cukup populer digunakan untuk berbagai tugas klasifikasi, misalnya untuk deteksi spam dan klasifikasi teks. Namun, pada tugas ini, algoritma yang digunakan bukan Naive Bayes biasa, melainkan Gaussian Naive Bayes. Varian ini didesain untuk bekerja dengan data kontinyu, berbeda dengan Naive Bayes biasa yang didesain untuk bekerja dengan data kategorikal. Pada Gaussian Naive Bayes, persebaran fitur diasumsikan mengikuti distribusi normal sehingga fungsi kepadatan distribusinya dapat dirumuskan sebagai:

$$\log P(x|C_k) = \frac{-1}{2} \log(2\pi\sigma_k^2) - \frac{(x-\mu_k)^2}{2\sigma_k^2}$$

Dengan  $\sigma_k$  adalah standar deviasi fitur dari kelas  $C_k$  dan  $\mu_k$  adalah rata-rata fitur dari kelas  $C_k$ . Varian ini merupakan satu dari berbagai variasi algoritma Naive Bayes lainnya seperti Multinomial Naive Bayes, Bernoulli Naive Bayes, dan Categorical Naive Bayes.

b. Alur Kerja Algoritma

Algoritma Gaussian Naive Bayes secara umum memiliki alur kerja berikut:

- 1) Hitung nilai mean, standar deviasi atau variansi (tergantung pada rumus kerapatan distribusi yang digunakan), dan nilai prior dari masing-masing fitur untuk setiap kelas. Pada algoritma implementasi, tahap ini dilakukan oleh metode fit dan disimpan pada atribut kelas.
- 2) Kemudian, pada tahap inferensi atau prediksi, hitung log *likelihood* dengan fungsi kerapatan distribusi probabilitas untuk tiap titik data kemudian jumlahkan
- 3) Hitung nilai log posterior dengan menjumlahkan nilai log likelihood dengan nilai log prior untuk tiap kelas
- 4) Kemudian, pilih kelas dengan nilai log posterior tertinggi

c. Penjelasan Implementasi

Pada implementasi algoritma, kami membuat sebuah kelas bernama `GaussianNaiveBayes` yang terdiri dari tujuh (7) metode, yakni sebagai berikut.

Tabel 3.1 Penjelasan Kelas Implementasi

Kelas	Penjelasan
<pre>def __init__(self, var_smoothing = 1e-9):     self.classes = []     self.mean = {}     self.std = {}     self.priors = {}     self.var_smoothing = var_smoothing</pre>	Konstruktor kelas yang akan menginisiasi larik kelas, kamus rata-rata, kamus standar deviasi, kamus prior, dan variabel <code>var_smoothing</code> (dari parameter).
<pre>def fit(self, x, y):     self.classes = np.unique(y)      for c in self.classes:         x_c = x[y == c]         self.mean[c] = x_c.mean(axis=0)         self.std[c] = np.maximum(x_c.std(axis=0), self.var_smoothing)         self.priors[c] = len(x_c) / len(x)</pre>	<p>Metode ini menerima <code>x</code> (fitur) dan <code>y</code> (kelas) sebagai parameter. Lalu, metode mengisi larik kelas dengan nilai <code>y</code> (unik). Kemudian, dilakukan iterasi untuk seluruh isi dalam kelas, menginisiasi <code>x_c</code> sebagai titik data dari kelas saat ini, lalu menghitung mean, std, dan prior dari masing-masing kelas untuk tiap fitur (perhatikan sintaks <code>axis=0</code>).</p> <p>Sebagai antisipasi kemungkinan adanya fitur untuk suatu kelas yang memiliki variansi nol, digunakan fungsi <code>max</code> untuk</p>

	memilih antara variabel <code>var_smoothing</code> dan nilai <code>std</code> . Perlu diingat bahwa nilai <code>var_smoothing</code> sangatlah kecil, sehingga kemungkinan besar hanya akan terpilih jika variansinya nol.
<pre>def calculate_log_likelihood(self, x, mean, std):     return (-0.5 * np.sum(np.log(2 * np.pi * std**2))) - np.sum(((x - mean)**2) / (2 * std**2))</pre>	Metode ini menerima <code>x</code> (fitur), <code>mean</code> , dan <code>std</code> kemudian mengembalikan nilai dari fungsi kerapatan distribusinya.
<pre>def calculate_log_posterior(self, x):     log_posteriors = {}      for c in self.classes:         log_likelihood = self.calculate_log_likelihood(x, self.mean[c], self.std[c]) # Compute the log likelihood P(x   c)         log_posteriors[c] = np.log(self.priors[c]) + log_likelihood      return log_posteriors</pre>	Metode ini menerima <code>x</code> (fitur), membuat kamus untuk nilai log posterior, kemudian melakukan iterasi untuk seluruh kelas dalam larik kelas, menghitung nilai <i>likelihood</i> dengan metode <code>calculate_log_likelihood</code> , menjumlahkan nilainya dengan nilai log prior, lalu dimasukkan ke kamus log posterior. Terakhir, metode akan mengembalikan kamus log posterior tadi.
<pre>def predict(self, x):     x = np.array(x, dtype=np.float64)      y_pred = []      for item in x:         log_posteriors = self.calculate_log_posterior(item)         y_pred.append(max(log_posteriors, key=log_posteriors.get)) # Get the class with the highest posterior probability      return y_pred</pre>	Metode ini menerima <code>x</code> (fitur) sebagai parameter, menginisiasi larik kosong <code>y_pred</code> untuk menyimpan hasil kelas prediksi. Kemudian, metode akan melakukan iterasi untuk seluruh <i>item</i> di <code>x</code> , menghitung nilai posterior untuk tiap item dengan metode <code>calculate_posterior</code> , lalu diambil kelas dengan nilai probabilitas posterior tertinggi. Terakhir, metode akan mengembalikan larik <code>y_pred</code> tadi.
<pre>def save_model(self, filename):     with open(filename, 'wb') as file:         pickle.dump(self, file)     print(f'Model saved as {filename}')</pre>	Metode ini menerima parameter <i>filename</i> kemudian menggunakan fungsi dari pustaka <code>pickle</code> untuk menyimpan atribut kelas dengan format <code>.pkl</code> . Metode juga akan mencetak konfirmasi tersimpannya berkas model ke layar beserta nama berkasnya.



<pre>def load_model(filename):     with open(filename, 'rb') as file:         model = pickle.load(file)         print(f'Model {filename} has been loaded')      return model</pre>	<p>Metode ini menerima parameter <i>filename</i> kemudian menggunakan fungsi dari pustaka pickle untuk memuat berkas model berformat .pkl dengan nama sesuai parameter <i>filename</i>. Metode akan mencetak pesan konfirmasi bahwa model telah berhasil dimuat ke layar.</p> <p>Perlu diketahui bahwa metode ini mengasumsikan bahwa berkas model berada pada direktori yang sama dengan program.</p>
--	--

## 4. Algoritma ID3 (DTL)

### a. Deskripsi Singkat

ID3 adalah algoritma Machine Learning yang mengubah hasil data latih menjadi sebuah pohon keputusan. Dengan pohon keputusan ini nantinya, data asli dapat diprediksi akan memiliki suatu input apa berdasarkan setiap value instance dari setiap fiturnya. Selayaknya suatu pohon, pohon keputusan ini dapat mempunyai cabang yang nantinya dapat bercabang lagi, hingga nanti di paling ujung disebut sebagai daun. Cabang disini berarti adalah hasil dari kemungkinan yang dimiliki oleh suatu *node* dari sebuah fitur dan daun berarti hasil prediksi dari target fitur yang diperlukan. Pada realisasinya, jika tipe data dari sebuah fitur merupakan bilangan angka kontinu, maka titik temu bisa memiliki sebanyak tak hingga cabang. Karena itu, tahap pertama dari pembuatan pohon ini adalah mendiskritkan data numerik. Untuk mendiskritkan data ini, digunakan metode menguji semua potensi breakpoint yang optimal dengan target memecah tiap fitur menjadi 2 bagian. Semua potensi ini nantinya perlu diadu dengan cara mencari gain terbesar dengan menguji setiap potensi thresholdnya. Selanjutnya jika threshold setiap fitur ditemukan barulah *training* bisa dilakukan. Yaitu dengan mencari gain terbesar dari setiap fitur dan menjadikan fitur tersebut adalah titik temu untuk cabang selanjutnya, dan seterusnya hingga tidak bisa dibuat cabang lagi atau tidak perlu dibuat cabang lagi karena titik temu selanjutnya bisa langsung fitur target.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Entropy(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i$$

Gain yang dimaksud adalah Information Gain atau informasi yang dapat didapatkan. Dari semua fitur yang tersedia, dicari Gain terbesar, artinya fitur dengan Gain terbesar tersebut adalah yang terbaik untuk dijadikan percabangan selanjutnya dibandingkan fitur-fitur lain menurut metode gain ini. Lalu entropinya adalah nilai dari masing-masing isi data yang unik terhadap outputnya, yaitu ingin dilihat seberapa berpengaruh isi data tersebut. Jika terlalu besar pengaruhnya, maka akan dengan mudah terjadi overfitting dan berbagai masalah lainnya yang mengakibatkan informasi yang didapat dari fitur tersebut menjadi kecil.

### **b. Alur Kerja Algoritma**

Algoritma ketika *training*

- 1) Pertama adalah mencari breakpoint dari fitur target, yaitu titik yang kemungkinan dapat menjadi Threshold optimal.
- 2) Untuk setiap fitur yang numerik, cari Threshold dengan gain terbesar. Lalu ubah setiap isi datanya (dari setiap fitur numerik tadi) menjadi boolean atau biner (dengan 0 berarti false dan 1 berarti true)
- 3) Untuk membuat pohon keputusannya, pertama dipilih terlebih dahulu fitur dengan gain terbesar. Lalu fitur tersebut akan menjadi titik temu cabang pertama. Selanjutnya dilakukan pemanggilan ulang fungsi untuk melakukan hal yang sama. Dicari kembali fitur dengan gain terbesar selanjutnya dan dijadikan titik temu cabang selanjutnya. Hal ini dilakukan terus menerus hingga hanya tersisa fitur target atau di dalam fitur target hanya ada 1 data unik. Jika sudah ketemu maka ini adalah ujung dari rekursi dan menjadi daun dari ujung-ujung cabang
- 4) Terakhir pohon ini dikembalikan dalam bentuk *dictionary* dan threshold juga perlu dikembalikan untuk mentransformasi data yang akan diprediksi nantinya

Algoritma ketika *predicting*

- 1) Data didiskritkan dengan threshold yang sama
- 2) Untuk setiap baris data, diprediksi berdasarkan pohon dengan pertama menyesuaikan kemungkinan dari titik temu utama dan di rekursi seterusnya ke cabang-cabang selanjutnya sesuai dengan data yang ada pada baris tersebut hingga mencapai daun. Daun yang ditemukan tersebut adalah hasil prediksi untuk baris tersebut

### c. Penjelasan Implementasi

Untuk mengimplementasikan bagian mencari threshold sebenarnya ini sangat memakan waktu karena dengan data yang sangat variatif bisa mencapai 90% atau lebih dari keseluruhan data. Karena itu dipilih 40 pertama saja untuk mendapatkan threshold dari model. Selain itu kode masih banyak kekurangan karena belum bisa mengimplementasikan semua kondisi error seperti tipe data yang masih salah di casting, jumlah iterasi yang bisa jadi melebihi index nya, dan beberapa hal lainnya

Hingga saat ini, model tree berhasil dibuat dalam bentuk dictionary. Akan tetapi karena beberapa kemungkinan kesalahan tersebut, tree masih tidak sempurna sehingga ketika dilakukan predict masih terdapat yang output None. Pada predict juga masih kekurangan dalam handle kondisi di luar normal

## 5. Penjelasan Exploratory Data Analysis (EDA)

Data yang akan dicek adalah dari train. Akan dicek terlebih dahulu jumlah baris, jumlah kolom, dan tipe data yang menghasilkan seperti berikut.

```
Jumlah baris: 175341, Jumlah kolom: 44
swin          float64
dwin          float64
stcpb         float64
dtcpb         float64
smean        float64
dmean        float64
trans_depth   float64
response_body_len float64
id            int64
proto         object
attack_cat    object
label        int64
sjit         float64
djit         float64
sinpkt       float64
dinpkt       float64
tcprrt       float64
synack       float64
ackdat       float64
is_sm_ips_ports float64
ct_state_ttl float64
ct_flw_http_mthd float64
is_ftp_login  float64
ct_ftp_cmd    float64
ct_srv_src    float64
ct_srv_dst    float64
ct_dst_ltm    float64
ct_src_ltm    float64
ct_src_dport_ltm float64
ct_dst_sport_ltm float64
ct_dst_src_ltm float64
state         object
dur           float64
sbytes        float64
dbytes        float64
sttl          float64
dttl          float64
sloss         float64
dloss         float64
service       object
sload         float64
dload         float64
spkts         float64
dpkts         float64
```

**Gambar 5.1 Jumlah Baris, Kolom, dan Tipe Data**

Dari hasil tersebut, terlihat bahwa pembagian antara categorical features dan numerical features adalah sebagai berikut dengan jumlah unique value yaitu sebagai berikut.

**Tabel 5.1 Categorical Features dan Numerical Features**

Categorical Features			Numerical Features		
Nama Feature	Unique Values	Tipe Data	Nama Feature	Unique Values	Tipe Data
proto	133	object	swin	11	float64
attack_cat	10	object	dwin	7	
state	9	object	stcpb	71744	
service	13	object	dtcpb	71536	
			smean	1356	
			dmean	1322	
			trans_depth	11	
			response_body_len	2327	
			id	175341	int64
			label	2	
			sjit	73895	float64
			djit	73045	
			sinpkt	72757	
			dinpkt	70746	
			tcprtt	41517	
			synack	38626	
			ackdat	36365	
			is_sm_ips_ports	2	
			ct_state_ttl	5	

	ct_flw_http_mthd	11	
	is_ftp_login	4	
	ct_ftp_cmd	4	
	ct_srv_src	52	
	ct_srv_dst	52	
	ct_dst_ltm	50	
	ct_src_ltm	50	
	ct_src_dport_ltm	47	
	ct_dst_sport_ltm	32	
	ct_dsr_src_ltm	54	
	dur	70713	
	sbytes	6997	
	dbytes	6432	
	sttl	11	
	dttl	6	
	sloss	394	
	dloss	365	
	sload	77315	
	dload	73756	
	spkts	470	
	dpkts	432	

Berikut adalah pengecekan nilai count, mean, std, min, 25%, 50%, 75%, dan max untuk masing-masing features.

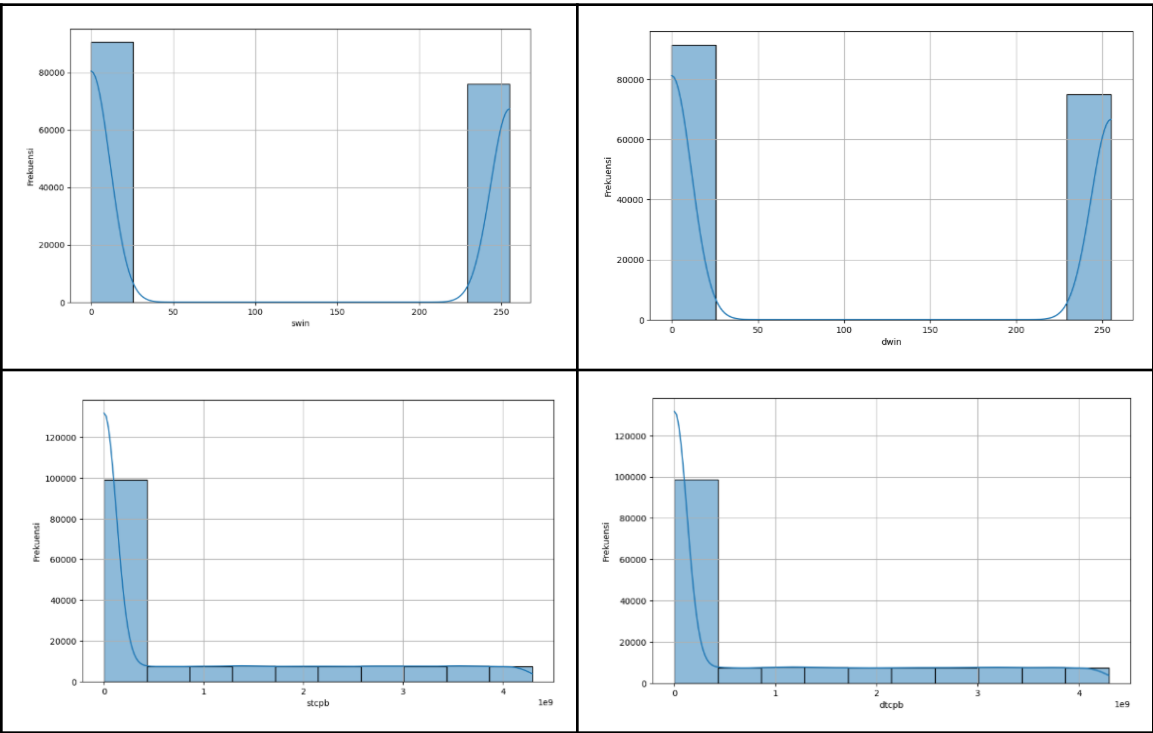
**Tabel 5.2 Describe Features**

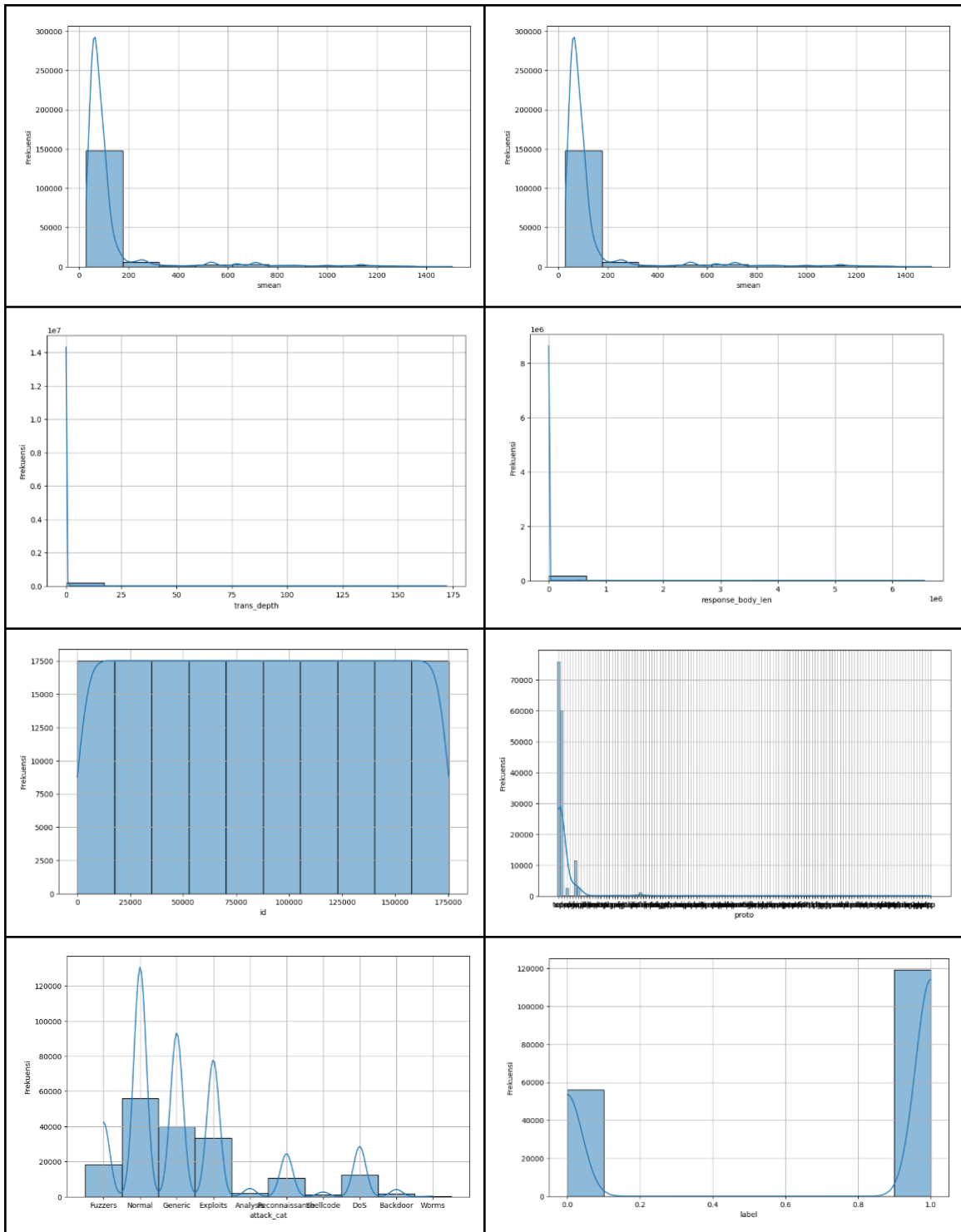
	swin	dwin	stcpb	dtcpb	smean	dmean	trans_depth	response_body_len	id	label	...
count	166801.000000	166862.000000	1.666690e+05	1.665380e+05	166553.000000	166488.000000	166558.000000	1.665500e+05	175341.000000	175341.000000	...
mean	116.184837	114.956407	9.696210e+08	9.693479e+08	136.803840	124.161041	0.106193	2.157133e+03	87670.000000	0.880622	...
std	126.994753	126.880855	1.355284e+09	1.354113e+09	204.753104	258.265755	0.794070	5.509898e+04	50616.731112	0.468237	...
min	0.000000	0.000000	0.000000e+00	0.000000e+00	28.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	...
25%	0.000000	0.000000	0.000000e+00	0.000000e+00	57.000000	0.000000	0.000000	0.000000e+00	43835.000000	0.000000	...
50%	0.000000	0.000000	0.000000e+00	0.000000e+00	73.000000	44.000000	0.000000	0.000000e+00	87670.000000	1.000000	...
75%	255.000000	255.000000	1.915654e+09	1.912673e+09	100.000000	89.000000	0.000000	0.000000e+00	131505.000000	1.000000	...
max	255.000000	255.000000	4.294959e+09	4.294882e+09	1504.000000	1458.000000	172.000000	6.558056e+06	175340.000000	1.000000	...

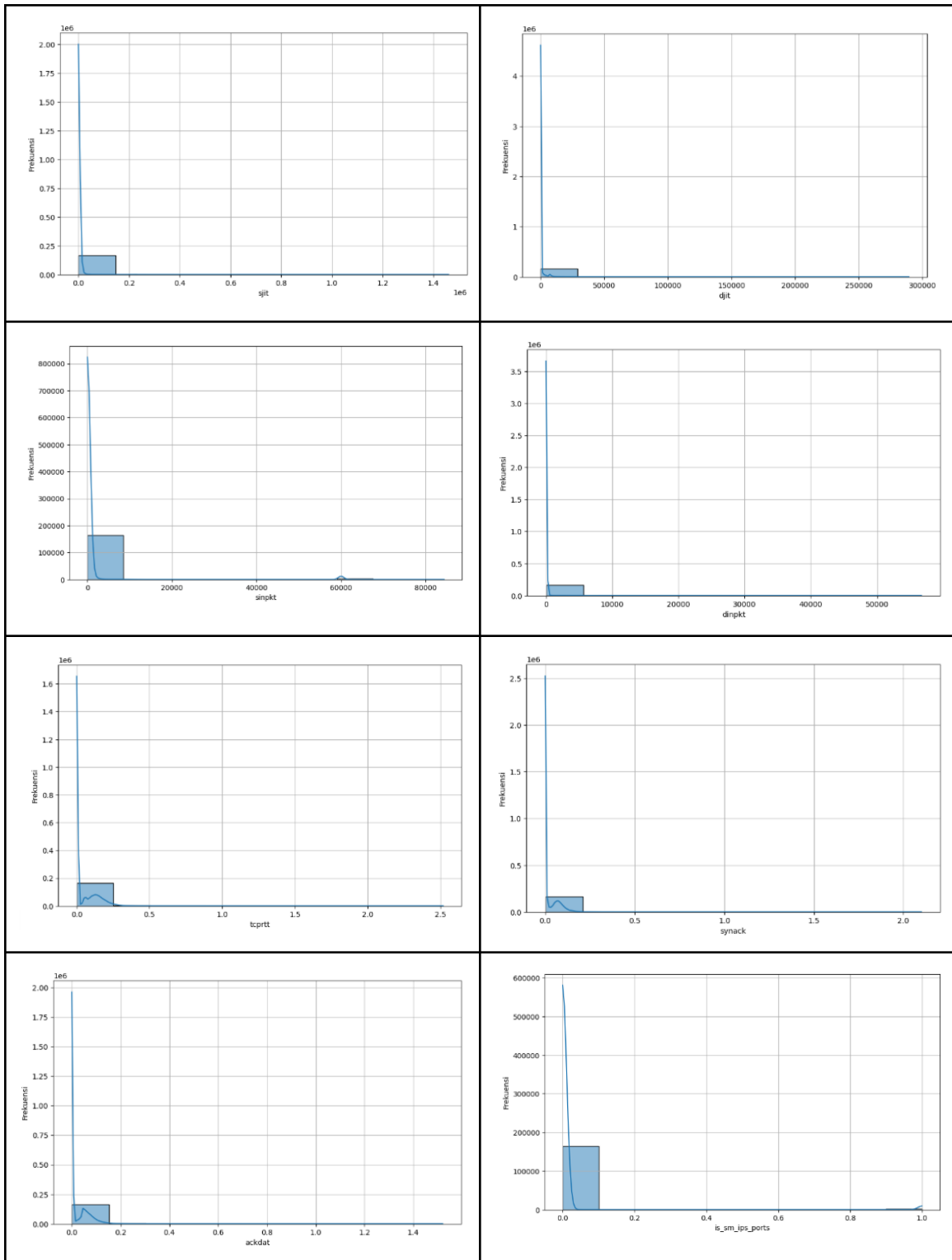
sbytes	dbytes	sttl	dttl	sloss	dloss	sload	dload	spkts	dpkts
1.667800e+05	1.664720e+05	166516.000000	166687.000000	166547.000000	166363.000000	1.665550e+05	1.665040e+05	166687.000000	166655.000000
8.734079e+03	1.493589e+04	179.521944	79.584179	4.904790	6.971888	7.342186e+07	6.718596e+05	20.266134	18.923015
1.712289e+05	1.430150e+05	102.957427	110.494848	64.867849	51.801936	1.884564e+08	2.422879e+06	136.433895	110.898133
2.800000e+01	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	1.000000	0.000000
1.140000e+02	0.000000e+00	62.000000	0.000000	0.000000	0.000000	1.303032e+04	0.000000e+00	2.000000	0.000000
4.500000e+02	1.640000e+02	254.000000	29.000000	0.000000	0.000000	8.714544e+05	1.426630e+03	2.000000	2.000000
1.418000e+03	1.096000e+03	254.000000	252.000000	3.000000	2.000000	8.888889e+07	2.807410e+04	12.000000	10.000000
1.296523e+07	1.465555e+07	255.000000	254.000000	4803.000000	5484.000000	5.988000e+09	2.242273e+07	9616.000000	10974.000000

Selanjutnya akan dilakukan visualisasi histogram dan KDE dari masing-masing feature yaitu sebagai berikut.

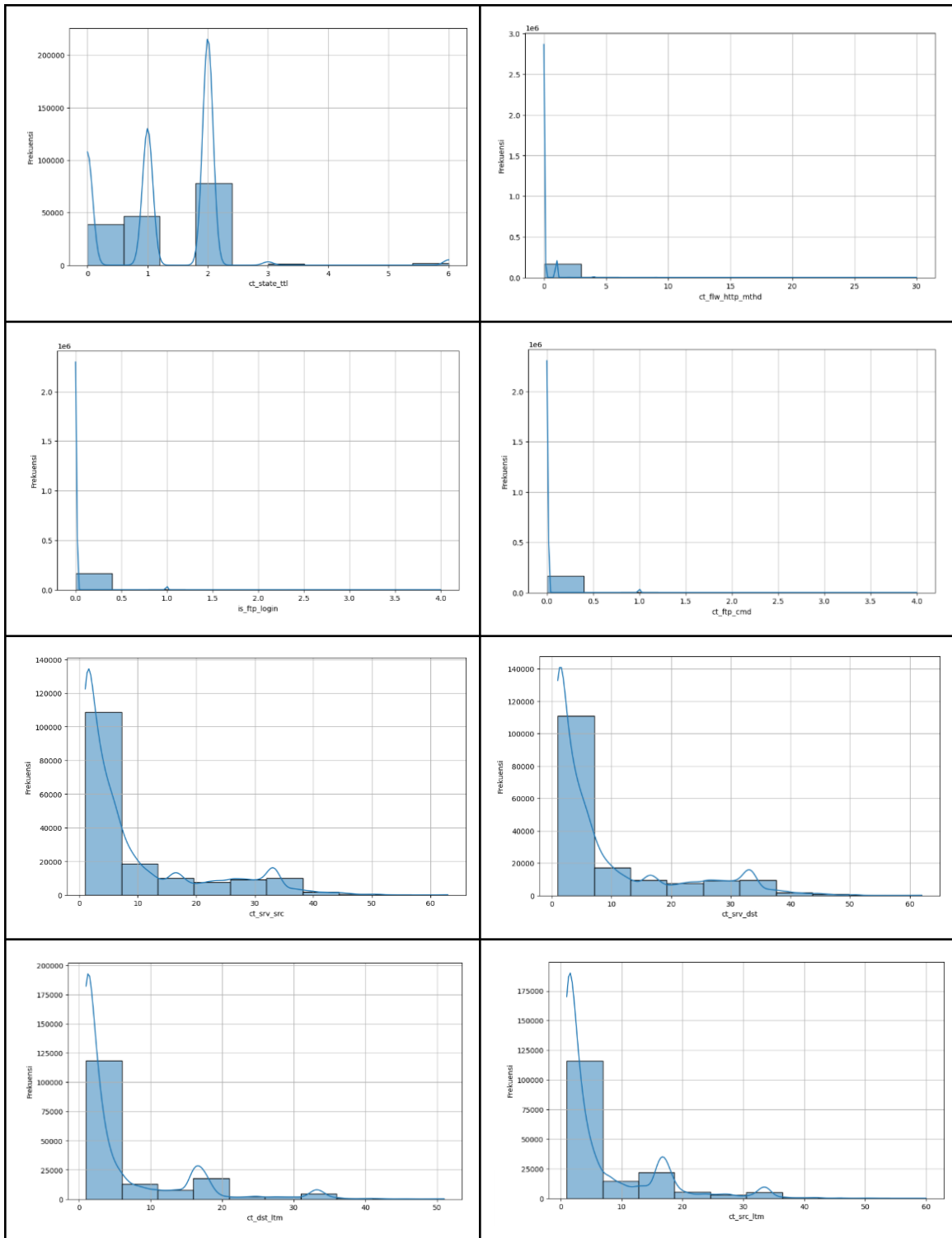
Tabel 5.3 Visualisasi Histogram dan KDE

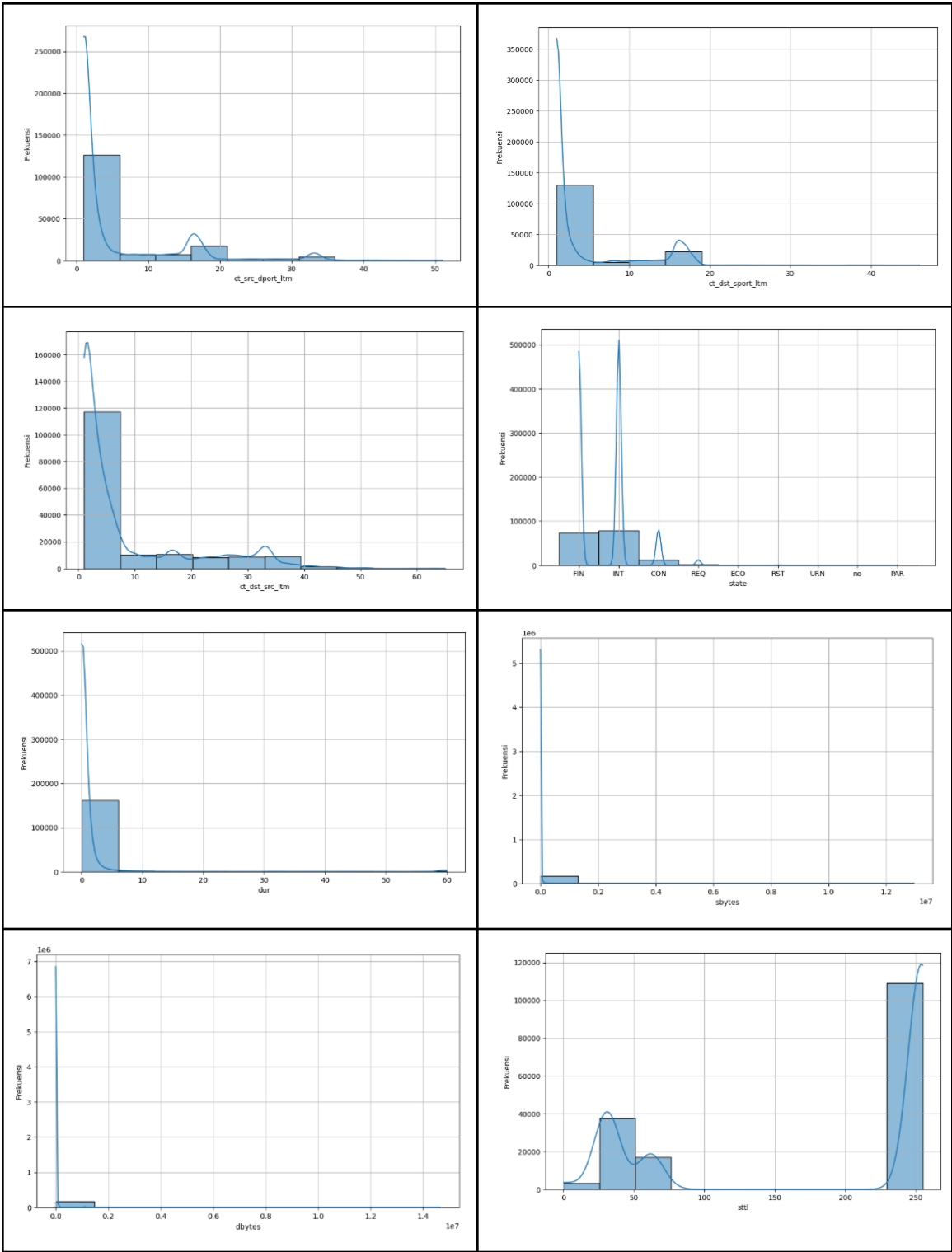


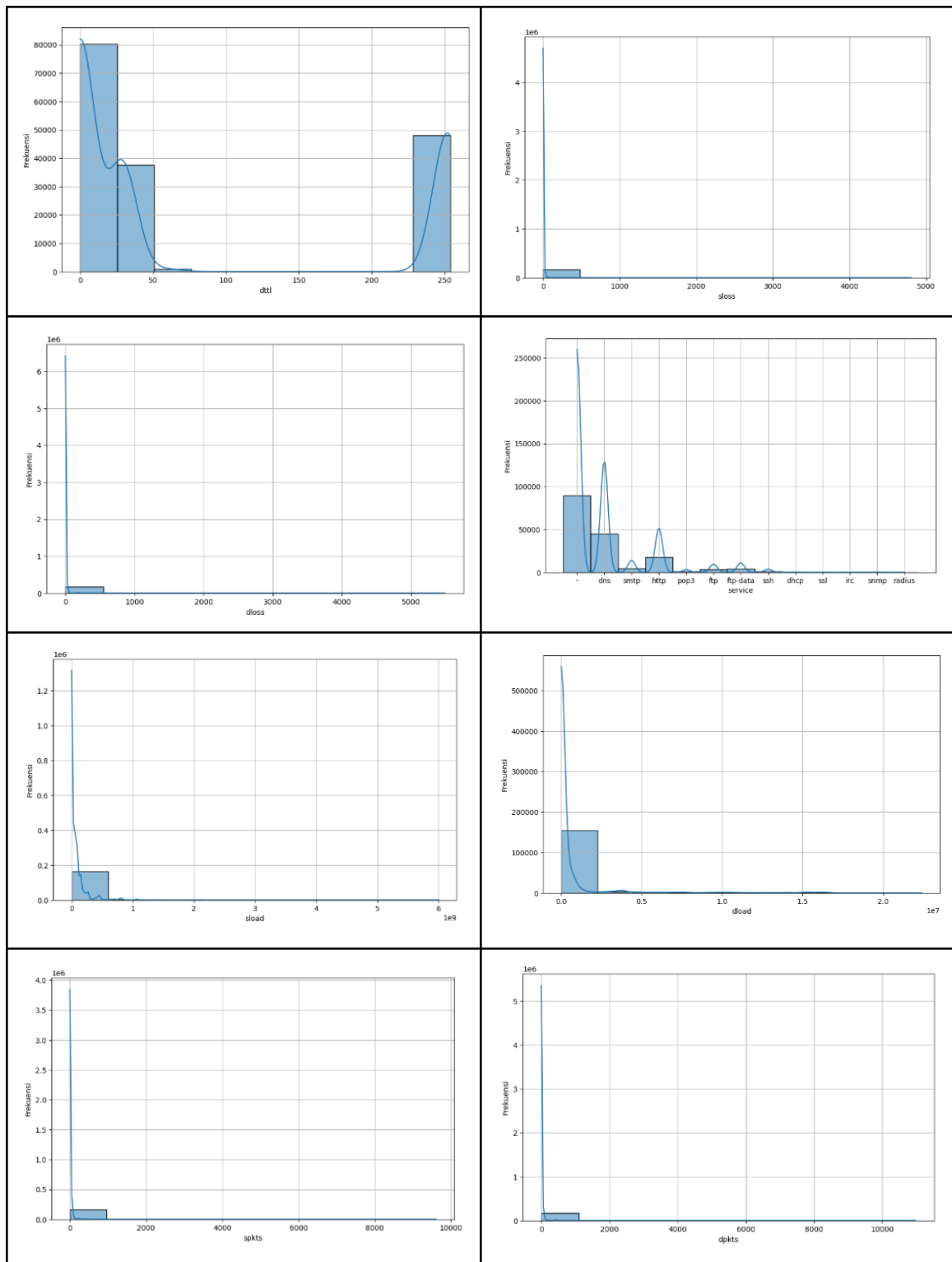




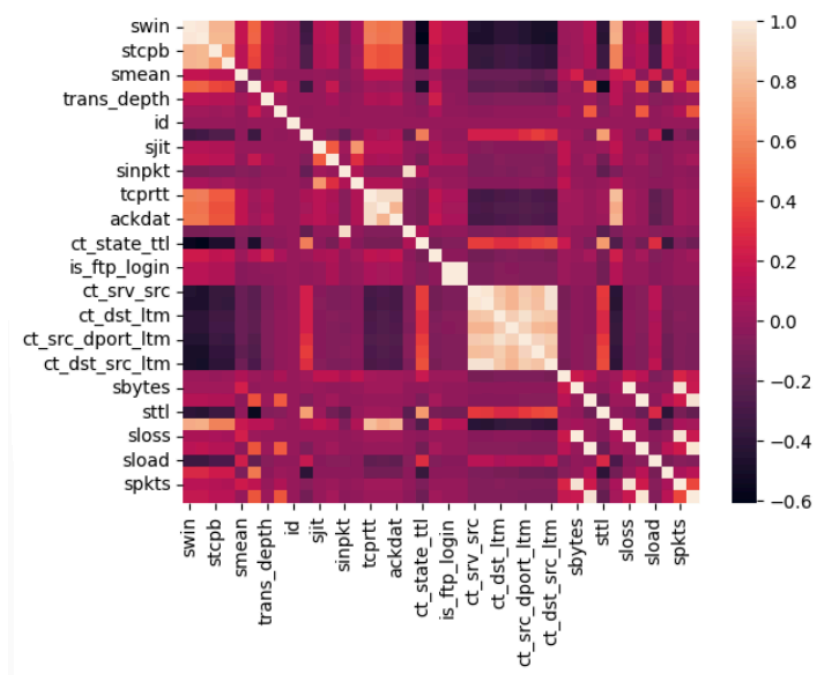








Selanjutnya akan dicek untuk korelasi antar numerical features menggunakan heatmap yaitu sebagai berikut.



Gambar 5.2 Heatmap Numerical Features

Dari hasil heatmap tersebut, berikut adalah hubungan yang kuat dari beberapa features.

- 1. swin, stcpb, smean
- 2. tcprrt, ackdat
- 3. ct\_srv\_src, ct\_dst\_ltm, ct\_src\_dport\_ltm, ct\_dst\_src\_ltm

6. Penjelasan Data Cleaning & Preprocessing

a. Data Cleaning

1. Handling Missing Data

Sebelum melakukan handling missing data, akan dicek terlebih dahulu missing values dari masing-masing features yang ada memanfaatkan .isnull().sum(). Hasil dari missing valuenya adalah sebagai berikut.

Tabel 6.1 Cek Missing Values

Categorical Features		Numerical Features	
Nama Feature	Missing Values	Nama Feature	Missing Values
proto	8826	swin	8740
attack_cat	0	dwin	8779

state	8805	stcpb	8672
service	8791	dtcpb	8803
		smean	8788
		dmean	8855
		trans_dep th	8785
		response_ body_len	8791
		id	0
		label	0
		sjit	8738
		djit	8846
		sinpkt	8707
		dinpkt	8734
		tcprrt	8836
		synack	8736
		ackdat	8595
		is_sm_ips _ports	8746
		ct_state_tt l	8635
		ct_flw_htt p_mthd	8647
		is_ftp_logi n	8647
		ct_ftp_cm d	8842
		ct_srv_src	8851

	ct_srv_dst	8774
	ct_dst_ltm	8738
	ct_src_ltm	8823
	ct_src_dp ort_ltm	8775
	ct_dst_sp ort_ltm	8788
	ct_dsr_src _ltm	8895
	dur	8722
	sbytes	8561
	dbytes	8869
	sttl	8825
	dttl	8654
	sloss	8794
	dloss	8978
	sload	8786
	dload	8837
	spkts	8654
	dpkts	8686

Karena dengan jumlah missing null values tersebut, akan dilakukan handling dengan cara yang terbagi antara categorical features dan numerical features. Untuk categorical features, akan dilakukan pengisian missing values dengan menggunakan mode/modus/most frequent dan untuk numerical features akan dilakukan pengisian missing values dengan menggunakan median. Selanjutnya, akan dipisahkan antara categorical dan numerical features sesuai yang sudah dilakukan sebelumnya dan akan dilakukan pengisian missing values dengan memanfaatkan SimpleImputer yaitu sebagai berikut.

**Tabel 6.2 Implementasi SimpleImputer**

```
# Pengisian missing value untuk categorical dan numerical
# Categorical
categorical_features_imputer =
SimpleImputer(strategy='most_frequent')
df_cleaning[categorical_features] =
categorical_features_imputer.fit_transform(df_cleaning[categor
ical_features])
# Numerical
numerical_features_imputer = SimpleImputer(strategy='median')
df_cleaning[numerical_features] =
numerical_features_imputer.fit_transform(df_cleaning[numerical
_features])
```

Setelah dilakukan pengisian missing values, setelah dicek nilai missing values sudah menjadi 0.

## 2. Dealing with Outliers

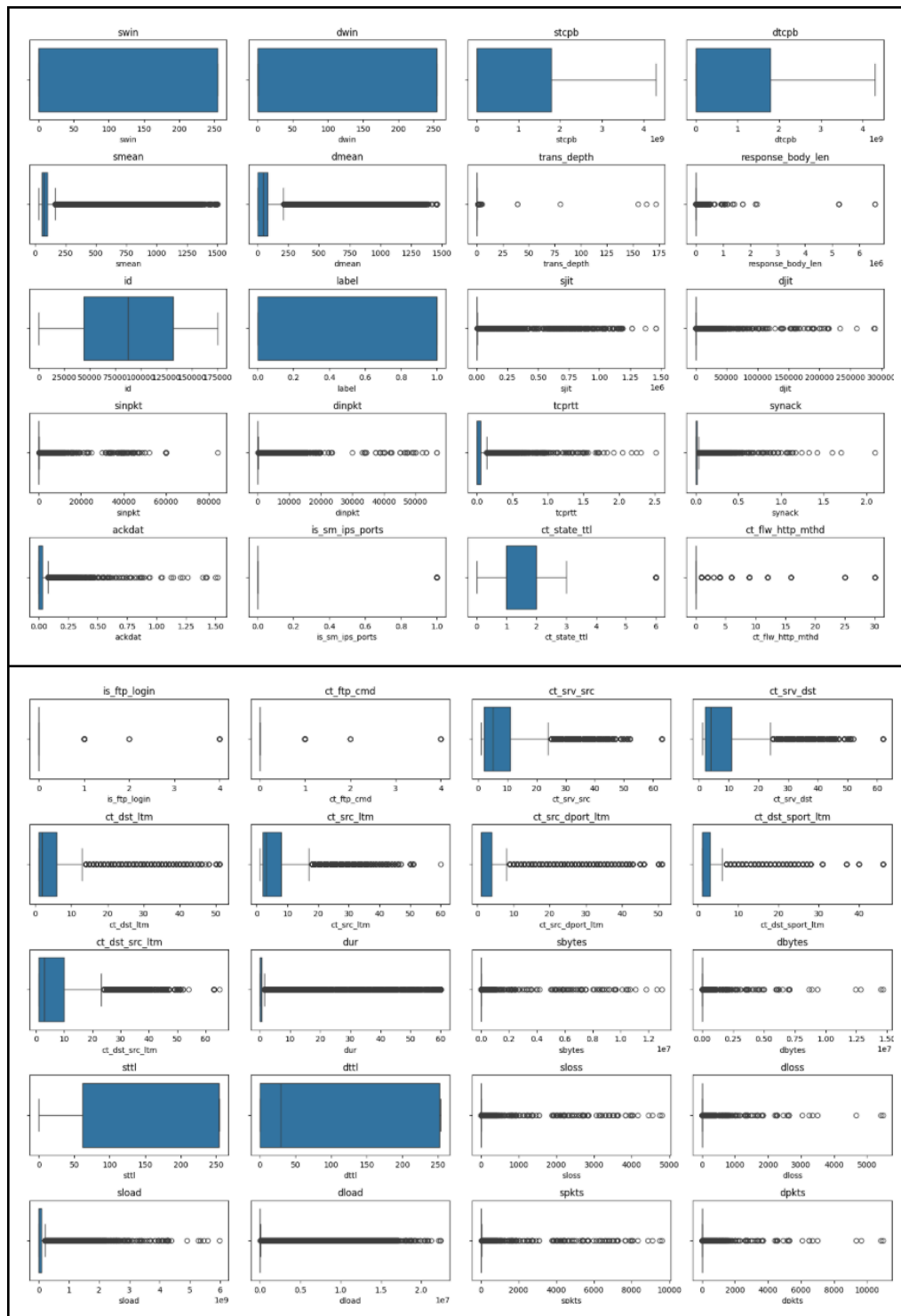
Setelah mengisi missing values, akan dilakukan proses dealing with outliers di mana sebelumnya akan dicek terlebih dahulu dari uji normalitas untuk menentukan apakah distribusi yang terjadi normal atau tidak. Metode yang digunakan adalah Shapiro-Wilk Test, yang akan dicek jika nilai p-value di bawah 0,05 maka distribusi tidak normal. Hasilnya adalah sebagai berikut.

```
Kolom: uid, Shapiro-Wilk Test: Statistic=0.6308709916870257, p-value=3.2994337391357935e-165
Kolom: dwin, Shapiro-Wilk Test: Statistic=0.6291819314716115, p-value=2.6953698041026904e-165
Kolom: stcpb, Shapiro-Wilk Test: Statistic=0.7137456217249332, p-value=1.8277860192547589e-156
Kolom: dtcpb, Shapiro-Wilk Test: Statistic=0.713826922359823, p-value=1.8499980836311631e-156
Kolom: dmean, Shapiro-Wilk Test: Statistic=0.4237168921733693, p-value=1.999456583517047e-180
Kolom: dmean, Shapiro-Wilk Test: Statistic=0.5178253251305147, p-value=3.183975651835143e-174
Kolom: trans_depth, Shapiro-Wilk Test: Statistic=0.859582854556735176, p-value=5.824836889399595e-198
Kolom: response_body_len, Shapiro-Wilk Test: Statistic=0.81492126935838185, p-value=9.78931785635472e-288
Kolom: id, Shapiro-Wilk Test: Statistic=0.9549381233058609, p-value=3.41286173182517e-182
Kolom: label, Shapiro-Wilk Test: Statistic=0.587358230574706, p-value=6.76598869285238e-169
Kolom: xjit, Shapiro-Wilk Test: Statistic=0.86824339893067644, p-value=1.899733261131384e-197
Kolom: djit, Shapiro-Wilk Test: Statistic=0.1894709886222637, p-value=5.822528973899351e-196
Kolom: xipkt, Shapiro-Wilk Test: Statistic=0.1079120876261945, p-value=4.33494335452785e-196
Kolom: dirpkt, Shapiro-Wilk Test: Statistic=0.846856892538362116, p-value=1.6083923621946138e-198
Kolom: tcprtt, Shapiro-Wilk Test: Statistic=0.5565874851415729, p-value=2.40059542338347e-171
Kolom: synack, Shapiro-Wilk Test: Statistic=0.514664812925786, p-value=1.8951276590841216e-174
Kolom: ackdat, Shapiro-Wilk Test: Statistic=0.5387163830653184, p-value=2.72919321926861e-173
Kolom: is_sm_ips_ports, Shapiro-Wilk Test: Statistic=0.89711575283398833, p-value=1.5718898172297214e-196
Kolom: ct_state_ttl, Shapiro-Wilk Test: Statistic=0.7717088648260992, p-value=2.876793721298808e-149
Kolom: ct_file_http_mtd, Shapiro-Wilk Test: Statistic=0.15114371701887983, p-value=2.833788636737834e-194
Kolom: is_ftp_login, Shapiro-Wilk Test: Statistic=0.88887182381457825, p-value=7.288035434856641e-197
Kolom: ct_ftp_cmd, Shapiro-Wilk Test: Statistic=0.88925684477671336, p-value=7.563214132669397e-197
Kolom: ct_srv_src, Shapiro-Wilk Test: Statistic=0.749215882951119, p-value=2.8512680334052e-152
Kolom: ct_srv_dst, Shapiro-Wilk Test: Statistic=0.737871381644867, p-value=7.531375441632209e-154
Kolom: ct_dst_ltm, Shapiro-Wilk Test: Statistic=0.666483892160857, p-value=9.581281788261805e-162
Kolom: ct_src_ltm, Shapiro-Wilk Test: Statistic=0.666483892160857, p-value=9.581281788261805e-162
Kolom: ct_src_sport_ltm, Shapiro-Wilk Test: Statistic=0.596489960485343, p-value=1.82212477588897e-168
Kolom: ct_dst_sport_ltm, Shapiro-Wilk Test: Statistic=0.5836437802147038, p-value=3.357641361563217e-169
Kolom: ct_dst_src_ltm, Shapiro-Wilk Test: Statistic=0.7879422576717659, p-value=2.26623480661828564e-157
Kolom: dur, Shapiro-Wilk Test: Statistic=0.7132756334899113, p-value=1.807665825987934e-192
Kolom: sbytes, Shapiro-Wilk Test: Statistic=0.81972541842974395, p-value=1.483241179887427e-199
Kolom: dbytes, Shapiro-Wilk Test: Statistic=0.8673131840801098, p-value=1.8106924618626836e-197
Kolom: sttl, Shapiro-Wilk Test: Statistic=0.6228772447477354, p-value=6.318914616764134e-166
Kolom: dttl, Shapiro-Wilk Test: Statistic=0.62963408984725, p-value=3.836297518419887e-165
Kolom: sloss, Shapiro-Wilk Test: Statistic=0.829342468828818546, p-value=3.429315706934368e-199
Kolom: dloss, Shapiro-Wilk Test: Statistic=0.88775808841397169, p-value=6.577927819540885e-197
Kolom: sload, Shapiro-Wilk Test: Statistic=0.3728029338187335, p-value=1.8722899809514685e-183
Kolom: dload, Shapiro-Wilk Test: Statistic=0.2957287198317595, p-value=1.558221957888135e-187
Kolom: spkts, Shapiro-Wilk Test: Statistic=0.86962813884273374, p-value=1.2472165414699785e-197
Kolom: dpkts, Shapiro-Wilk Test: Statistic=0.11442537313585976, p-value=8.8364369811416e-196
```

**Gambar 6.1 Hasil Shapiro-Wilk Test**

Dari hasil tersebut, terlihat bahwa semua kolom memiliki nilai p-value yang berada jauh di bawah 0,05 yang menunjukkan semua kolom tidak terdistribusi dengan normal. Selanjutnya, akan dicek outliersnya dari boxplot yang mana hasilnya adalah sebagai berikut.

**Tabel 6.3 Cek Outliers dengan Boxplots**



Terlihat bahwa beberapa features memiliki outliers. Selanjutnya akan dilakukan identifikasi outliers dengan menggunakan metode IQR. Metode IQR akan menghitung Q1 yaitu  $\text{quantile}(0,5)$ , Q3 yaitu  $\text{quantile}(0,75)$ , dan IQR yang merupakan selisih dari Q3 dan Q1.



Selanjutnya akan ditentukan batas bawah yaitu  $Q1 - 1,5 * IQR$  dan batas atas yaitu  $Q3 + 1,5 * IQR$ . Untuk mengidentifikasi outliers, akan dicek dengan nilai yang berada di luar batas bawah dan batas atas. Hasil outliersnya adalah sebagai berikut.

**Tabel 6.4 Jumlah Outliers**

Numerical Features	
Nama Feature	Outliers
swin	0
dwin	0
stcpb	0
dtcpb	0
smean	19690
dmean	22697
trans_depth	16948
response_body_len	10723
id	0
label	0
sjit	20205
djit	20609
sinpkt	15717
dinpkt	17320
tcprrt	21852
synack	38551
ackdat	13634
is_sm_ips_ports	2632
ct_state_ttl	1838
ct_flw_http_mthd	16996

is_ftp_login	2443
ct_ftp_cmd	2443
ct_srv_src	22771
ct_srv_dst	22825
ct_dst_ltm	30290
ct_src_ltm	18116
ct_src_dport_ltm	35713
ct_dst_sport_ltm	35713
ct_dsr_src_ltm	24432
dur	15994
sbytes	23142
dbytes	30374
sttl	0
dttl	0
sloss	27947
dloss	26996
sload	12907
dload	39592
spkts	23470
dpkts	19752

Setelah itu, akan dilakukan dealing with outliers menggunakan clipping. Clipping akan melakukan perubahan pada nilai bawah dengan mengubahnya menjadi batas bawah dan nilai atas dengan mengubahnya menjadi batas atas yang mana implementasinya adalah sebagai berikut.

**Tabel 6.5 Implementasi Clipping**

```
# Clipping outliers
```

```
for columns in numerical_features_outliers:
    numerical_features_outliers[columns] =
numerical_features_outliers[columns].clip(lower=lower_bound[columns], upper=upper_bound[columns])
```

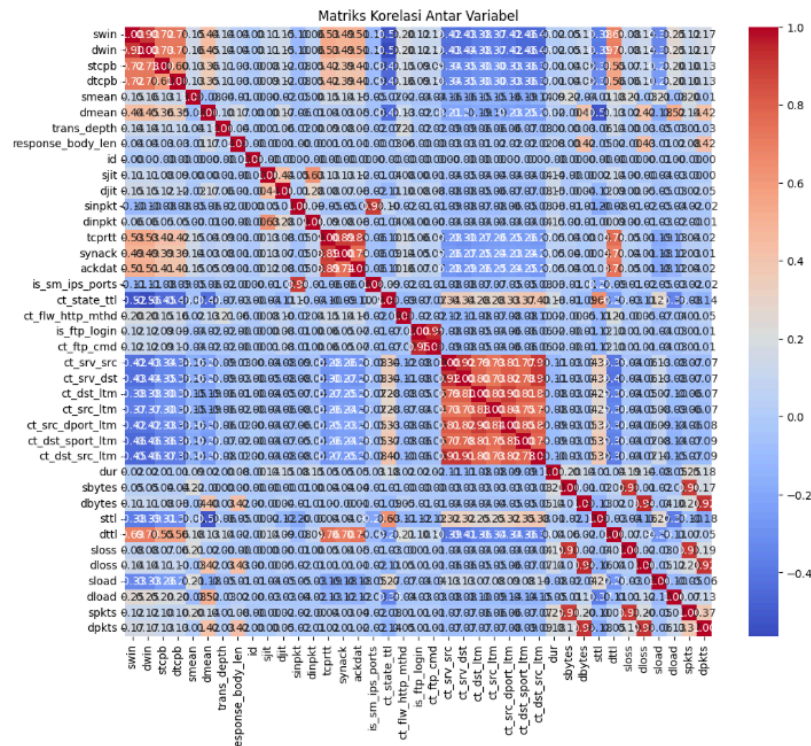
Setelah dilakukan clipping, jumlah outliersnya sudah menjadi 0 semua.

### 3. Remove Duplicates

Tahap selanjutnya yang dilakukan adalah remove duplicates dengan memanfaatkan `.drop_duplicates()`. Setelah pengecekan duplicates tidak ada duplikat dan tetap akan dilakukan remove duplicates sehingga jumlah data sebelum dan sesudah remove duplicates sama.

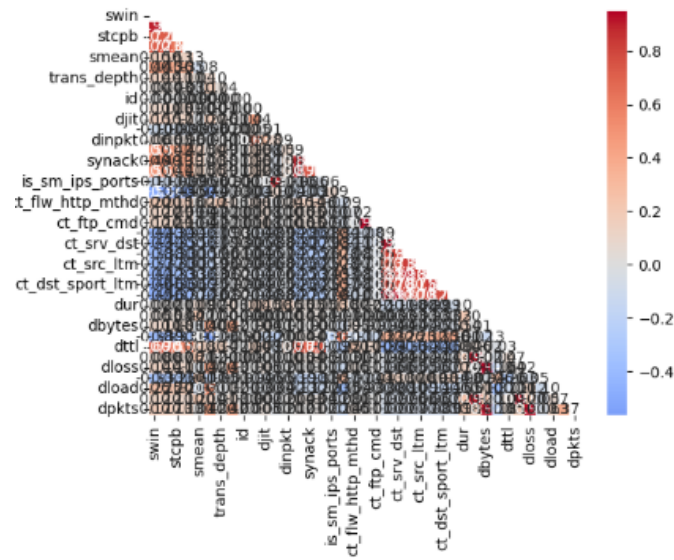
### 4. Feature Engineering

Tahap terakhir dari data cleaning adalah feature engineering. Sebelum dilakukan feature engineering, akan dilakukan drop feature label karena tidak akan digunakan. Lalu akan dilakukan visualisasi korelasi dengan menggunakan heatmap pada numerical features yaitu sebagai berikut.



**Gambar 6.2 Heatmap Numerical Features**

Selanjutnya, akan dicoba dilakukan mask dari korelasi sebelumnya menghasilkan segitiga bawah dari segitiga atas hasil seperti berikut.



**Gambar 6.3 Mask Numerical Features**

Proses feature engineering yang akan dilakukan dimulai dari menghitung matriks korelasi antara numerical features tersebut, yang menunjukkan hubungan antarannya. Selanjutnya, akan dilakukan pembuatan mask yang mana menghasilkan setengah dari segitiga matriks tersebut dan akan dilakukan drop pada features yang memiliki korelasi yang tinggi yaitu di atas 0,95. Hal ini dikarenakan akan dilakukan filtering dan menghilangkan features yang berkorelasi sangat tinggi tersebut, di mana dari heatmap korelasi akan menghasilkan dua kali di mana korelasi antara A ke B akan sama dengan korelasi B ke A sehingga akan diabaikan untuk setengah dari korelasi matriks tersebut sehingga memanfaatkan mask sebelumnya. Dengan melakukan penghapusan pada features yang memiliki korelasi yang tinggi, akan membantu membuat model menjadi lebih sederhana, meningkatkan stabilitasnya, serta mempercepat komputasi. Dari hasil tersebut, terlihat bahwa tidak ada yang memiliki hubungan yang sangat tinggi sehingga jumlah kolom yang masih sama dengan sebelumnya yaitu 43 kolom.

## **b. Data Preprocessing**

### **1. Feature Scaling**

Feature scaling dilakukan untuk memastikan semua numerikal fitur dalam dataset memiliki skala yang seragam. Proses ini perlu dilakukan karena tanpa scaling, fitur dengan skala yang lebih besar dapat mendominasi model yang mana akan mengurangi akurasi dan memperlambat proses pelatihan. Seperti yang telah dilakukan pada bagian *Exploratory Data Analysis* sudah dilakukan proses untuk statistik

deskriptif dari dataset yang mencakup informasi dasar dari semua fitur numeriknya dengan hasil yang ditunjukkan pada tabel 5.3. Dari proses itu dapat diketahui gambaran umum mengenai distribusi data, dengan contoh seperti berikut.

**Tabel 6.6 Contoh Describe Features Numerik**

	swin	stcpb	smean	sloss	dload
count	166601.0 00000	1.666690 e+05	166553.0 00000	166547.0 00000	1.665040 e+05
mean	116.18483 7	9.696210 e+08	136.8038 40	4.904790	6.718596 e+05
min	0.000000	0.000000 e+00	28.00000 0	0.000000	0.000000 e+00
max	255.0000 00	4.294959 e+09	1504.000 000	4803.000 000	2.242273 e+07

Dari tabel 6. Kita ketahui bahwa benar adanya distribusi data numerik yang sangat luas, ada yang di skala satuan, ribuan, hingga milyar an sehingga proses feature scaling akan dilakukan.

Dalam melakukan feature scaling digunakan metode *Standardization (Z-score Scaling)* pada fitur numerik datasetnya. Metode ini akan mengubah skala data sehingga tiap fitur numeriknya memiliki nilai *mean* 0 dan *standard deviation* 1. Dengan menerapkan metode ini, yang sebelumnya ada fitur yang memiliki skala beragam rendah hingga tinggi akan memiliki skala yang seragam. Dalam prosesnya fitur 'id' yang termasuk dalam kelompok numerik akan dikecualikan untuk feature scaling karena id merupakan identifier yang perlu dipertahankan bentuknya. Berikut adalah cuplikan kode yang digunakan.

**Tabel 6.7 Implementasi Scaling**

```
scaler = StandardScaler()  
train_set_split[train_numerical_features_without_id] =  
scaler.fit_transform(train_set_split[train_numerical_features_  
without_id])
```

Setelah proses scaling berhasil diterapkan, distribusi data untuk tiap fitur numeriknya menjadi seragam dengan contoh hasil seperti berikut.

**Tabel 6.8 Contoh Describe Features Numerik Hasil Scaling**

	swin	stcpb	smean	sloss	dload
count	1.402720e+05	1.402720e+05	1.402720e+05	1.402720e+05	1.402720e+05
mean	-4.391757e-17	4.936294e-17	-3.596479e-17	3.115260e-18	9.852327e-18
std	1.000004e+00	1.000004e+00	1.000004e+00	1.000004e+00	1.000004e+00
min	-8.760629e-01	-6.906500e-01	-5.271400e-01	-7.440542e-02	-2.707735e-01
max	1.141513e+00	2.520026e+00	6.834261e+00	7.686091e+01	9.170963e+00

Proses scaling berhasil dilakukan yang ditandai dengan nilai *mean* untuk seluruh fitur mendekati nilai 0 dan *standard deviation* yang mendekati nilai 1.

## 2. Encoding Categorical Variables

Proses *Feature Encoding* dilakukan bertujuan untuk mengubah data kategorikal menjadi format numerik agar dapat digunakan oleh algoritma *machine learning*. Berikut adalah contoh data kategorikal yang dimiliki oleh dataset UNSW

```
Kolom kategorikal Training sebelum encoding:
      proto attack_cat state service
96203    udp    Generic  INT     dns
58960    tcp    Exploits  FIN      -
65069    udp    Generic  INT     dns
64133    unas      DoS    INT      -
111445   udp    Normal   INT      -
```

**Gambar 6.4 Head Kolom Kategorikal**

Terdapat 4 fitur kategorikal yang ada dalam dataset, tetapi untuk proses *feature encoding* ini fitur target 'attack\_cat' yang merupakan kategori

kelas serangan tidak di encode agar nantinya dapat digunakan sebagai target dalam model klasifikasi.

Untuk melakukan *feature encode* digunakan metode *One-Hot Encoding* untuk fitur 'proto', 'state', dan 'service', karena fitur kategorikal tersebut bersifat nominal yang tidak memiliki urutan atau hierarki yang jelas antar kategori. Dengan metode ini tiap kategori dalam fitur akan menjadi kolom biner terpisah, dimana setiap kolom akan merepresentasikan kebenaran kategori dalam setiap baris data, dengan begitu nilai-nilai dalam fitur tersebut dikonversi menjadi 0 atau 1, yang menandakan kategori tersebut ada atau tidak untuk setiap data point. Berikut adalah hasil encode untuk ketiga fitur kategorikalnya.

```

Encoded 'state' Feature (after One-Hot Encoding):
state_ECO state_FIN state_INT state_PAR state_REQ state_RST \
96203      0.0      0.0      1.0      0.0      0.0      0.0
58960      0.0      1.0      0.0      0.0      0.0      0.0
65069      0.0      0.0      1.0      0.0      0.0      0.0
64133      0.0      0.0      1.0      0.0      0.0      0.0
111445     0.0      0.0      1.0      0.0      0.0      0.0

state_no
96203      0.0
58960      0.0
65069      0.0
64133      0.0
111445     0.0

Encoded 'service' Feature (after One-Hot Encoding):
service_dhcp service_dns service_ftp service_ftp-data \
96203        0.0        1.0        0.0        0.0
58960        0.0        0.0        0.0        0.0
65069        0.0        1.0        0.0        0.0
64133        0.0        0.0        0.0        0.0
111445       0.0        0.0        0.0        0.0

service_http service_irc service_pop3 service_radius service_smtp \
96203        0.0        0.0        0.0        0.0        0.0
58960        0.0        0.0        0.0        0.0        0.0
65069        0.0        0.0        0.0        0.0        0.0
64133        0.0        0.0        0.0        0.0        0.0
111445       0.0        0.0        0.0        0.0        0.0

service_snmp service_ssh service_ssl
96203        0.0        0.0        0.0
58960        0.0        0.0        0.0
65069        0.0        0.0        0.0
64133        0.0        0.0        0.0
111445       0.0        0.0        0.0

Encoded 'proto' Feature (after One-Hot Encoding):
proto_a/n proto_aes-sp3-d proto_any proto_argus proto_aris \
96203      0.0      0.0      0.0      0.0      0.0
58960      0.0      0.0      0.0      0.0      0.0
65069      0.0      0.0      0.0      0.0      0.0
64133      0.0      0.0      0.0      0.0      0.0
111445     0.0      0.0      0.0      0.0      0.0

proto_arp proto_ax.25 proto_bbn-rcc proto_bna proto_br-sat-mon \
96203      0.0      0.0      0.0      0.0      0.0
58960      0.0      0.0      0.0      0.0      0.0
65069      0.0      0.0      0.0      0.0      0.0
64133      0.0      0.0      0.0      0.0      0.0
111445     0.0      0.0      0.0      0.0      0.0

... proto_visa proto_vmtp proto_vrrp proto_wb-expak proto_wb-mon \
96203      ...      0.0      0.0      0.0      0.0      0.0
58960      ...      0.0      0.0      0.0      0.0      0.0
65069      ...      0.0      0.0      0.0      0.0      0.0
64133      ...      0.0      0.0      0.0      0.0      0.0
111445     ...      0.0      0.0      0.0      0.0      0.0

proto_wsn proto_xnet proto_xns-idp proto_xtp proto_zero
96203      0.0      0.0      0.0      0.0      0.0
58960      0.0      0.0      0.0      0.0      0.0
65069      0.0      0.0      0.0      0.0      0.0
64133      0.0      0.0      0.0      0.0      0.0
111445     0.0      0.0      0.0      0.0      0.0

[5 rows x 132 columns]

```

**Gambar 6.5 Hasil Feature Encoding**

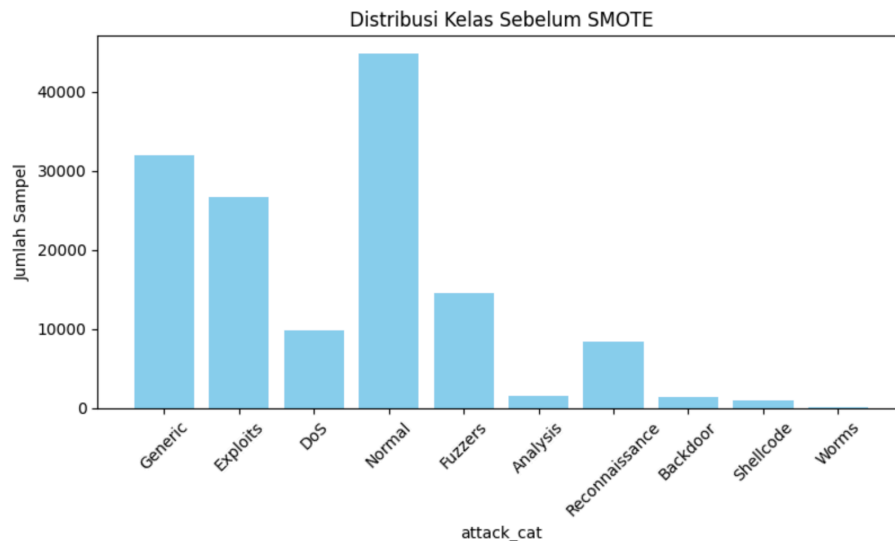
### 3. Handling Imbalanced Classes

Proses *Handling Imbalanced datasets* perlu dilakukan karena dataset yang tidak seimbang dapat mempengaruhi kinerja model *machine learning*. Dalam proses penanganan *imbalanced dataset* digunakan metode SMOTE (*Synthetic Minority Oversampling Technique*), metode ini akan mensintesis sample baru untuk kelas minoritas sehingga distribusi semua kelas menjadi seimbang. Dengan penerapan ini, nantinya model tidak hanya terlatih untuk data yang diulang, melainkan juga pada variasi sintesis yang mendekati data aslinya. Hal ini perlu



dilakukan untuk memastikan model tidak bias terhadap kelas mayoritas sehingga dapat mengklasifikasikan seluruh kelas termasuk minoritasnya.

Sebelum proses SMOTE diterapkan, distribusi kelas pada dataset untuk fitur target sangat tidak seimbang, dimana kelas seperti 'Worms' hanya memiliki sedikit sampel dibandingkan dengan kelas mayoritasnya seperti 'Generic' dan 'Normal', hal tersebut ditunjukkan dari gambar 6.6 berikut.

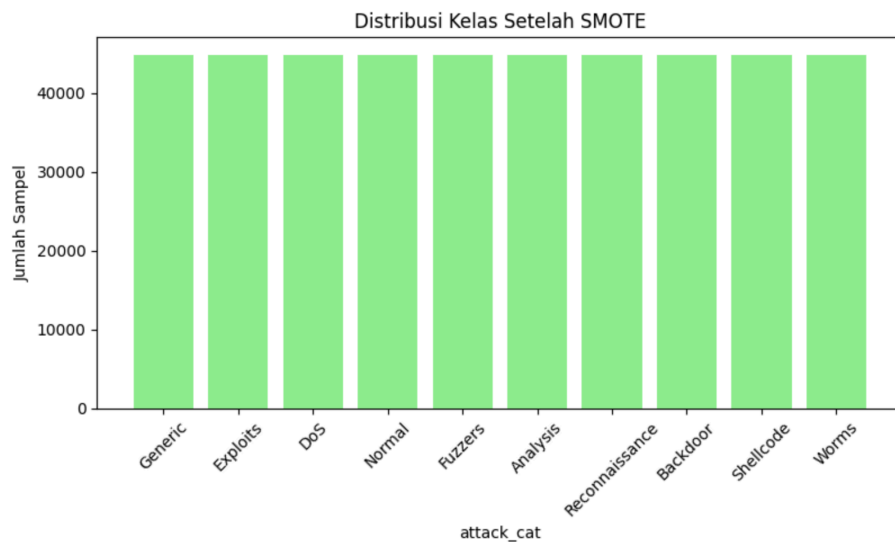


Distribusi sebelum SMOTE:

```
Generic: 31945
Exploits: 26694
DoS: 9832
Normal: 44812
Fuzzers: 14540
Analysis: 1586
Reconnaissance: 8434
Backdoor: 1387
Shellcode: 938
Worms: 104
```

**Gambar 6.6 Data Distribusi Kelas Sebelum *Handling Imbalanced***

Setelah proses dilakukan, dataset untuk fitur target memiliki distribusi kelas 'attack\_cat' yang seimbang ditunjukkan dengan hasil yang tertera pada gambar 6.7.



Distribusi setelah SMOTE:

Generic: 44812  
Exploits: 44812  
DoS: 44812  
Normal: 44812  
Fuzzers: 44812  
Analysis: 44812  
Reconnaissance: 44812  
Backdoor: 44812  
Shellcode: 44812  
Worms: 44812

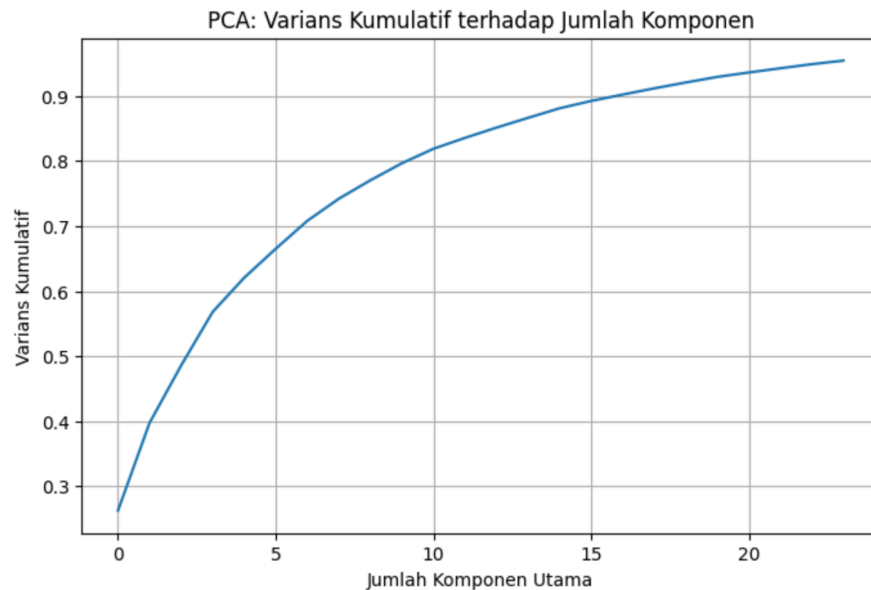
**Gambar 6.7 Data Distribusi Kelas Setelah *Handling Imbalanced***

#### 4. Dimensionality Reduction

Proses *dimensionality reduction* diperlukan untuk mengurangi jumlah fitur pada dataset dengan mempertahankan sebanyak mungkin informasi pentingnya, dengan melakukan proses ini akan menyederhanakan data sehingga lebih mudah di analisis dan meningkatkan performa model *machine learning*. Pada proses ini digunakan *Principal Component Analysis* (PCA) dimana akan mengubah fitur-fitur asli yang mungkin saling berkorelasi menjadi sejumlah komponen utama yang tidak berkorelasi sehingga menghasilkan representasi data yang lebih ringkas.

Dalam implementasinya PCA digunakan untuk mempertahankan 95% varians data aslinya, yang memastikan bahwa informasi mayoritas tetap terjaga walaupun jumlah fitur berkurang. Berikut adalah hasil proses *dimensionality reduction* yang telah dilakukan.

Jumlah fitur sebelum PCA: 189  
Jumlah fitur setelah PCA: 24



**Gambar 6.8 Hasil PCA**

## 7. Penjelasan *Data Pipeline*

Berikut adalah implementasi dari data pipeline menggunakan Pipeline dari pustaka imblearn yang mendukung penggunaan SMOTE pada Pipeline.

```
class FeatureEngineering(BaseEstimator, TransformerMixin):  
    def __init__(self):  
        self.to_drop = []  
  
    def fit(self, X, y=None):  
        dataframe = pd.DataFrame(X)  
  
        numerical_features =  
dataframe.select_dtypes(include=['number']).columns.tolist()  
  
        # Compute the correlation matrix  
corr_matrix = dataframe[numerical_features].corr().abs()  
  
        # Mask the upper triangle of the correlation matrix  
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))  
tri_df = corr_matrix.mask(mask)
```

```
        # Identify highly correlated features (r > 0.95)
        self.to_drop = [c for c in tri_df.columns if any(tri_df[c] >
0.95)]

        return self

    def transform(self, X):
        dataframe = pd.DataFrame(X)

        # Drop the highly correlated features
        drop_highly_correlated = dataframe.drop(columns=self.to_drop)

        return drop_highly_correlated

class OutlierHandler(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        dataframe = pd.DataFrame(X)

        numerical_features =
dataframe.select_dtypes(include=['number']).columns.tolist()

        # Calculate IQR for numerical features
        Q1 = dataframe[numerical_features].quantile(0.25)
        Q3 = dataframe[numerical_features].quantile(0.75)
        IQR = Q3 - Q1

        # Define lower and upper bounds for outliers
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Apply clipping to outliers
        for column in numerical_features:
            dataframe[column] =
dataframe[column].clip(lower=lower_bound[column],
upper=upper_bound[column])

        return dataframe
```

```
numerical_pipeline = ImbPipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler()),
])

onehot_pipeline = ImbPipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot_encoder', OneHotEncoder(handle_unknown='ignore',
sparse_output=False))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_pipeline, numerical_features),
        ('onehot', onehot_pipeline, categorical_features)
    ]
)

pipeline = ImbPipeline([
    ('preprocessor', preprocessor),
    ("feature_engineering", FeatureEngineering()), # Remove highly
correlated features
    ("outlier_handler", OutlierHandler()), # Handle outliers using
IQR
    ("smote", SMOTE(random_state=42)),
    ("pca", PCA(n_components=0.95)),
    ('final_transformer', None)
])
```

*Pipeline* tersebut terdiri dari enam komponen, yakni *preprocessor* untuk *scaling* dan *impute* data kosong, *feature engineering* untuk membuang fitur yang memiliki korelasi sangat tinggi, *outlier handler* untuk membuang data pencilan dengan metode IQR, *SMOTE* untuk melakukan *oversampling*, *PCA* untuk mengurangi dimensi data yang berlebih akibat *one-hot encoding*, dan *final transformer* yang tidak melakukan apa-apa (bila tidak dimasukan, maka imblearn akan *error* karena menganggap *PCA* tidak memiliki *transform*).

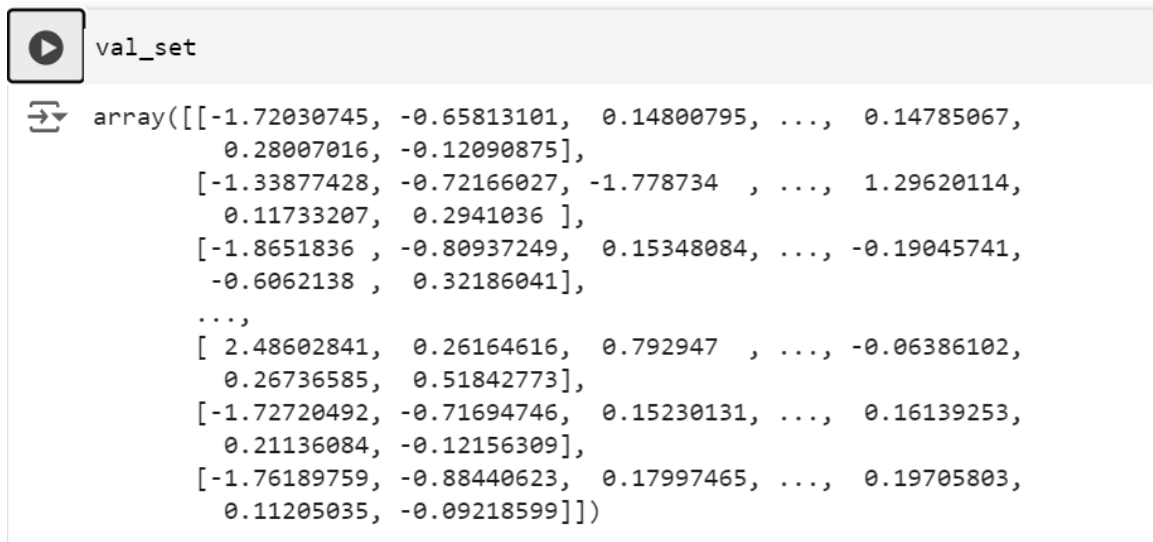
Contoh penggunaan data pipelinenya adalah sebagai berikut.

```
x = df_train.drop(['attack_cat', 'id', 'label'], axis=1)
y = df_train['attack_cat']

x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.2,
random_state=42)

pipeline.fit(x_train, y_train)
train_set = pipeline.transform(x_train)
val_set = pipeline.transform(x_val)
```

Output-nya akan menunjukkan hasil seperti berikut.



```
val_set
array([[ -1.72030745, -0.65813101,  0.14800795, ...,  0.14785067,
         0.28007016, -0.12090875],
       [-1.33877428, -0.72166027, -1.778734 , ...,  1.29620114,
         0.11733207,  0.2941036 ],
       [-1.8651836 , -0.80937249,  0.15348084, ..., -0.19045741,
        -0.6062138 ,  0.32186041],
       ...,
       [ 2.48602841,  0.26164616,  0.792947 , ..., -0.06386102,
         0.26736585,  0.51842773],
       [-1.72720492, -0.71694746,  0.15230131, ...,  0.16139253,
         0.21136084, -0.12156309],
       [-1.76189759, -0.88440623,  0.17997465, ...,  0.19705803,
         0.11205035, -0.09218599]])
```

Gambar 7.1 Output Data Pipeline

## 8. Perbandingan Algoritma Implementasi Mandiri dengan Pustaka

### SciKit-Learn

#### a. K-Nearest Neighbor

Hasil prediksi algoritma KNN yang diimplementasikan from scratch dibandingkan dengan algoritma pustaka Sci-Kit Learn menghasilkan hasil yang sama persis. Akurasi yang didapatkan sebesar 0.318 saat dilakukan pengujian. Namun, perlu dicatat bahwa *benchmark* dilakukan dengan data *dummy* karena keterbatasan memori, tetapi hasilnya harusnya tetap sama.

### b. Gaussian Naive Bayes

Hasil prediksi algoritma implementasi mandiri dan algoritma pustaka Sci-Kit Learn menghasilkan **hasil yang sama persis**. **Akurasi yang didapatkan sebesar 0.689936960074714** atau sekitar 68,9% ketika diuji terhadap data validasi (hasil *split*). Nilai presisi, *recall*, dan skor F1 untuk tiap kelas juga sama pada kedua algoritma. **Sementara itu, pada Kaggle paling baik mendapatkan skor publik sebesar 0.31059 atau 31,1%**. Hal ini kemungkinan disebabkan oleh algoritma Gaussian Naive Bayes yang relatif simpel (mengandalkan fungsi kepadatan distribusi), algoritma sisanya relatif trivial sehingga tidak akan berbeda jauh hasilnya. **Satu-satunya perbedaan yang dapat diamati hanya kecepatan pemrosesan**, baik pada tahap pelatihan maupun inferensi. Algoritma dari pustaka Sci-Kit Learn jauh lebih cepat. Hal tersebut kemungkinan disebabkan oleh algoritma pustaka tersebut yang jauh lebih teroptimasi. Tidak ada pula *hyperparameter* yang bisa dioptimasi pada algoritma ini (kecuali *smoothing*).

### c. ID3

## 9. Kontribusi Anggota dalam Kelompok

Pengerjaan tugas besar 2 IF3170 Inteligensi Artifisial ini dikerjakan oleh lima orang dengan pembagian tugas sebagai berikut.

**Tabel 9.1 Pembagian Tugas Anggota Kelompok**

NIM	Nama	Kontribusi
13221011	Jazila Faza Aliyya Nurfauzi	Pengerjaan Data Cleaning & Preprocessing
		Pengerjaan laporan bagian Data Cleaning & Preprocessing
13221055	Ahmad Hafiz Aliim	Implementasi algoritma ID3
		Pengerjaan laporan bagian algoritma ID3
13221065	Caitleen Devina	Pengerjaan Data Cleaning & Preprocessing
		Pengerjaan laporan bagian Data Cleaning & Preprocessing
18221130	Rayhan Maheswara Pramanda	Implementasi algoritma

		Gaussian Naive Bayes
		Pengerjaan laporan bagian algoritma Gaussian Naive Bayes
		Membantu <i>refactoring</i> penyatuan <i>pipeline</i> akhir
18321008	Jasmine Callista Aurellie Irfan	Implementasi algoritma K-Nearest Neighbor
		Pengerjaan laporan bagian algoritma K-Nearest Neighbor



## 10. Referensi

1. Sakshi Raheja, "Train-Test-Validation Split in 2025",  
<https://www.analyticsvidhya.com/blog/2023/11/train-test-validation-split/#:~:text=the%20overall%20dataset.-,How%20to%20Split%20Train%2DTest%20%3F,distribution%20of%20classes%20or%20outcomes>
2. Alamin Musa Magaga, "Identifying, Cleaning and replacing outliers ! Titanic Dataset",  
<https://www.google.com/url?q=https%3A%2F%2Fmedium.com%2Fanalytics-vidhya%2Fidentifying-cleaning-and-replacing-outliers-titanic-dataset-20182a062893>
3. Deepika Singh, "Cleaning up Data from Outliers",  
<https://www.google.com/url?q=https%3A%2F%2Fwww.pluralsight.com%2Fresources%2Fblog%2Fguides%2Fcleaning-up-data-from-outliers>
4. Alvira Swalin, "How to Make Your Machine Learning Models Robust to Outliers",  
<https://www.google.com/url?q=https%3A%2F%2Fwww.kdnuggets.com%2F2018%2F08%2Fmake-machine-learning-models-robust-outliers.html>
5. Akash Dey, "How to handle Outliers",  
<https://www.google.com/url?q=https%3A%2F%2Fwww.kaggle.com%2Fcode%2Faimack%2Fhow-to-handle-outliers>
6. "2020-07-08-02-Feature-selection-I-selecting-for-feature-information.ipynb",  
[https://colab.research.google.com/github/goodboychan/chans\\_jupyter/blob/main/\\_notebooks/2020-07-08-02-Feature-selection-I-selecting-for-feature-information.ipynb#scrollTo=XF2qJxHsBY5a](https://colab.research.google.com/github/goodboychan/chans_jupyter/blob/main/_notebooks/2020-07-08-02-Feature-selection-I-selecting-for-feature-information.ipynb#scrollTo=XF2qJxHsBY5a)
7. Ilham Nur Hermawan, "Belajar Data Mining: Preprocessing Data dengan Google Colab",  
<https://www.google.com/url?q=https%3A%2F%2Fhamhrmwn.medium.com%2Fbelajar-data-mining-preprocessing-data-dengan-google-colab-8f056a6f0e5b>
8. Saturn Cloud, "Binning a Column with Python Pandas",  
<https://www.google.com/url?q=https%3A%2F%2Fsaturncloud.io%2Fblog%2Fbinning-a-column-with-python-pandas%2F%23%3A%7E%3Atext%3DBinning%2520is%2520the%2520process%2520of%2C30%25E2%2580%259D%252C%2520and%2520so%2520on>
9. R. Kukuh, "Melakukan Feature Scaling pada Dataset",  
<https://www.google.com/url?q=https%3A%2F%2Fmedium.com%2F%40rkukuh%2Fmelakukan-feature-scaling-pada-dataset-229531bb08de>
10. Imbalanced Learn, "SMOTE",  
<https://www.google.com/url?q=https%3A%2F%2Fimbalanced-learn.org%2>

[Fstable%2Fpreferences%2Fgenerated%2Fimblearn.over\\_sampling.SMOTE.html%23imblearn.over\\_sampling.SMOTE](#)

11. Eda Kavlakoglu, "Reducing dimensionality with principal component analysis with Python",

<https://www.google.com/url?q=https%3A%2F%2Fdeveloper.ibm.com%2Ftutorials%2Fawb-reducing-dimensionality-with-principal-component-analysis%2F>