

**PEMBELAJARAN MESIN (H)  
FINAL PROJECT**



**Dosen Pengampu:**

Aldinata Rizky Revanda, S.Kom., M.Kom.

**Oleh :**

Haliza Nur Kamila Apalwan / 5025231038

Stefanus Yosua Mamamoba / 5025231066

Sharfina Ardhiyanti Anam / 5025231111

Rafael Jonathan Raja Nicholas Harianja / 5025231252

Juang Maulana Taruna Putra / 5025231257

Reihan Arianza / 5025231274

## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>2</b>
<b>BAB I PENDAHULUAN.....</b>	<b>3</b>
1.1 Latar Belakang.....	3
1.2 Rumusan Masalah.....	4
1.3 Tujuan.....	4
1.4 Manfaat.....	4
1.5 Batasan Masalah.....	5
<b>BAB II METODOLOGI.....</b>	<b>5</b>
2.1 Metode Uji.....	5
2.1.1 Model Machine Learning.....	5
2.1.1.1 Random Forest Classifier.....	5
2.1.1.2 Support Vector Machine.....	5
2.1.1.3 CatBoost Classifier.....	6
2.1.1.4 Gradient Boosting Classifier.....	6
2.1.1.5 K-Nearest Neighbor.....	6
2.1.1.6 Logistic Regression.....	6
2.1.2 Teknik Pre-processing.....	7
2.1.2.1 SMOTE.....	7
2.1.2.2 Standard Scaler.....	7
2.1.2.3 Feature Selection.....	7
2.1.3 Teknik Evaluasi.....	7
2.1.3.1 Accuracy.....	7
2.1.3.2 Precision.....	8
2.1.3.3 Recall.....	8
2.1.3.4 F1-Score.....	8
2.2 Dataset.....	8
2.3 Desain Sistem.....	15
2.3.1 Tahap EDA.....	15
2.3.2 Tahap Labeling.....	16
2.3.3 Tahap Pembuatan Model.....	16
2.3.4 Skenario Pengujian 1: Penanganan Ketidakseimbangan Data (SMOTE).....	18
2.3.5 Skenario Pengujian 2: Perbandingan Data dengan Standarisasi vs. tanpa Standarisasi.....	18
2.3.6 Skenario Pengujian 3: Perbandingan Data dengan Feature Selection vs. tanpa Feature Selection.....	19
2.3.7 Pemilihan Model Terbaik Antar Model.....	19
<b>BAB III HASIL DAN PEMBAHASAN.....</b>	<b>20</b>
3.1 Random Forest.....	20
3.1.1 Skenario 1: Imbalance vs. SMOTE.....	20
3.1.2 Skenario 2: Normalisasi dengan Imbalance.....	21
3.1.3 Skenario 3: Selection Fitur dengan Imbalance.....	22
3.1.4 Kesimpulan.....	23
3.2 Logistic Regression.....	24

3.2.1 Skenario 1: Imbalance vs. SMOTE.....	24
3.2.2 Skenario 2: Normalisasi dengan SMOTE.....	26
3.2.3 Skenario 3: Selection Fitur dan Normalisasi dengan SMOTE.....	27
3.2.4 Kesimpulan.....	28
3.3 Gradien Boost.....	29
3.3.1 Skenario 1: Imbalance vs. SMOTE.....	29
3.3.2 Skenario 2: Normalisasi dengan Imbalance.....	31
3.3.3 Skenario 3: Imbalance dengan Selection Fitur.....	32
3.3.4 Kesimpulan.....	32
3.4 CatBoost.....	33
3.4.1 Skenario 1: Imbalance vs. SMOTE.....	33
3.4.2 Skenario 2: Normalisasi dengan Imbalance.....	34
3.4.3 Skenario 3: Selection Fitur dan Normalisasi dengan Imbalance.....	35
3.4.4 Skenario 3: Selection Fitur dengan Imbalance.....	37
3.4.5 Kesimpulan.....	37
3.5 SVM.....	37
3.5.1 Skenario 1: Imbalance vs. SMOTE.....	37
3.5.2 Skenario 2: Normalisasi dengan SMOTE.....	39
3.5.3 Skenario 3: Selection Fitur dan Normalisasi dengan SMOTE.....	40
3.5.4 Kesimpulan.....	41
3.6 KNN.....	42
3.6.1 Skenario 1: Imbalance vs. SMOTE.....	42
3.6.2 Skenario 2: Normalisasi dengan SMOTE.....	44
3.6.3 Skenario 3: Selection Fitur dan Normalisasi dengan SMOTE.....	45
3.6.4 Kesimpulan.....	46
<b>BAB IV KESIMPULAN DAN SARAN.....</b>	<b>51</b>
<b>DAFTAR PUSTAKA.....</b>	<b>52</b>

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Perkembangan teknologi informasi yang pesat telah mendorong pertumbuhan jumlah pengguna internet secara global. Menurut laporan terkini, hingga awal tahun 2025, terdapat sekitar 5,56 miliar pengguna internet di seluruh dunia, meningkat sebesar 136 juta pengguna dibanding tahun sebelumnya. Meningkatnya jumlah pengguna ini berdampak langsung pada volume lalu lintas jaringan yang semakin padat, yang kemudian membuka celah bagi potensi ancaman siber dan aktivitas jaringan yang mencurigakan.

Sistem keamanan jaringan seperti firewall merupakan salah satu mekanisme perlindungan dasar yang umum digunakan. Namun, firewall memiliki keterbatasan dalam mendeteksi serangan yang bersumber dari dalam jaringan itu sendiri. Oleh karena itu, dikembangkanlah Intrusion Detection System (IDS), yaitu sistem yang dirancang untuk mendeteksi aktivitas tidak sah atau mencurigakan dalam jaringan komputer. IDS bekerja melalui beberapa pendekatan, seperti signature-based detection, anomaly-based detection, dan hybrid detection.

Sayangnya, pendekatan konvensional pada IDS memiliki kelemahan seperti rendahnya akurasi dalam mendeteksi serangan baru (*zero-day attacks*) serta tingginya *false positive rate*, yang dapat mengganggu proses monitoring jaringan. Untuk mengatasi keterbatasan tersebut, pendekatan berbasis *Machine Learning* (ML) mulai banyak digunakan. Dengan kemampuan ML dalam mempelajari pola dari data historis dan mengenali anomali, IDS berbasis ML mampu memberikan deteksi yang lebih adaptif dan akurat.

Terdapat berbagai metode yang tersedia dalam machine learning untuk diterapkan pada kasus klasifikasi serangan jaringan, seperti *CatBoost*, *Random Forest*, *Gradient Boosting*, *K-Nearest Neighbors (KNN)*, *Support Vector Machine (SVM)*, dan *Logistic Regression*. Setiap metode memiliki karakteristik, keunggulan, dan kelemahannya masing-masing dalam mengolah data IDS, baik dari segi akurasi, efisiensi komputasi, hingga interpretabilitas. Oleh karena itu, penting untuk melakukan analisis perbandingan antar model, termasuk dalam konteks pengaruh normalisasi data, seleksi fitur, serta kontribusi model terhadap peningkatan performa deteksi serangan.

### 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana performa masing-masing metode klasifikasi seperti *CatBoost*, *Random Forest*, *Gradient Boosting*, KNN, SVM, dan *Logistic Regression* dalam mendeteksi serangan jaringan berdasarkan dataset CICIDS2018?

2. Seberapa besar pengaruh *preprocessing* seperti normalisasi dan seleksi fitur terhadap performa dari masing-masing model klasifikasi?
3. Apa metode klasifikasi yang sesuai dalam klasifikasi serangan pada dataset IDS?

### 1.3 Tujuan

Tujuan dari penelitian sebagai berikut:

1. Mengimplementasikan dan membandingkan performa beberapa model klasifikasi serangan jaringan yakni *CatBoost*, *Random Forest*, *Gradient Boosting*, KNN, SVM, dan *Logistic Regresion* menggunakan dataset CICIDS2018.
2. Menganalisis pengaruh normalisasi data dan seleksi fitur terhadap efektivitas model dalam melakukan klasifikasi.
3. Mengidentifikasi metode klasifikasi yang paling efektif dan efisien untuk diterapkan dalam kasus deteksi serangan pada dataset IDS.

### 1.4 Manfaat

Beberapa manfaat yang dapat diperoleh dari penelitian ini antara lain:

1. Bagi peneliti:
  - a. Memberikan pengalaman langsung dalam implementasi dan evaluasi model *machine learning* pada kasus nyata di bidang keamanan jaringan.
  - b. Meningkatkan pemahaman mengenai peran *preprocessing* data seperti normalisasi dan seleksi fitur dalam sistem deteksi serangan.
2. Bagi pihak lain:
  - a. Memberikan referensi bagi pengembangan IDS berbasis *machine learning* dengan pendekatan komparatif model.
  - b. Dapat dijadikan acuan dalam penelitian selanjutnya yang berkaitan dengan optimasi performa model klasifikasi untuk deteksi intrusi.

### 1.5 Batasan Masalah

Penelitian ini menetapkan batasan ruang lingkup tertentu agar analisis tetap sesuai dengan tujuan yang telah ditetapkan:

1. Dataset yang digunakan adalah CICIDS2018 dan difokuskan pada subset data tertentu yang mencakup berbagai jenis serangan dan trafik normal
2. Penelitian hanya mengimplementasikan model klasifikasi *supervised learning* seperti *CatBoost*, *Random Forest*, *Gradient Boosting*, KNN, SVM, dan *Logistic Regresion* tanpa melakukan percobaan terhadap metode lain yang memungkinkan.

## BAB II METODOLOGI

### 2.1 Metode Uji

#### 2.1.1 Model *Machine Learning*

##### 2.1.1.1 *Random Forest Classifier*

*Random Forest* merupakan suatu algoritma machine learning yang menggunakan banyak decision tree selama proses training. *Random Forest* mula-mula membuat beberapa sampel acak dari subset. Setelah itu dilakukan klasifikasi atau regresi dengan mengambil rata-rata dari seluruh pohon. Algoritma ini mampu bertahan dari overfitting karena setiap pohon yang terbentuk terdiri dari subset yang berbeda-beda. Namun metode ini harus menciptakan banyak pohon yang membuat program menjadi lebih kompleks dan butuh sumber daya [3].

##### 2.1.1.2 *Support Vector Machine*

*Support Vector Machine* (SVM) telah dikembangkan dan berhasil diaplikasikan untuk memprediksi seri waktu, pengenalan wajah, maupun pemrosesan data biologis untuk diagnosis medis. Ide utama dari SVM adalah menggunakan *hyperplane* atau garis optimal untuk memisahkan kelas-kelas dari data. Algoritma ini dirancang dalam kerangka *statistical learning theory* dengan menemukan fungsi yang meminimalkan kesalahan prediksi. Algoritma ini bekerja dengan dua prinsip, yaitu *empirical risk minimization (ERM)* dan *structural risk minimization (SRM)*. ERM bertujuan untuk meminimalkan kesalahan pada pelatihan, sedangkan SRM bertujuan untuk memilih model dengan kompleksitas terbaik [4].

##### 2.1.1.3 *CatBoost Classifier*

*CatBoost* merupakan modifikasi dari algoritma *gradient boosting* yang sudah ada yang menghindari kebocoran target. Metode ini memiliki fitur utama, yaitu *ordered boosting* yang lebih tahan pada pergeseran prediksi. Selain itu, metode ini juga mampu untuk mengolah fitur kategorikal secara efisien. menggunakan *ordered target encoding* [2]. Hal ini membuat *CatBoost* lebih hemat dalam penggunaan memori dibandingkan *one-hot encoding* dan meningkatkan akurasi model. Keunggulan ini memungkinkan *CatBoost* untuk memberikan performa prediksi yang lebih *robust* dan akurat pada dataset dengan fitur kategorikal yang kompleks.

##### 2.1.1.4 *Gradient Boosting Classifier*

*Gradient Boosting Machine* (GBM) adalah metode *ensemble* untuk masalah regresi dan klasifikasi di pembelajaran mesin. *Gradient Boosting Classifier* (GBC) adalah kasus khusus pada GBM yang didesain untuk klasifikasi. Metode ini mengonstruksi pohon dan memprediksi residu dari prediksi yang sebelumnya [1]. Tidak seperti GBM yang digunakan untuk regresi, GBC melibatkan prediksi untuk hasil biner. Dengan demikian setiap *terminal node* pada tree mengeluarkan nilai yang ditentukan oleh kesalahan residu dan probabilitas dari iterasi sebelumnya.

#### 2.1.1.5 *K-Nearest Neighbor*

KNN merupakan salah satu algoritma pembelajaran mesin yang mudah untuk dipahami. Algoritma ini digunakan untuk berbagai macam masalah, seperti klasifikasi gambar, penelusuran kejadian, dan lain-lain. Ide dari metode ini adalah mengklasifikasi objek yang masih asing dengan objek tetangganya pada dataset [5]. Berbagai macam modifikasi dilakukan untuk mengembangkan algoritma ini, seperti penggunaan beban, sumber daya komputasi, dan lain-lain. KNN banyak digunakan karena sifatnya yang non-parametrik dan fleksibel dalam menangani data berdimensi tinggi maupun rendah.

#### 2.1.1.6 *Logistic Regression*

Algoritma ini menerapkan analisis jaringan otak yang mengklasifikasikan keterhubungan antar edge. Logistic Regression sering digunakan karena tidak bergantung pada *p-value*, sehingga lebih fleksibel. Kelebihan dalam konteks *brain imaging* adalah tidak perlu melakukan preseleksi fitur secara manual, menghilangkan ketergantungan pada distribusi klasik dan *p-value* [6]. Model ini mengasumsikan bahwa keberadaan suatu edge mengikuti distribusi Bernoulli, dan menggunakan fungsi logit untuk memetakan kombinasi linier fitur ke dalam rentang probabilitas. Estimasi parameter dilakukan dengan metode *Maximum Likelihood Estimation* (MLE), yang dapat diaplikasikan pada seluruh pasangan wilayah otak secara efisien.

### 2.1.2 Teknik *Pre-processing*

#### 2.1.2.1 *SMOTE*

*Synthetic Minority Over-sampling Technique* (SMOTE) adalah suatu teknik pada pembelajaran mesin yang melakukan *over-sampling* pada kelas minoritas dan *under-sampling* untuk pada kelas mayoritas pada dataset. Proses *over-sampling* dilakukan dengan membentuk kelas data minoritas sintesis. Hal ini bertujuan untuk memberi kemudahan untuk mengidentifikasi secara spesifik fitur sebagai wilayah pemilihan untuk kelas minoritas. Algoritma ini menggunakan *K-Nearest Neighbor* untuk memilih kelas minoritasnya dengan mencari perbedaan dari sampel atau fitur vektor yang dipertimbangkan dan mencari tetangganya, kemudian mengalikan perbedaan dengan nol atau satu, lalu menyimpannya pada vektor fitur yang dipertimbangkan. Pendekatan ini secara efektif membuat kelas minoritas menjadi lebih general [7].

#### 2.1.2.2 *Standard Scaler*

Terdapat bermacam-macam teknik penskalaan, salah satunya adalah *Standard Scaler*. Metode ini menggunakan nilai Z. Menghitung Z atau mencari distribusi normalnya dapat dilakukan dengan mengurangi nilai  $x$  dengan rata-rata nilai  $x$ , kemudian membaginya dengan standar deviasinya. Kelebihan dari pendekatan ini adalah kemampuannya untuk mengubah nilai positif dan negatif dari suatu atribut ke dalam distribusi [8]. Biasanya teknik ini cocok untuk algoritma berbasis gradien atau jarak, seperti *Logistic Regression*, *SVM*, dan *MLP* untuk menyamakan jarak dan lebih

akurat [9]. Hasil akhir dari perubahan tersebut adalah data memiliki rata-rata nol dan standar deviasi satu.

### 2.1.2.3 Feature Selection

One-Way-ANOVA adalah metode yang digunakan untuk menyeleksi fitur yang digunakan untuk analisis data dan menarik informasi penting dari *P-value*. Teknik ini kokoh karena setiap sampel diasumsikan memiliki nilai varians yang sama dan independen [10]. Tujuan dari penggunaan fitur ini untuk melihat keterhubungan antar kelas. Semakin besar nilai dari *F-score* maka hubungannya semakin erat dan sebaliknya. Metode ini cocok untuk digunakan pada model yang perlu melakukan klasifikasi, namun setiap data kategorikal harus dilakukan proses encoding atau *mapping*.

### 2.1.3 Teknik Evaluasi

#### 2.1.3.1 Accuracy

Akurasi menentukan nilai kebenaran dari suatu pengklasifikasian atau total dari nilai diagonal pada *confusion matrix* [13]. Pendekatan ini cocok untuk digunakan pada data yang seimbang dari jumlah kelas positif maupun yang negatif. Sebaliknya, metode ini kurang sesuai untuk data yang tidak seimbang karena dapat menghasilkan nilai evaluasi yang menyesatkan. Dalam kasus seperti itu, model cenderung hanya mempelajari pola dari kelas mayoritas, sehingga performa terhadap kelas minoritas menjadi buruk. Oleh karena itu, akurasi perlu dilengkapi dengan metrik lain seperti *precision*, *recall*, atau *F1-score* untuk mendapatkan gambaran performa model yang lebih akurat.

#### 2.1.3.2 Precision

*Precision* atau *Confidence* pada data mining merupakan proporsi dari kasus terprediksi positif yang sebenarnya positif. Pendekatan ini lebih umum diketahui sebagai *true positive accuracy* [11]. Biasanya fitur ini digunakan pada kasus klasifikasi spam pada email. Hal ini penting karena kesalahan dalam pengklasifikasian dapat menghilangkan informasi penting. Semakin tinggi nilai presisi, semakin selektif sistem dalam menentukan kebenaran suatu informasi.

#### 2.1.3.3 Recall

*Recall* atau sensitivitas merupakan proporsi kebenaran dari suatu prediksi terhadap kasus positif. Fitur ini menunjukkan seberapa banyak kasus positif yang relevan berhasil terdeteksi oleh model. Metode ini secara umum dikenal sebagai *true positive rate* [11]. Salah satu contoh penerapannya adalah dalam deteksi penyakit kanker, karena lebih baik salah mendiagnosis pasien sehat sebagai sakit dibandingkan tidak mendeteksi pasien yang



benar-benar sakit. Selain itu, dalam kasus keamanan siber, recall digunakan untuk menilai sejauh mana kemampuan model dalam mendeteksi ancaman yang benar-benar ada.

#### 2.1.3.4 *F1-Score*

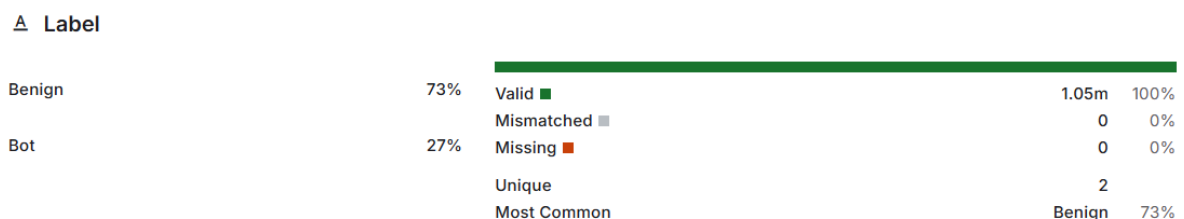
*F-measure* atau yang lebih dikenal dengan *F1-Score*, digunakan dalam berbagai kasus, seperti ekstraksi informasi dari teks. Selain itu, metode ini juga digunakan untuk mengidentifikasi penyebutan entitas yang sama dalam kumpulan dokumen. *F1-Score* sering dimanfaatkan dalam klasifikasi teks, terutama untuk mengoptimalkan sistem yang berhadapan dengan data dinamis seperti berita atau email, di mana pengumpulan label kelas yang benar sangat terbatas atau memerlukan waktu. Yang paling penting, *F1-Score* sangat efektif untuk menangani masalah klasifikasi data yang tidak seimbang, karena mampu memberikan penilaian yang seimbang antara *precision* dan *recall*, sehingga model tidak bias terhadap kelas mayoritas saja [12].

## 2.2 Dataset

Penelitian ini dilakukan secara sistematis untuk mengeksplorasi dan menganalisis data yang diperoleh dari Kaggle, sebuah platform daring yang populer dalam bidang kompetisi data dan sumber ilmu pengetahuan. Dataset yang digunakan berisi informasi mengenai kumpulan data lalu lintas jaringan (*network traffic*) yang telah diberi label untuk membedakan antara aktivitas normal dan aktivitas mencurigakan atau berbahaya (seperti serangan atau intrusi).

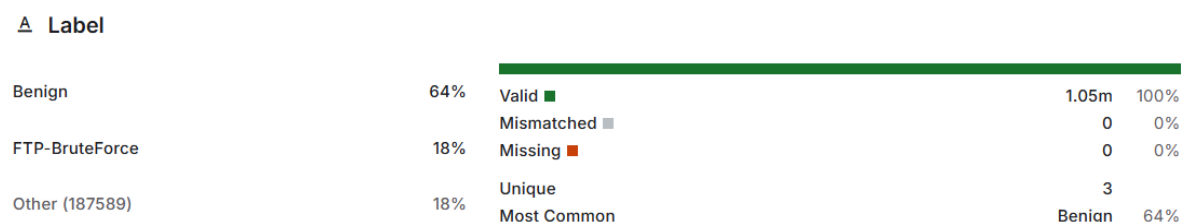
Dataset ini terdiri dari 10 sub-dataset, dengan masing-masing file memiliki 80 kolom fitur dan 2 hingga 3 label jenis serangan. Berikut adalah visualisasi proporsi label yang dimiliki oleh masing-masing sub-dataset:

- Botnet-Friday-02-03-2018.csv



Gambar 2.1 Persentase jumlah tiap label pada Botnet

- Bruteforce-Wednesday-14-02-2018.csv



Gambar 2.2 Persentase jumlah tiap label pada Bruteforce

- DDoS1-Tuesday-20-02-2018.csv

#### Label

DDoS attacks-LOIC-HTTP	55%	Valid	1.05m	100%
		Mismatched	0	0%
Benign	45%	Missing	0	0%
		Unique	2	
		Most Common	DDoS attac...	55%

Gambar 2.3 Persentase jumlah tiap label pada DDoS1

- DDoS2-Wednesday-21-02-2018.csv

#### Label

DDoS attack-HOIC	65%	Valid	1.05m	100%
		Mismatched	0	0%
Benign	34%	Missing	0	0%
		Unique	3	
Other (1730)	0%	Most Common	DDoS attac...	65%

Gambar 2.4 Persentase jumlah tiap label pada DDoS2

- DoS1-Thursday-15-02-2018.csv

#### Label

Benign	95%	Valid	1.05m	100%
		Mismatched	0	0%
DoS attacks-GoldenEye	4%	Missing	0	0%
		Unique	3	
Other (10990)	1%	Most Common	Benign	95%

Gambar 2.5 Persentase jumlah tiap label pada DoS1

- DoS2-Friday-16-02-2018.csv

#### Label

DoS attacks-Hulk	44%	Valid	1.05m	100%
		Mismatched	0	0%
Benign	43%	Missing	0	0%
		Unique	3	
Other (139890)	13%	Most Common	DoS attack...	44%

Gambar 2.6 Persentase jumlah tiap label pada DoS1

- Infil1-Wednesday-28-02-2018.csv

▲ Label

Benign	89%	Valid ■	613k	100%
		Mismatched ■	0	0%
Infiltration	11%	Missing ■	0	0%
		Unique	2	
		Most Common	Benign	89%

Gambar 2.7 Persentase jumlah tiap label pada Infiltration1

- Infil2-Thursday-01-03-2018.csv

▲ Label

Benign	72%	Valid ■	331k	100%
		Mismatched ■	0	0%
Infiltration	28%	Missing ■	0	0%
		Unique	2	
		Most Common	Benign	72%

Gambar 2.8 Persentase jumlah tiap label pada Infiltration2

- Web1-Thursday-22-02-2018.csv

▲ Label

Benign	100%	Valid ■	1.05m	100%
		Mismatched ■	0	0%
Brute Force -Web	0%	Missing ■	0	0%
Other (113)	0%	Unique	4	
		Most Common	Benign	100%

Gambar 2.9 Persentase jumlah tiap label pada Web1

- Web2-Friday-23-02-2018.csv

▲ Label

Benign	100%	Valid ■	1.05m	100%
		Mismatched ■	0	0%
Brute Force -Web	0%	Missing ■	0	0%
Other (204)	0%	Unique	4	
		Most Common	Benign	100%

Gambar 2.10 Persentase jumlah tiap label pada Web2

Setiap sub-dataset terdiri dari fitur-fitur statistik lalu lintas jaringan yang dihasilkan dari aktivitas komunikasi dalam jaringan. Berikut adalah fitur yang terdapat dalam dataset:

Fitur	Penjelasan
<b>Metadata dan Flow-Level Features</b>	
Dst Port	Port tujuan dari koneksi jaringan.
Protocol	Protokol komunikasi yang digunakan, seperti TCP atau UDP.
Timestamp	Waktu saat koneksi dimulai.
Flow Duration	Lama durasi koneksi dalam satuan nanodetik.
<b>Jumlah Paket dan Panjang Data</b>	
Tot Fwd Pkts	Total paket yang dikirim dari sumber ke tujuan.
Tot Bwd Pkts	Total paket yang dikirim dari tujuan ke sumber.
TotLen Fwd Pkts	Total panjang data (dalam byte) dari semua paket ke arah tujuan.
TotLen Bwd Pkts	Total panjang data (dalam byte) dari semua paket dari arah tujuan.
<b>Statistik Panjang Paket</b>	
Fwd Pkt Len Max/Min/Mean/Std	Statistik panjang paket arah forward (maksimum, minimum, rata-rata, dan standar deviasi).
Bwd Pkt Len Max/Min/Mean/Std	Statistik panjang paket arah backward (maksimum, minimum, rata-rata, dan standar deviasi).
<b>Laju dan Interval</b>	
Flow Byts/s	Jumlah byte per detik pada seluruh aliran.
Flow Pkts/s	Jumlah paket per detik pada seluruh aliran.
Flow IAT Mean/Std/Max/Min	Statistik <i>Inter Arrival Time</i> antar paket dalam satu aliran.
Fwd IAT Tot/Mean/Std/Max/Min	Statistik waktu antar kedatangan paket arah forward.
Bwd IAT Tot/Mean/Std/Max/Min	Statistik waktu antar kedatangan paket arah backward.
<b>Flag TCP/IP</b>	

Fwd/Bwd PSH Flags	Jumlah paket arah forward/backward dengan Push flag aktif.
Fwd/Bwd URG Flags	Jumlah paket arah forward/backward dengan Urgent flag aktif.
FIN/SYN/RST/PSH/ACK/ URG Flag Cnt	Jumlah masing-masing flag TCP yang digunakan dalam aliran.
CWE Flag Count	Jumlah penggunaan flag Congestion Window Reduced (CWR).
ECE Flag Cnt	Jumlah penggunaan flag ECN-Echo, untuk kontrol kemacetan.
<b>Header dan Ukuran Segmen</b>	
Fwd/Bwd Header Len	Panjang header paket arah forward dan backward.
Fwd Pkts/s, Bwd Pkts/s	Jumlah paket per detik pada masing-masing arah.
Pkt Len Min/Max/ Mean/Std/Var	Statistik panjang seluruh paket dalam aliran.
Pkt Size Avg	Rata-rata ukuran paket.
Fwd/Bwd Seg Size Avg	Rata-rata ukuran segmen TCP arah forward/backward.
Fwd Seg Size Min	Ukuran minimum segmen TCP arah forward.
<b>Subflow</b>	
Subflow Fwd/Bwd Pkts	Jumlah paket pada subflow arah forward/backward.
Subflow Fwd/Bwd Byts	Jumlah byte pada subflow arah forward/backward.
<b>Window Size dan Data Rate</b>	
Init Fwd/Bwd Win Byts	Ukuran awal window TCP untuk arah forward/backward.
Fwd Byts/b Avg, Bwd Byts/b Avg	Rata-rata byte per blok data pada masing-masing arah.
Fwd/Bwd Blk Rate Avg	Laju rata-rata blok data per arah.
Fwd Pkts/b Avg	Rata-rata paket per blok data arah forward.
<b>Aktivitas dan Waktu Keaktifan</b>	
Fwd Act Data Pkts	Jumlah paket forward yang berisi data aktif.
Active Mean/Std/ Max/Min	Statistik waktu aktif antar periode lalu lintas data.

Label	
Label	Kategori atau kelas dari aliran data, seperti <i>Benign</i> , <i>Botnet</i> , <i>D</i> , <i>Bruteforce</i> , dan lainnya.

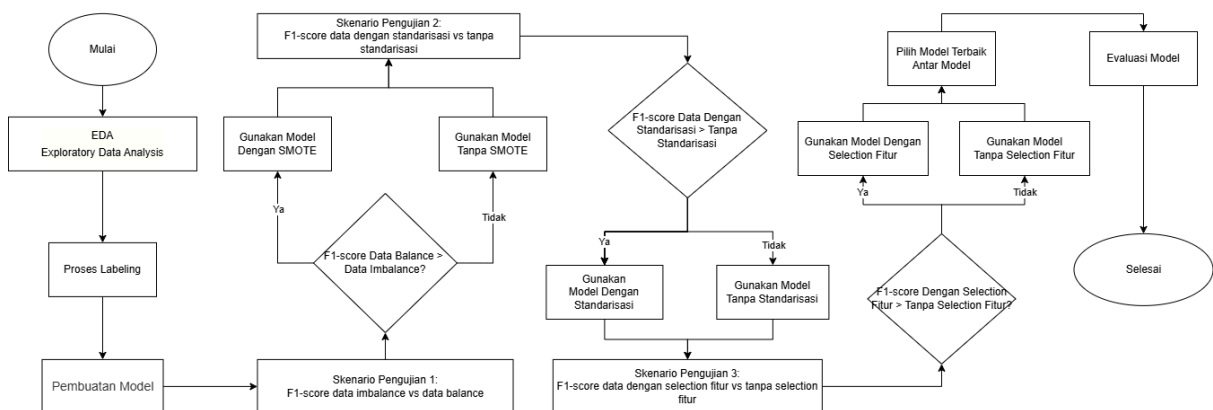
Tabel 2.1 Daftar fitur

Total label yang ditemukan pada dataset ini sebanyak 15 jenis yang merepresentasikan berbagai bentuk lalu lintas jaringan, baik normal maupun berbahaya. Label-label ini digunakan sebagai target dalam proses klasifikasi untuk mendeteksi serangan siber pada jaringan. Adapun daftar lengkap label adalah sebagai berikut:

1. Benign : Menunjukkan lalu lintas jaringan yang normal tanpa indikasi serangan atau aktivitas mencurigakan.
2. Bot : Mengindikasikan adanya aktivitas dari perangkat yang telah terinfeksi dan menjadi bagian dari botnet, yang dikendalikan oleh penyerang untuk menjalankan tugas-tugas otomatis berbahaya.
3. SSH-Bruteforce : Serangan brute force terhadap protokol SSH (Secure Shell) dengan mencoba berbagai kombinasi login secara terus-menerus untuk mendapatkan akses ilegal.
4. FTP-Bruteforce : Serangan brute force yang menargetkan layanan FTP (File Transfer Protocol) untuk mencoba masuk dengan cara menebak username dan password.
5. DDoS attack-HOIC : Serangan Distributed Denial of Service (DDoS) menggunakan tool HOIC (High Orbit Ion Cannon) yang membanjiri server dengan permintaan HTTP untuk melumpuhkan layanan.
6. DDoS attack-LOIC-UDP : Serangan DDoS berbasis protokol UDP menggunakan tool LOIC (Low Orbit Ion Cannon), yang mengirimkan paket dalam jumlah besar ke target.
7. DDoS attacks-LOIC-HTTP : Serangan DDoS serupa namun dengan target protokol HTTP, masih menggunakan tool LOIC.
8. DoS attacks-Hulk : Serangan DoS (Denial of Service) yang menciptakan lalu lintas HTTP dinamis dalam jumlah besar untuk membebani server dan membuatnya tidak dapat merespons permintaan lain.
9. DoS attacks-SlowHTTPTest : Serangan DoS yang mengeksploitasi kelemahan dalam menangani permintaan HTTP lambat, menyebabkan server menggantung dalam waktu lama.

10. DoS attacks-GoldenEye : Serangan DoS terhadap aplikasi web menggunakan permintaan HTTP yang berulang dan berat untuk membebani sumber daya server.
11. DoS attacks-Slowloris : Serangan DoS yang membuka banyak koneksi HTTP parsial dan menjaganya tetap terbuka dalam waktu lama, menghambat koneksi baru.
12. Infiltration : Menunjukkan aktivitas penyusupan ke dalam jaringan dengan tujuan memperoleh akses tanpa terdeteksi untuk mencuri atau merusak data.
13. Brute Force-Web : Serangan brute force terhadap halaman login web untuk menebak kredensial secara otomatis dan berulang.
14. Brute Force-XSS : Gabungan antara serangan brute force dengan injeksi skrip lintas situs (*Cross-Site Scripting*), yang menargetkan celah keamanan pada formulir input web.
15. SQL Injection : Serangan yang menyisipkan perintah SQL ke dalam input aplikasi untuk mengeksploitasi celah dalam sistem basis data, biasanya digunakan untuk membaca atau memanipulasi data tanpa izin.

## 2.3 Desain Sistem



Gambar 2.11 Diagram Alur Desain Sistem

### 2.3.1 Tahap EDA

Tahap EDA bertujuan untuk memahami struktur dan karakteristik awal dari dataset CICIDS2018 sebelum masuk ke proses pelabelan dan pembuatan model. Pada tahap ini, dilakukan beberapa langkah pembersihan data (data cleaning) untuk menangani nilai-nilai yang tidak valid, seperti *infinity* dan *missing values (NaN)*, yang dapat mengganggu proses pelatihan model.

Berikut adalah langkah-langkah yang dilakukan dalam EDA:

- 1) Mengganti Nilai Inf dengan NaN

Kolom Flow Byts/s dan Flow Pkts/s diketahui mengandung nilai inf atau -inf yang muncul akibat pembagian angka dengan nol atau kesalahan perhitungan lainnya. Nilai ini tidak dapat diproses oleh sebagian besar algoritma Machine Learning, sehingga perlu diganti terlebih dahulu dengan nilai NaN.

2) Mengganti Nilai NaN di Flow Byts/s dengan Nilai Median

Untuk menghindari distorsi data, nilai kosong pada kolom Flow Byts/s diisi dengan nilai median dari kolom tersebut. Penggunaan median dipilih karena lebih robust terhadap outlier dibandingkan mean.

3) Mengganti Nilai NaN di Flow Pkts/s dengan Penjumlahan Forward dan Backward

Nilai kosong pada kolom Flow Pkts/s diisi menggunakan pendekatan komputasional dengan menjumlahkan Fwd Pkts/s dan Bwd Pkts/s yang secara logika mewakili total aliran paket per detik.

### 2.3.2 Tahap Labeling

Pada tahap labeling, dilakukan proses transformasi terhadap kolom *Label* dalam dataset CICIDS2018 yang semula berbentuk teks menjadi format numerik. Setiap jenis serangan maupun aktivitas normal pada jaringan diberikan label angka tertentu agar dapat digunakan sebagai target dalam proses klasifikasi. Proses ini dilakukan dengan metode *label encoding* menggunakan pemetaan manual yang disesuaikan dengan kategori serangan yang tersedia dalam dataset. Berikut adalah daftar label numerik yang digunakan dalam penelitian ini:

0 : Bot

1 : Benign (lalu lintas normal atau tidak berbahaya)

2 : SSH-Bruteforce

3 : DDoS attack - HOIC

4 : DDoS attack - LOIC UDP

5 : DoS attacks - Hulk

6 : Infiltration

7 : DoS attacks - GoldenEye

8 : DoS attacks - Slowloris

9 : DDoS attacks - LOIC HTTP

10 : Brute Force - Web

11 : Brute Force - XSS

12 : SQL Injection

13 : DoS attacks - SlowHTTPTest



### 2.3.3 Tahap Pembuatan Model

Pada tahap ini, dilakukan proses pembentukan model klasifikasi untuk mendeteksi jenis serangan berdasarkan fitur-fitur numerik yang tersedia dalam dataset. Tahapan ini melibatkan pemisahan fitur (variabel input) dan label (variabel target), pembagian dataset menjadi data latih dan data uji, serta pelatihan model menggunakan beberapa algoritma klasifikasi.

#### 1) Pemisahan Fitur dan Label

Pertama-tama, data dipisahkan antara fitur (X) dan label (y). Kolom Label digunakan sebagai variabel target klasifikasi, sedangkan kolom Timestamp diabaikan karena tidak memberikan informasi yang relevan dalam proses prediksi. Dengan demikian, fitur yang digunakan merupakan seluruh kolom numerik kecuali kolom *Label* dan *Timestamp*.

```
X = df.drop(columns=['Label', 'Timestamp'])
y = df['Label']
```

#### 2) Pembagian Data Latih dan Uji Coba

Dataset dibagi menjadi dua bagian menggunakan fungsi *train\_test\_split* dari pustaka *scikit-learn*, dengan proporsi 80% sebagai data latih dan 20% sebagai data uji. Parameter *random\_state* disetel untuk menjaga reproduibilitas hasil eksperimen.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

#### 3) Inisialisasi dan Pembuatan Model

Setelah pembagian data, dilakukan inisialisasi dan pelatihan model menggunakan beberapa algoritma klasifikasi sebagai berikut:

##### 1. *Random Forest Classifier*

Menggunakan *RandomForestClassifier* dari pustaka *sklearn.ensemble* untuk membentuk banyak pohon keputusan dan mengambil mayoritas voting sebagai hasil prediksi.

##### 2. *Support Vector Machine (SVM)*

Menggunakan SVC dari *sklearn.svm*, algoritma ini memaksimalkan margin antara kelas-kelas data dengan memanfaatkan *hyperplane* optimal.

##### 3. *CatBoost Classifier*

Menggunakan *CatBoostClassifier* dari pustaka *catboost*, algoritma ini memanfaatkan boosting yang efisien dalam menangani fitur kategorikal dan menghindari kebocoran target.

##### 4. *Gradient Boosting Classifier*

Menggunakan *GradientBoostingClassifier* dari *sklearn.ensemble*, model ini

melatih pohon secara berurutan untuk meminimalkan kesalahan dari prediksi sebelumnya.

5. *K-Nearest Neighbor (KNN)*

Menggunakan *KNeighborsClassifier* dari *sklearn.neighbors*, model ini mengklasifikasikan data baru berdasarkan mayoritas kelas dari k tetangga terdekat.

6. *Logistic Regression*

Menggunakan *LogisticRegression* dari *sklearn.linear\_model*, model ini memprediksi probabilitas dari kelas target menggunakan kombinasi linier fitur dan fungsi logit.

Masing-masing model kemudian dilatih dengan data latih yang telah disiapkan menggunakan metode *.fit()* pada objek model tersebut.

### 2.3.4 Skenario Pengujian 1: Penanganan Ketidakseimbangan Data (SMOTE)

Pada skenario 1, setiap algoritma klasifikasi yang telah diinisialisasi akan dilatih dan diuji dalam dua kondisi data pelatihan: data asli yang belum diseimbangkan, dan data yang telah diseimbangkan menggunakan SMOTE. Dimana penjelasan umum mengenai SMOTE telah dipaparkan pada bab 2.1.2 dan akan dilakukan metodologi percobaan sebagai berikut:

1. Percobaan 1: Pelatihan dengan Data Imbalance

Pada percobaan ini, setiap model akan dilatih menggunakan data pelatihan ( $X_{train}$ ) dan label asli ( $y_{train}$ ) yang masih menunjukkan ketidakseimbangan kelas. Prediksi kemudian dilakukan pada data uji yang asli pula ( $X_{test}$ ).

```
model.fit(X_train, y_train)
y_pred = model_lr.predict(X_test)
```

2. Percobaan 2: Pelatihan dengan Data Balance (SMOTE)

Pada percobaan ini, teknik SMOTE diterapkan untuk menyeimbangkan distribusi kelas. Setelah proses SMOTE dilakukan, model akan dilatih pada data yang telah diseimbangkan ini ( $X_{train\_resampled}$ ,  $y_{train\_resampled}$ ).

```
# SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled =
smote.fit_resample(X_train, y_train)
```

Untuk setiap algoritma klasifikasi, F1-score (*weighted average*) dari kedua percobaan (dengan dan tanpa SMOTE) akan dibandingkan. Versi model yang menghasilkan F1-score

lebih tinggi akan dipilih sebagai model terbaik untuk algoritma tersebut pada Skenario 1 dan diteruskan sebagai input pada Skenario Pengujian 2.

### 2.3.5 Skenario Pengujian 2: Perbandingan Data dengan Standarisasi vs. tanpa Standarisasi

Skenario pengujian 2 bertujuan untuk mengevaluasi implementasi standarisasi fitur terhadap performa model dengan memanfaatkan model terbaik yang telah dilakukan pada Skenario 1. Penjelasan umum mengenai *Standard Scaler* telah dipaparkan pada bab 2.1.2. Setiap model akan dilatih menggunakan data pelatihan ( $X_{train\_norm}$ ) dan diprediksi pada data uji ( $X_{test\_norm}$ ) yang telah dilakukan standarisasi, dimana input dari standarisasi ini disesuaikan dengan hasil data dari Skenario 1.

```
scaler = StandardScaler()
X_train_norm = scaler.fit_transform(X_train_s1)
X_test_norm = scaler.transform(X_test_s1)
```

F1-score (*weighted average*) dari kedua percobaan (dengan dan tanpa standarisasi) akan dibandingkan. Versi model yang menghasilkan F1-score lebih tinggi akan dipilih sebagai model terbaik untuk algoritma tersebut pada Skenario 2 dan diteruskan sebagai input pada Skenario Pengujian 3.

### 2.3.6 Skenario Pengujian 3: Perbandingan Data dengan Feature Selection vs. tanpa Feature Selection

Skenario pengujian terakhir ini bertujuan untuk mengoptimalkan jumlah fitur yang digunakan oleh model. Penjelasan umum mengenai Feature Selection (One-Way-ANOVA) telah dipaparkan pada bab 2.1.2. Model terbaik dari Skenario 2 akan dilatih menggunakan data pelatihan yang telah melalui proses pemilihan fitur *SelectKBest*.

```
k_param = [10, 20, 30, 40, 50, 60, 70, 80]

for ks in k_param:
    selector = SelectKBest(score_func=f_classif, k=ks)
    X_train_selected = selector.fit_transform(X_train, y_train)
    X_test_selected = selector.transform(X_test)
```

F1-score (*weighted average*) dari kedua percobaan (dengan dan tanpa selection feature) akan dibandingkan. Versi model yang menghasilkan F1-score lebih tinggi akan dipilih sebagai model terbaik untuk algoritma tersebut pada Skenario 3.

### 2.3.7 Pemilihan Model Terbaik Antar Model

Setiap algoritma klasifikasi akan mendapatkan versi terbaiknya pada Skenario 3 setelah melalui semua skenario pengujian (penyeimbangan data, standarisasi, dan pemilihan fitur). Selanjutnya akan dilakukan perbandingan performa akhir dari semua model yang telah dioptimalkan tersebut. Tujuan dari tahap ini adalah untuk mengidentifikasi algoritma klasifikasi mana yang memberikan performa terbaik secara keseluruhan pada kasus deteksi serangan pada dataset IDS.

## BAB III

### HASIL DAN PEMBAHASAN

#### 3.1 Random Forest

##### 3.1.1 Skenario 1: Imbalance vs. SMOTE

Pada skenario ini, model Random Forest dilatih dan diuji menggunakan dataset asli yang tidak seimbang (Imbalance) dan juga data yang telah diseimbangkan dengan SMOTE. Model Random Forest menunjukkan performa keseluruhan yang tinggi pada data tidak seimbang.

Random Forest Performance:				
Accuracy : 0.9470				
Precision: 0.9470				
Recall : 0.9470				
F1-score : 0.9470				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8807	0.8596	0.8700	292
2	0.9968	0.9968	0.9968	316
3	1.0000	1.0000	1.0000	325
4	0.8911	0.9174	0.9041	339
5	1.0000	1.0000	1.0000	330
6	0.8683	0.8850	0.8766	313
7	1.0000	1.0000	1.0000	320
8	1.0000	1.0000	1.0000	325
9	0.9085	0.8797	0.8939	316
10	0.9504	0.9583	0.9544	120
11	0.9545	0.9545	0.9545	44
12	0.8125	0.8125	0.8125	16
13	0.1765	0.2000	0.1875	15
14	0.0769	0.0667	0.0714	15
accuracy			0.9470	3399
macro avg	0.8344	0.8354	0.8348	3399
weighted avg	0.9470	0.9470	0.9470	3399

Gambar 3.1 Report Klasifikasi RF Imbalance

Mayoritas kelas menunjukkan performa yang sangat baik, dengan F1-score mendekati 1.00. Namun, kelas-kelas minoritas seperti DoS attacks-SlowHTTPTest (label 13) dan FTP-BruteForce (label 14) memiliki F1-score yang sangat rendah (0.1875 dan 0.0714). Ini mengindikasikan bahwa model kesulitan mendeteksi serangan-serangan tersebut pada dataset yang tidak seimbang.

Setelah SMOTE, performa Random Forest sedikit berubah, tetapi *F1-score* untuk kelas minoritas menunjukkan peningkatan.

Random Forest Performance with SMOTE				
Accuracy : 0.9459				
Precision: 0.9461				
Recall : 0.9459				
F1-score : 0.9459				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8660	0.8630	0.8645	292
2	1.0000	0.9968	0.9984	316
3	1.0000	1.0000	1.0000	325
4	0.8839	0.9204	0.9017	339
5	1.0000	1.0000	1.0000	330
6	0.8746	0.8690	0.8718	313
7	1.0000	1.0000	1.0000	320
8	1.0000	1.0000	1.0000	325
9	0.9106	0.8703	0.8900	316
10	0.9587	0.9667	0.9627	120
11	0.9773	0.9773	0.9773	44
12	0.7778	0.8750	0.8235	16
13	0.1579	0.2000	0.1765	15
14	0.0000	0.0000	0.0000	15
accuracy			0.9459	3399
macro avg	0.8271	0.8359	0.8311	3399
weighted avg	0.9461	0.9459	0.9459	3399

Gambar 3.2 Report Klasifikasi RF SMOTE

Meskipun overall F1-score sedikit menurun dibandingkan tanpa SMOTE (0.9459 vs 0.9470), F1-score untuk kelas DoS attacks-SlowHTTPTest (label 13) dan FTP-BruteForce (label 14) tetap rendah. Dengan perbandingan F1-score weighted average antara Imbalance (0.9470) dan SMOTE (0.9459), didapatkan bahwa data Imbalance memiliki F1-score yang lebih tinggi. Oleh karena itu, untuk skenario pelatihan selanjutnya, data Imbalance akan digunakan untuk model Random Forest.

Dengan perbandingan antara Imbalance dan SMOTE kita dapatkan dimana menggunakan Imbalance memiliki f1-score yang lebih tinggi maka untuk training selanjutnya kita menggunakan Imbalance.

### 3.1.2 Skenario 2: Normalisasi dengan Imbalance

Pada skenario ini, model Random Forest dilatih menggunakan data Imbalance yang telah dinormalisasi.

Random Forest Performance with Normalisasi				
Accuracy : 0.9382				
Precision: 0.9381				
Recall : 0.9382				
F1-score : 0.9381				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8681	0.8562	0.8621	292
2	0.9968	0.9968	0.9968	316
3	1.0000	1.0000	1.0000	325
4	0.8609	0.8584	0.8597	339
5	0.9970	1.0000	0.9985	330
6	0.8639	0.8722	0.8680	313
7	1.0000	1.0000	1.0000	320
8	1.0000	1.0000	1.0000	325
9	0.8486	0.8513	0.8499	316
10	0.9669	0.9750	0.9710	120
11	0.9767	0.9545	0.9655	44
12	0.8750	0.8750	0.8750	16
13	0.2222	0.2667	0.2424	15
14	0.0833	0.0667	0.0741	15
accuracy			0.9382	3399
macro avg	0.8373	0.8382	0.8375	3399
weighted avg	0.9381	0.9382	0.9381	3399

Gambar 3.3 Report Klasifikasi RF Normalisasi

Performa Random Forest setelah normalisasi sedikit menurun *overall F1-score*-nya menjadi . Kelas minoritas tetap menunjukkan *F1-score* rendah. Ini konsisten dengan sifat model berbasis pohon yang tidak terlalu sensitif terhadap skala data, dan normalisasi tidak memberikan perbaikan yang signifikan.

### 3.1.3 Skenario 3: Selection Fitur dengan Imbalance

Pada skenario ini, model Random Forest dilatih menggunakan data Imbalance yang telah dinormalisasi dan diseleksi fiturnya.

Random Forest Performance with Imbalance and Feature Selection k=70					
Accuracy : 0.9444					
Precision: 0.9444					
Recall : 0.9444					
F1-score : 0.9443					
Detail per kelas:					
	precision	recall	f1-score	support	
0	1.0000	1.0000	1.0000	313	
1	0.8759	0.8459	0.8606	292	
2	0.9968	0.9968	0.9968	316	
3	1.0000	1.0000	1.0000	325	
4	0.8873	0.9056	0.8964	339	
5	1.0000	1.0000	1.0000	330	
6	0.8621	0.8786	0.8703	313	
7	1.0000	1.0000	1.0000	320	
8	0.9969	1.0000	0.9985	325	
9	0.8964	0.8766	0.8864	316	
10	0.9504	0.9583	0.9544	120	
11	0.9773	0.9773	0.9773	44	
12	0.7778	0.8750	0.8235	16	
13	0.1765	0.2000	0.1875	15	
14	0.0769	0.0667	0.0714	15	
accuracy			0.9444	3399	
macro avg	0.8316	0.8387	0.8349	3399	
weighted avg	0.9444	0.9444	0.9443	3399	

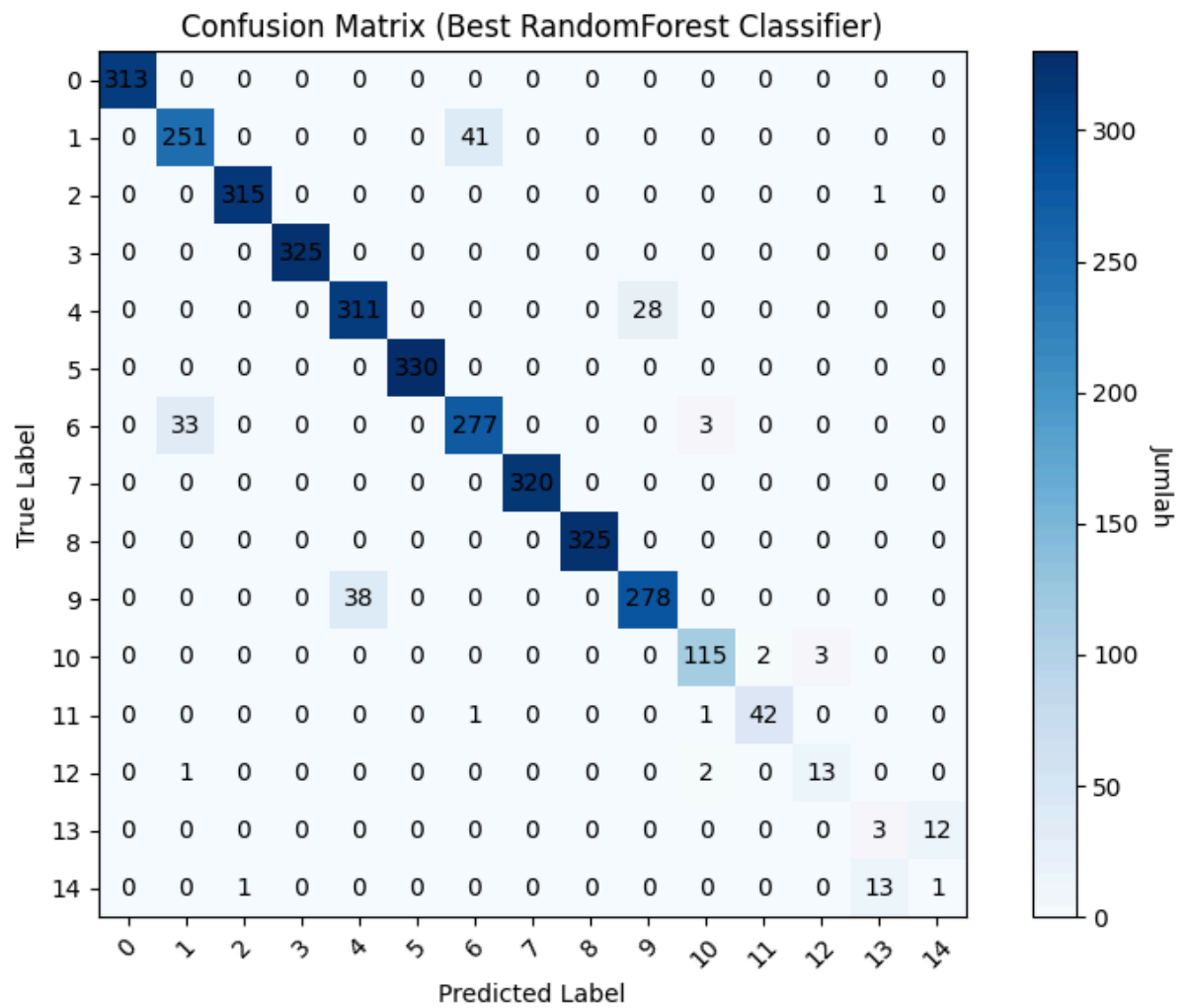
Gambar 3.4 Report Klasifikasi RF Selection Fitur

Performa Random Forest dengan seleksi fitur juga konsisten, dengan *overall F1-score* dan *F1-score* rendah pada kelas minoritas. Ini menunjukkan bahwa seleksi fitur tidak mengatasi masalah deteksi kelas minoritas pada model Random Forest.

### 3.1.4 Kesimpulan

Model Random Forest mencapai F1-score weighted average terbaik sebesar 0.9470 pada Skenario 1 (Data Imbalance). Meskipun demikian, model ini secara konsisten kesulitan dalam mendeteksi kelas minoritas ekstrem seperti DoS attacks-SlowHTTPTest (0.1875) dan FTP-BruteForce (0.0714). Penerapan SMOTE, normalisasi, dan seleksi fitur tidak secara signifikan meningkatkan kemampuan deteksi kelas minoritas pada model ini, dan dalam beberapa kasus bahkan sedikit menurunkan overall F1-score. Optimalitas performa RF tetap pada data aslinya tanpa banyak preprocessing.





Gambar 3.5 Report Klasifikasi RF Confusion Matrix

## 3.2 Logistic Regression

### 3.2.1 Skenario 1: Imbalance vs. SMOTE

Model Logistic Regression dilatih dan diuji menggunakan dataset asli yang tidak seimbang (Imbalance) dan juga data yang telah diseimbangkan dengan SMOTE.

Model Logistic Regression menunjukkan performa yang jauh lebih rendah dibandingkan Random Forest.

Logistik Regression Performance				
Accuracy : 0.3048				
Precision: 0.2573				
Recall : 0.3048				
F1-score : 0.2574				
Detail per kelas:				
	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	313
1	0.6526	0.2123	0.3204	292
2	0.0000	0.0000	0.0000	316
3	0.0000	0.0000	0.0000	325
4	0.7063	1.0000	0.8278	339
5	0.0000	0.0000	0.0000	330
6	0.3765	0.5016	0.4301	313
7	0.1063	0.4344	0.1708	320
8	0.6598	0.6923	0.6757	325
9	0.1440	0.2342	0.1783	316
10	0.0816	0.1333	0.1013	120
11	0.5217	0.5455	0.5333	44
12	0.0000	0.0000	0.0000	16
13	0.0000	0.0000	0.0000	15
14	0.0000	0.0000	0.0000	15
accuracy			0.3048	3399
macro avg	0.2166	0.2502	0.2158	3399
weighted avg	0.2573	0.3048	0.2574	3399

Gambar 3.6 Report Klasifikasi LR Imbalance

Logistic Regression menunjukkan performa yang buruk pada kelas-kelas minoritas, bahkan beberapa kelas memiliki F1-score 0.00. Ini menunjukkan sensitivitas model yang sangat rendah terhadap data yang tidak seimbang. Logistic Regression menunjukkan peningkatan signifikan pada *F1-score* kelas minoritas setelah SMOTE.

Logistik Regression Performance with SMOTE				
Accuracy : 0.3133				
Precision: 0.3308				
Recall : 0.3133				
F1-score : 0.2783				
Detail per kelas:				
	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	313
1	0.8772	0.1712	0.2865	292
2	0.0000	0.0000	0.0000	316
3	0.0000	0.0000	0.0000	325
4	0.7063	1.0000	0.8278	339
5	0.0000	0.0000	0.0000	330
6	0.4220	0.3802	0.4000	313
7	0.1495	0.5531	0.2354	320
8	0.7143	0.6923	0.7031	325
9	0.5606	0.2342	0.3304	316
10	0.1269	0.3500	0.1863	120
11	0.5217	0.5455	0.5333	44
12	0.0000	0.0000	0.0000	16
13	0.0867	1.0000	0.1596	15
14	0.0000	0.0000	0.0000	15
accuracy			0.3133	3399
macro avg	0.2777	0.3284	0.2442	3399
weighted avg	0.3308	0.3133	0.2783	3399

Gambar 3.7 Report Klasifikasi LR SMOTE

Logistic Regression menunjukkan *F1-score weighted average* yang lebih baik (0.2783) dengan SMOTE dibandingkan tanpa SMOTE (0.2574). Oleh karena itu, untuk skenario pelatihan selanjutnya, data dengan SMOTE akan digunakan untuk model Logistic Regression.

### 3.2.2 Skenario 2: Normalisasi dengan SMOTE

Normalisasi secara signifikan meningkatkan performa Logistic Regression, terutama pada kelas minoritas.

Logistik Regression Performance dengan Normalisasi				
Accuracy : 0.8673				
Precision: 0.8890				
Recall : 0.8673				
F1-score : 0.8654				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	0.9872	0.9936	313
1	0.8051	0.6507	0.7197	292
2	0.9906	0.9968	0.9937	316
3	0.9939	1.0000	0.9969	325
4	0.7048	1.0000	0.8268	339
5	0.8512	0.9879	0.9144	330
6	0.7697	0.7476	0.7585	313
7	0.9924	0.8187	0.8973	320
8	0.9939	1.0000	0.9969	325
9	0.9943	0.5506	0.7088	316
10	0.7619	0.6667	0.7111	120
11	0.3761	0.9318	0.5359	44
12	0.4483	0.8125	0.5778	16
13	0.4688	1.0000	0.6383	15
14	0.0000	0.0000	0.0000	15
accuracy			0.8673	3399
macro avg	0.7434	0.8100	0.7513	3399
weighted avg	0.8890	0.8673	0.8654	3399

Gambar 3.8 Report Klasifikasi LR Normalisasi

Dengan normalisasi dan SMOTE, Logistic Regression menunjukkan peningkatan F1-score weighted average yang sangat besar menjadi 0.8654 (dari 0.2783 pada SMOTE). Kelas minoritas seperti DoS attacks-SlowHTTPTest juga meningkat signifikan (0.6383). Meskipun FTP-BruteForce tetap 0.0000, secara keseluruhan terjadi peningkatan substansial. Normalisasi ini sangat penting untuk model berbasis gradien seperti Logistic Regression.

### 3.2.3 Skenario 3: Selection Fitur dan Normalisasi dengan SMOTE

Logistic Regression diuji dengan jumlah fitur (70).

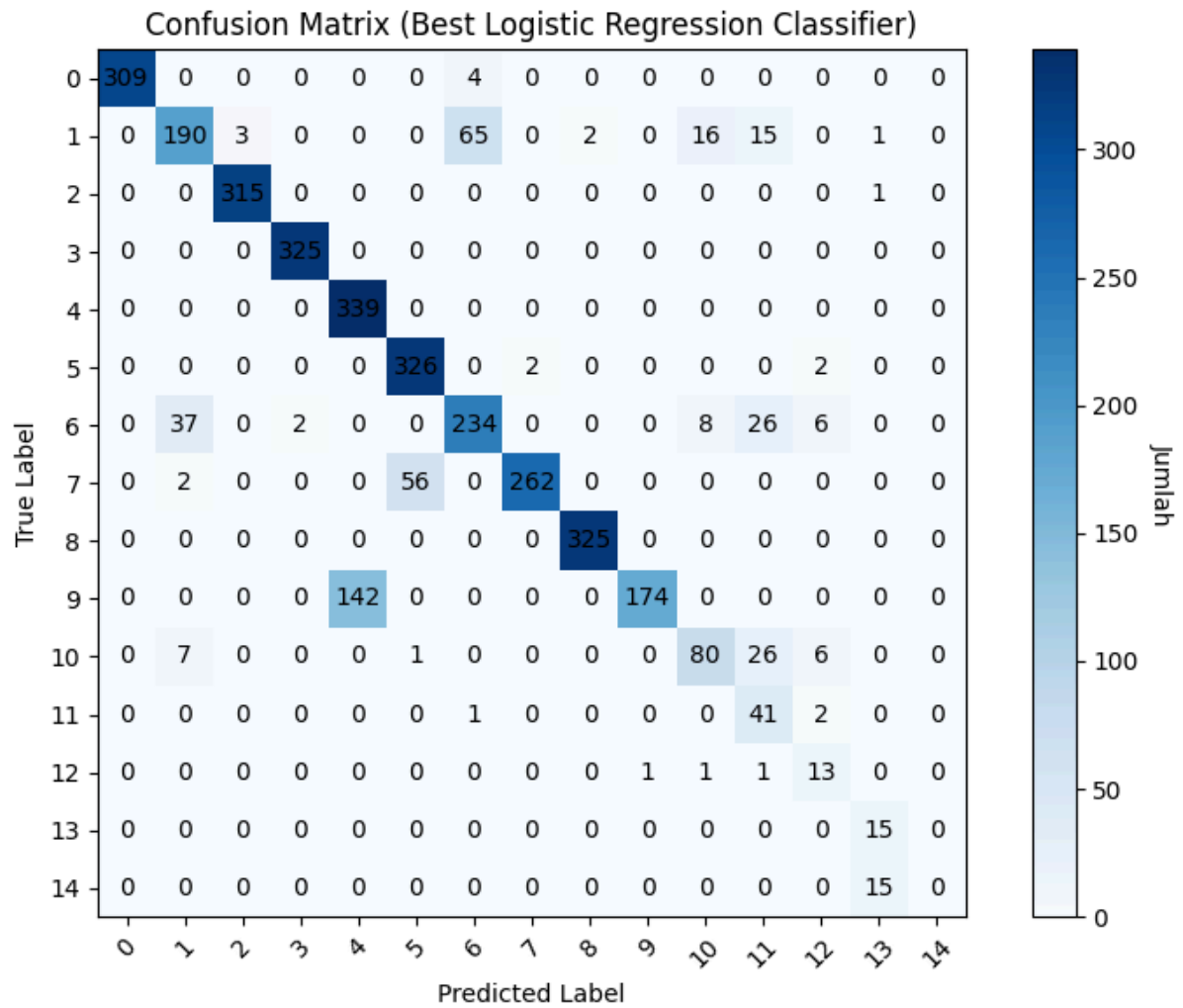
Logistik Regression Performance dengan Normalisasi with k= 70				
Accuracy : 0.8673				
Precision: 0.8890				
Recall : 0.8673				
F1-score : 0.8654				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	0.9872	0.9936	313
1	0.8051	0.6507	0.7197	292
2	0.9906	0.9968	0.9937	316
3	0.9939	1.0000	0.9969	325
4	0.7048	1.0000	0.8268	339
5	0.8512	0.9879	0.9144	330
6	0.7697	0.7476	0.7585	313
7	0.9924	0.8187	0.8973	320
8	0.9939	1.0000	0.9969	325
9	0.9943	0.5506	0.7088	316
10	0.7619	0.6667	0.7111	120
11	0.3761	0.9318	0.5359	44
12	0.4483	0.8125	0.5778	16
13	0.4688	1.0000	0.6383	15
14	0.0000	0.0000	0.0000	15
accuracy			0.8673	3399
macro avg	0.7434	0.8100	0.7513	3399
weighted avg	0.8890	0.8673	0.8654	3399

Gambar 3.9 Report Klasifikasi LR Selection Fitur

Terlihat bahwa F1-score weighted average tetap stabil (0.8654) setelah seleksi fitur, menunjukkan bahwa seleksi fitur berhasil mempertahankan fitur-fitur penting yang menunjang performa model.

### 3.2.4 Kesimpulan

Model Logistic Regression mencapai *F1-score weighted average* terbaik sebesar pada Skenario 2 (Normalisasi dengan SMOTE) dan Skenario 3 (Seleksi Fitur dan Normalisasi dengan SMOTE). Peningkatan ini sangat drastis dibandingkan performa awal pada data imbalance (0.2574). Kombinasi *data balancing* (SMOTE), normalisasi, dan seleksi fitur berhasil meningkatkan kemampuan model ini dalam mendeteksi kelas minoritas secara signifikan, menjadikannya pilihan yang lebih seimbang untuk deteksi berbagai jenis serangan.



Gambar 3.10 Report Klasifikasi LR Confusion Matrix

### 3.3 Gradient Boosting

#### 3.3.1 Skenario 1: Imbalance vs. SMOTE

Gradient Boosting Performance:				
Accuracy : 0.9459				
Precision: 0.9461				
Recall : 0.9459				
F1-score : 0.9459				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8660	0.8630	0.8645	292
2	1.0000	0.9968	0.9984	316
3	1.0000	1.0000	1.0000	325
4	0.8839	0.9204	0.9017	339
5	1.0000	1.0000	1.0000	330
6	0.8746	0.8690	0.8718	313
7	1.0000	1.0000	1.0000	320
8	1.0000	1.0000	1.0000	325
9	0.9106	0.8703	0.8900	316
10	0.9587	0.9667	0.9627	120
11	0.9773	0.9773	0.9773	44
12	0.7778	0.8750	0.8235	16
13	0.1579	0.2000	0.1765	15
14	0.0000	0.0000	0.0000	15
accuracy			0.9459	3399
macro avg	0.8271	0.8359	0.8311	3399
weighted avg	0.9461	0.9459	0.9459	3399

Gambar 3.11 Report Klasifikasi GB Imbalance

Model dengan skenario ini menunjukkan performa baik dengan F1-score *weighted average* sebesar 0.9459. Terlihat jika kelas mayoritas Benign, SSH-Bruteforce, DDos attacks, DOS attacks performanya mendekati atau sama dengan 1.0000. Tetapi untuk kelas paling sedikit seperti SQL Injection, DoS attacks-SlowHTTPTest, FTP-Bruteforce performanya rendah dari 0.8235, 0.1765, dan 0.000.

Gradient Boosting Performance with SMOTE				
Accuracy : 0.9453				
Precision: 0.9459				
Recall : 0.9453				
F1-score : 0.9453				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8821	0.8459	0.8636	292
2	0.9968	0.9937	0.9952	316
3	1.0000	1.0000	1.0000	325
4	0.8750	0.9292	0.9013	339
5	1.0000	1.0000	1.0000	330
6	0.8679	0.8818	0.8748	313
7	0.9969	1.0000	0.9984	320
8	0.9969	0.9969	0.9969	325
9	0.9186	0.8576	0.8871	316
10	0.9355	0.9667	0.9508	120
11	1.0000	0.9545	0.9767	44
12	0.7000	0.8750	0.7778	16
13	0.2222	0.2667	0.2424	15
14	0.1538	0.1333	0.1429	15
accuracy			0.9453	3399
macro avg	0.8364	0.8468	0.8405	3399
weighted avg	0.9459	0.9453	0.9453	3399

Gambar 3.12 Report Klasifikasi GB SMOTE

Setelah di SMOTE, performa model ini mengalami penurunan F1-score weighted average menjadi 0.9453. Kelas mayoritas tetap stabil dengan performa tinggi, kelas minoritas seperti Dos attacks-SlowHTTPTest meningkat dari 0.1765 menjadi 0.2424, dan FTP Bruteforce yang awalnya 0.000 sekarang menjadi 0.1429

Disini terjadi pertukaran performa dimana kelas minoritas yang tadinya tidak bisa dideteksi sekarang menjadi bisa tetapi penurunan performa skor untuk keseluruhan nilai pada model. Sehingga F1-score weighted average antar Imbalance bernilai 0.9459 dan SMOTE 0.9453.



### 3.3.2 Skenario 2: Normalisasi dengan Imbalance

Di skenario ini, Gradient Boosting akan di latih dengan data Imbalance yang sudah dinormalisasi menggunakan Standard Scaler.

Gradient Boosting Performance with Normalisasi Imbalance					
Accuracy : 0.9400					
Precision: 0.9408					
Recall : 0.9400					
F1-score : 0.9401					
Detail per kelas:					
	precision	recall	f1-score	support	
0	1.0000	1.0000	1.0000	313	
1	0.8641	0.8493	0.8566	292	
2	1.0000	0.9968	0.9984	316	
3	1.0000	1.0000	1.0000	325	
4	0.8489	0.9115	0.8791	339	
5	1.0000	1.0000	1.0000	330	
6	0.8634	0.8882	0.8756	313	
7	1.0000	1.0000	1.0000	320	
8	1.0000	1.0000	1.0000	325	
9	0.8969	0.8259	0.8600	316	
10	0.9664	0.9583	0.9623	120	
11	0.9767	0.9545	0.9655	44	
12	0.7692	0.6250	0.6897	16	
13	0.1250	0.1333	0.1290	15	
14	0.1250	0.1333	0.1290	15	
accuracy			0.9400	3399	
macro avg	0.8290	0.8184	0.8230	3399	
weighted avg	0.9408	0.9400	0.9401	3399	

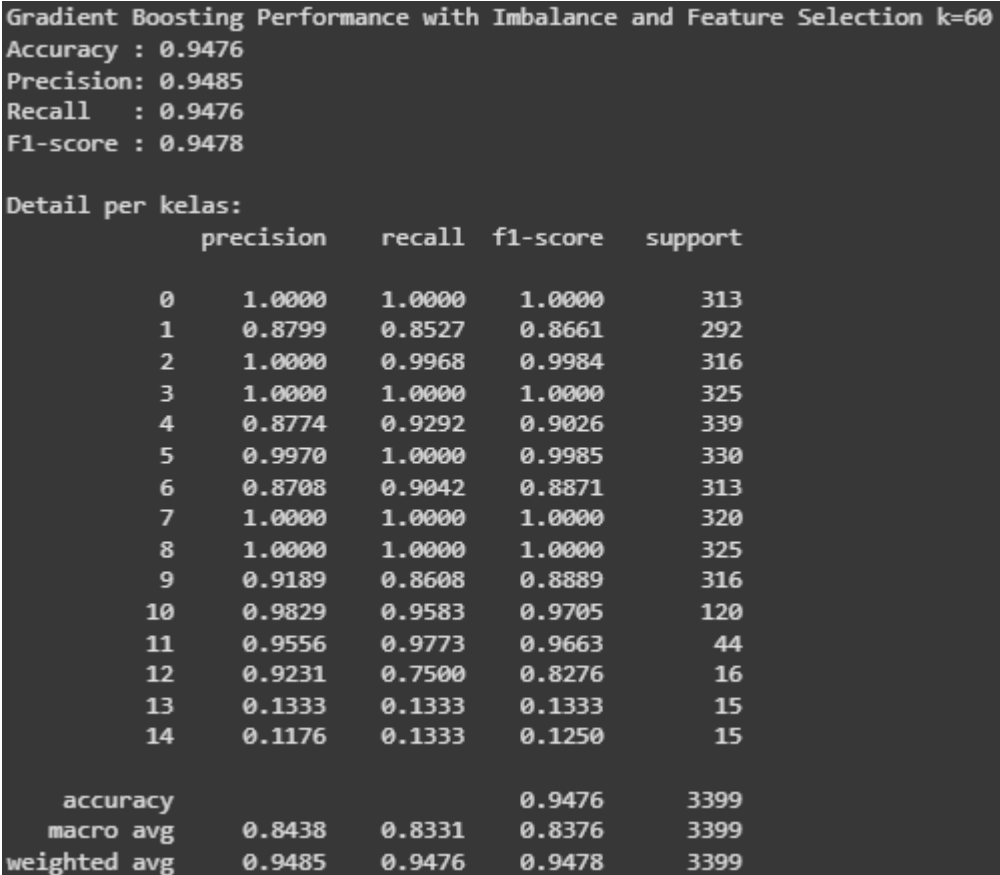
Gambar 3.13 Report Klasifikasi GB Normalisasi

Hasil pada Gambar 3.13 menampilkan performa model F1 score weighted average menjadi 0.9401 Penurunan performa dari 0.9459 (tanpa normalisasi) menjadi 0.9401 (dengan normalisasi) Kelas mayoritas masih mempertahankan performa tinggi, namun dengan sedikit degradasi. Kelas minoritas seperti SQL Injection (label 12) mengalami penurunan dari 0.8235 menjadi 0.6897.

Sehingga Gradient Boosting tidak memerlukan normalisasi data karena dapat penurunan performa setelah diberikan preprocessing.

### 3.3.3 Skenario 3: Imbalance dengan Selection Fitur

Pada skenario terakhir model ini diuji dengan SelectKBest. Hasil terbaik ketika  $k = 60$  fitur.



```
Gradient Boosting Performance with Imbalance and Feature Selection k=60
Accuracy : 0.9476
Precision: 0.9485
Recall   : 0.9476
F1-score : 0.9478

Detail per kelas:
      precision    recall  f1-score   support

0      1.0000      1.0000      1.0000       313
1      0.8799      0.8527      0.8661       292
2      1.0000      0.9968      0.9984       316
3      1.0000      1.0000      1.0000       325
4      0.8774      0.9292      0.9026       339
5      0.9970      1.0000      0.9985       330
6      0.8708      0.9042      0.8871       313
7      1.0000      1.0000      1.0000       320
8      1.0000      1.0000      1.0000       325
9      0.9189      0.8608      0.8889       316
10     0.9829      0.9583      0.9705       120
11     0.9556      0.9773      0.9663         44
12     0.9231      0.7500      0.8276         16
13     0.1333      0.1333      0.1333          15
14     0.1176      0.1333      0.1250          15

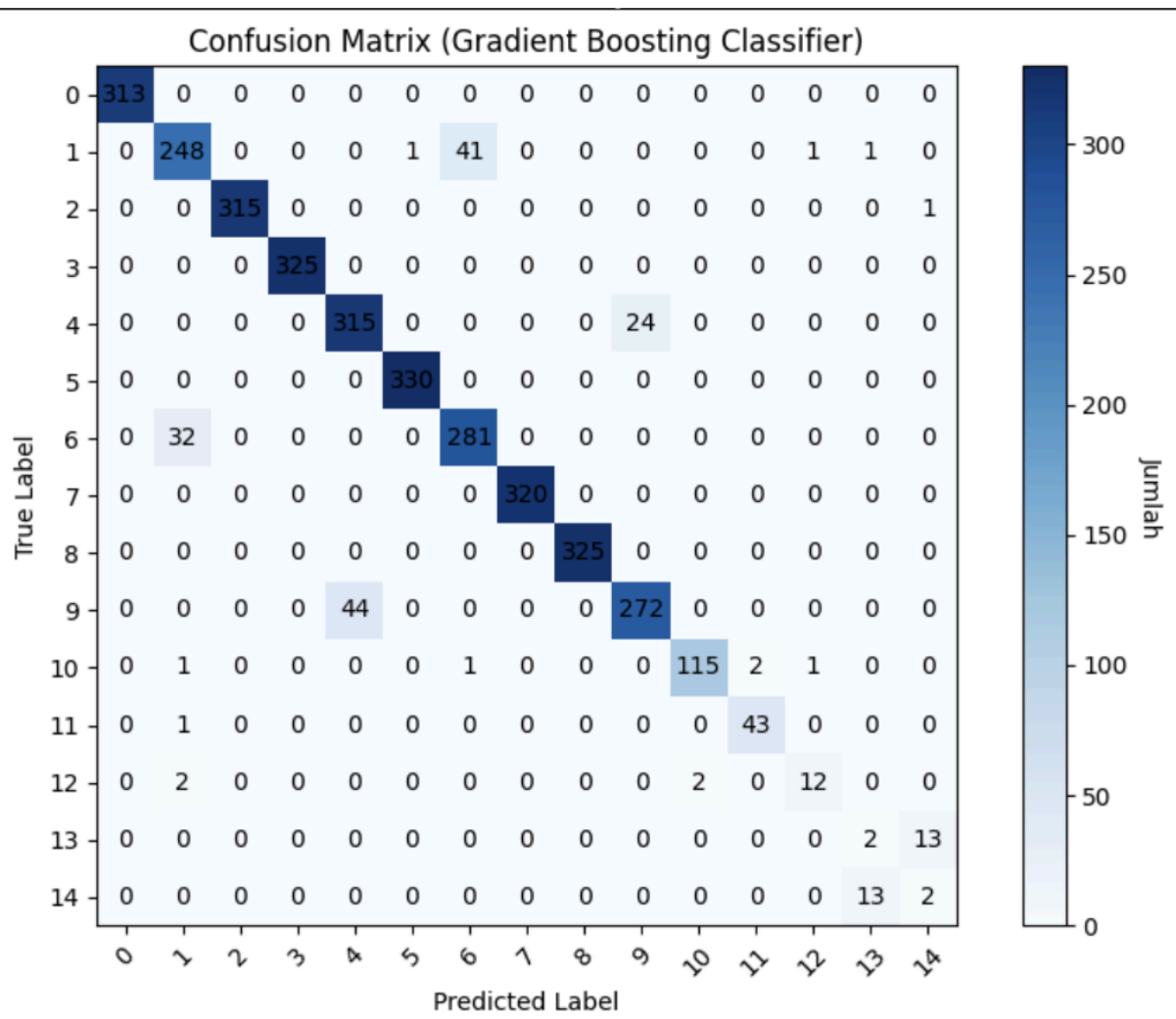
accuracy          0.9476       3399
macro avg         0.8438      0.8331      0.8376       3399
weighted avg      0.9485      0.9476      0.9478       3399
```

Gambar 3.14 Report Klasifikasi GB Selection Fitur dan Imbalance

Terdapat peningkatan F1-score weighted average menjadi 0.9478 yang adalah hasil terbaik untuk model Gradient Boosting. Sehingga seleksi fitur memberikan hasil terbaik bahwa eliminasi fitur noise dan redundant meningkatkan kemampuan model.

### 3.3.4 Kesimpulan

Model Gradient Boosting mencapai F1-score terbaik pada Skenario Selection Fitur dengan  $K = 60$ . Nilai F1 score sebesar 0.9478. Model menunjukkan kemampuan baik untuk deteksi kelas mayoritas, tetapi masih kesulitan pada kelas Dos attacks-SlowHTTPTest (F1-score 0.1290) dan FTP - BruteFroce ( F1-score 0.1290). Secara keseluruhan ini stabilitas baik dengan preprocessing minimal.



Gambar 3.15 Report Klasifikasi GB Confusion Matrix

### 3.4 CatBoost

#### 3.4.1 Skenario 1: Imbalance vs. SMOTE

Accuracy : 0.9482				
Precision: 0.9482				
Recall : 0.9482				
F1-score : 0.9481				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8794	0.8493	0.8641	292
2	1.0000	0.9968	0.9984	316
3	1.0000	1.0000	1.0000	325
4	0.8928	0.9086	0.9006	339
5	1.0000	1.0000	1.0000	330
6	0.8642	0.8946	0.8791	313
7	1.0000	1.0000	1.0000	320
8	1.0000	1.0000	1.0000	325
9	0.9000	0.8829	0.8914	316
10	0.9752	0.9833	0.9793	120
11	1.0000	0.9773	0.9885	44
12	0.8000	0.7500	0.7742	16
13	0.2857	0.4000	0.3333	15
14	0.1000	0.0667	0.0800	15
accuracy			0.9482	3399
macro avg	0.8465	0.8473	0.8459	3399
weighted avg	0.9482	0.9482	0.9481	3399

Gambar 3.16 Report Klasifikasi CB Imbalance

Catboost dengan skenario ini menunjukkan performa terbaik di antara semua skenario model yang di uji. Kemampuan luar biasa dalam deteksi mayoritas kelas dan kelas minoritas seperti Dos attacks-SlowHTTPTest (F1-score 0.3333) dan FTP-BruteForce (F1-score 0.0800) masih menunjukkan deteksi yang lebih baik dibandingkan model lain pada kondisi yang sama.

CatBoost Performance with SMOTE					
Accuracy : 0.9465					
Precision: 0.9468					
Recall : 0.9465					
F1-score : 0.9465					
Detail per kelas:					
	precision	recall	f1-score	support	
0	1.0000	1.0000	1.0000	313	
1	0.8953	0.8493	0.8717	292	
2	1.0000	0.9968	0.9984	316	
3	1.0000	1.0000	1.0000	325	
4	0.8832	0.9145	0.8986	339	
5	1.0000	1.0000	1.0000	330	
6	0.8642	0.8946	0.8791	313	
7	1.0000	1.0000	1.0000	320	
8	1.0000	1.0000	1.0000	325	
9	0.9046	0.8703	0.8871	316	
10	0.9426	0.9583	0.9504	120	
11	0.9773	0.9773	0.9773	44	
12	0.7222	0.8125	0.7647	16	
13	0.2222	0.2667	0.2424	15	
14	0.0769	0.0667	0.0714	15	
accuracy			0.9465	3399	
macro avg	0.8326	0.8405	0.8361	3399	
weighted avg	0.9468	0.9465	0.9465	3399	

Gambar 3.17 Report Klasifikasi CB SMOTE

Setelah diberikan preprocessing SMOTE, F1-score menurun menjadi 0.9465. Tetapi peningkatan pada kelas minoritas seperti DoS attacks-SlowHTTPTEST yang meningkat menjadi 0.2424 dan FTP-BruteForce menjadi 0.0714.

### 3.4.2 Skenario 2: Normalisasi dengan Imbalance

Penerapan normalisasi data imbalance menghasilkan F1-score 0.9415 yang mana adalah penurunan dibandingkan data tanpa normalisasi (0.9481)

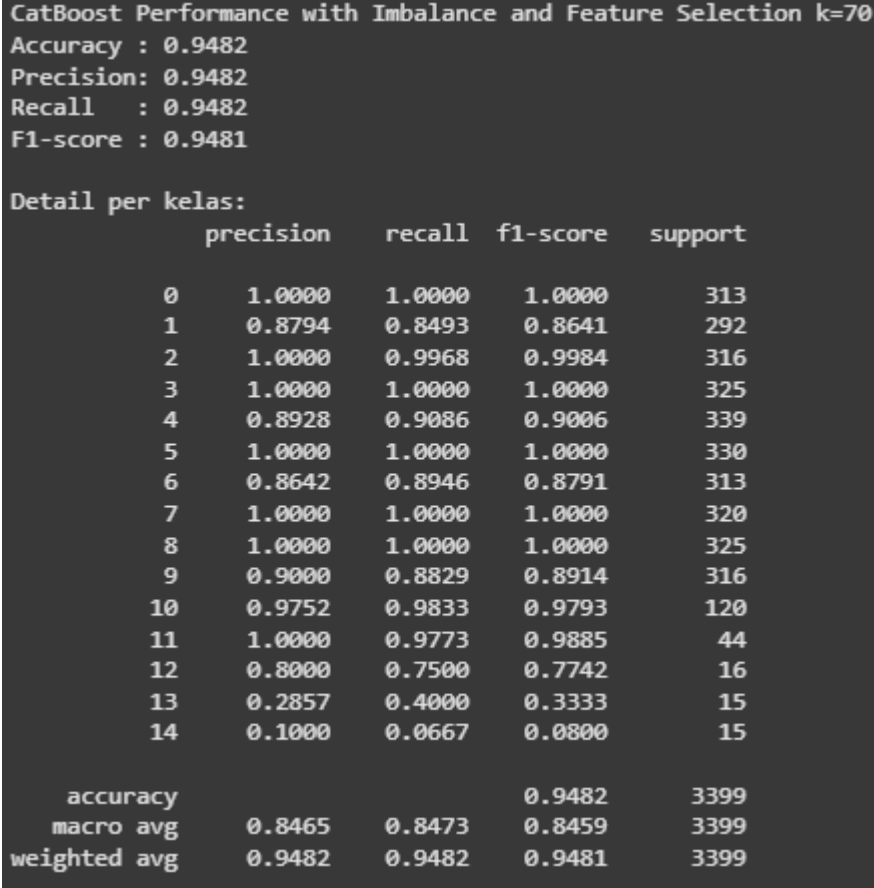
CatBoost Performance with Normalisasi				
Accuracy : 0.9420				
Precision: 0.9414				
Recall : 0.9420				
F1-score : 0.9415				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8651	0.8562	0.8606	292
2	1.0000	0.9968	0.9984	316
3	1.0000	1.0000	1.0000	325
4	0.8555	0.8909	0.8728	339
5	1.0000	1.0000	1.0000	330
6	0.8675	0.8786	0.8730	313
7	1.0000	1.0000	1.0000	320
8	1.0000	1.0000	1.0000	325
9	0.8775	0.8386	0.8576	316
10	0.9752	0.9833	0.9793	120
11	1.0000	0.9773	0.9885	44
12	0.8000	0.7500	0.7742	16
13	0.3600	0.6000	0.4500	15
14	0.0000	0.0000	0.0000	15
accuracy			0.9420	3399
macro avg	0.8401	0.8514	0.8436	3399
weighted avg	0.9414	0.9420	0.9415	3399

Gambar 3.18 Report Klasifikasi CB Normalisasi

Data imbalance memberikan performa bagus. Hal ini menunjukkan bahwa CatBoost baik dalam menangani class imbalance tanpa perlu dibantu preprocessing

### 3.4.3 Skenario 3: Selection Fitur dengan Imbalance

Pada skenario ini, CatBoost menggunakan seleksi fitur dengan data Imbalance. Setelah mencoba menentukan nilai k dari 10 hingga 80. Pengujian dengan hasil optimal berada pada k = 70.



```
CatBoost Performance with Imbalance and Feature Selection k=70
Accuracy : 0.9482
Precision: 0.9482
Recall   : 0.9482
F1-score : 0.9481

Detail per kelas:
      precision    recall  f1-score   support

    0       1.0000      1.0000      1.0000       313
    1       0.8794      0.8493      0.8641       292
    2       1.0000      0.9968      0.9984       316
    3       1.0000      1.0000      1.0000       325
    4       0.8928      0.9086      0.9006       339
    5       1.0000      1.0000      1.0000       330
    6       0.8642      0.8946      0.8791       313
    7       1.0000      1.0000      1.0000       320
    8       1.0000      1.0000      1.0000       325
    9       0.9000      0.8829      0.8914       316
   10       0.9752      0.9833      0.9793       120
   11       1.0000      0.9773      0.9885        44
   12       0.8000      0.7500      0.7742        16
   13       0.2857      0.4000      0.3333         15
   14       0.1000      0.0667      0.0800         15

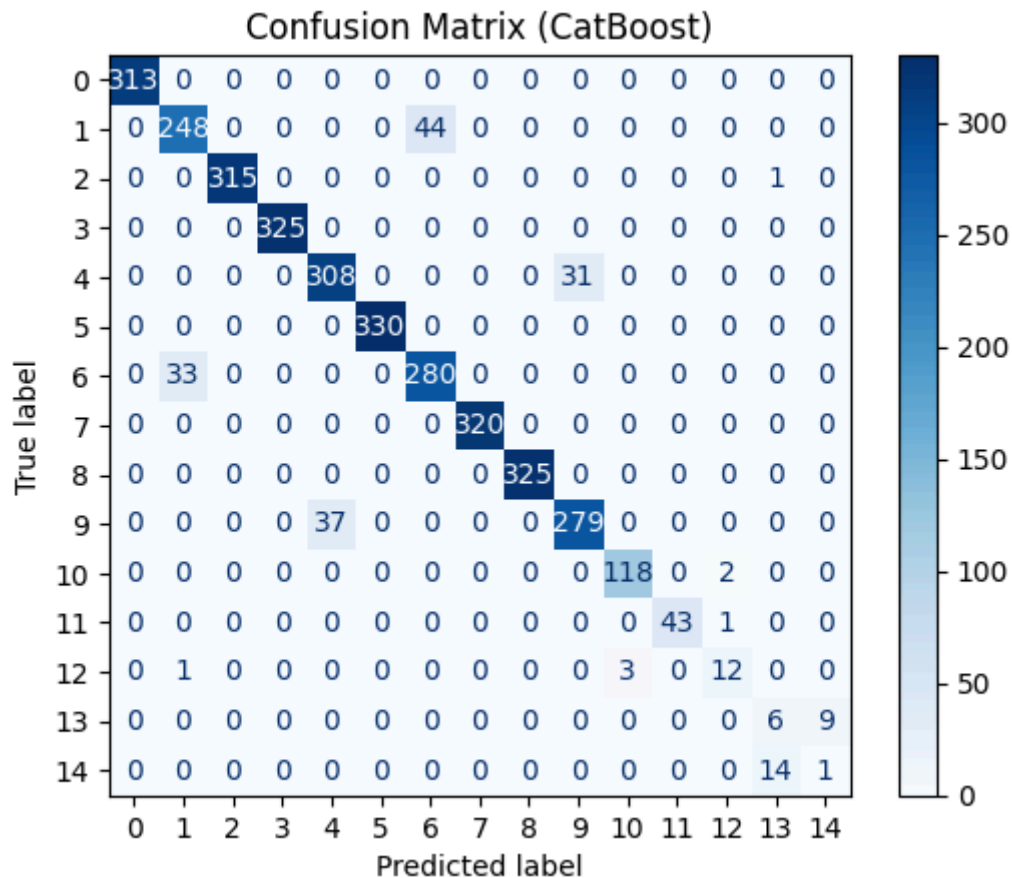
 accuracy
macro avg      0.8465      0.8473      0.8459       3399
weighted avg    0.9482      0.9482      0.9481       3399
```

Gambar 3.19 Report Klasifikasi CB Imbalance dengan Selection Feature

Hasil pada gambar menunjukkan performa yang sangat menarik dengan F1-score weighted average 0.9481, yang mengembalikan performa ke level optimal seperti data Imbalance asli. Accuracy, Precision, Recall sama dengan nilai 0.9482 dan F1-score 0.9481. Nilai ini sama seperti data imbalance pada skenario 1.

### 3.4.4 Kesimpulan

Model Catboost menunjukkan performa terbaik pada dua skenario, yaitu Data imbalance tanpa preprocessing dan data dengan selection fitur  $k = 70$  dengan data imbalance. Ini menunjukkan bahwa model catboost stabil dan optimal baik dengan preprocessing maupun tidak. Model ini juga tahan terhadap overfit dan fitur noise.



Gambar 3.20 Report Klasifikasi CB Confusion Matrix



### 3.5 SVM

#### 3.5.1 Skenario 1: Imbalance vs. SMOTE

SVM Performance:				
Accuracy : 0.0859				
Precision: 0.0074				
Recall : 0.0859				
F1-score : 0.0136				
Detail per kelas:				
	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	313
1	0.0859	1.0000	0.1582	292
2	0.0000	0.0000	0.0000	316
3	0.0000	0.0000	0.0000	325
4	0.0000	0.0000	0.0000	339
5	0.0000	0.0000	0.0000	330
6	0.0000	0.0000	0.0000	313
7	0.0000	0.0000	0.0000	320
8	0.0000	0.0000	0.0000	325
9	0.0000	0.0000	0.0000	316
10	0.0000	0.0000	0.0000	120
11	0.0000	0.0000	0.0000	44
12	0.0000	0.0000	0.0000	16
13	0.0000	0.0000	0.0000	15
14	0.0000	0.0000	0.0000	15
accuracy			0.0859	3399
macro avg	0.0057	0.0667	0.0105	3399
weighted avg	0.0074	0.0859	0.0136	3399

Gambar 3.21 Report Klasifikasi SVM Imbalance

Model SVM menunjukkan performa yang sangat buruk pada data tidak seimbang dengan F1-score weighted average hanya 0.0136. Hampir semua kelas memiliki F1-score 0.0000, kecuali kelas Benign (label 1) yang mencapai 0.1582. Hal ini menunjukkan SVM sangat sensitif terhadap ketidakseimbangan data (*imbalance*) dan hampir tidak mampu mendeteksi kelas minoritas.

SVM Performance with SMOTE				
Accuracy : 0.3463				
Precision: 0.3479				
Recall : 0.3463				
F1-score : 0.2475				
Detail per kelas:				
	precision	recall	f1-score	support
0	0.2290	1.0000	0.3726	313
1	0.0000	0.0000	0.0000	292
2	0.4068	0.9810	0.5751	316
3	0.1290	0.0246	0.0413	325
4	0.4795	1.0000	0.6482	339
5	0.0000	0.0000	0.0000	330
6	0.0000	0.0000	0.0000	313
7	0.6667	0.0688	0.1246	320
8	0.9885	0.2646	0.4175	325
9	0.6981	0.2342	0.3507	316
10	0.0000	0.0000	0.0000	120
11	0.5106	0.5455	0.5275	44
12	0.0049	0.0625	0.0090	16
13	0.0000	0.0000	0.0000	15
14	0.0000	0.0000	0.0000	15
accuracy			0.3463	3399
macro avg	0.2742	0.2787	0.2044	3399
weighted avg	0.3479	0.3463	0.2475	3399

Gambar 3.22 Report Klasifikasi SVM SMOTE

Setelah penerapan SMOTE, terjadi peningkatan signifikan dengan F1-score weighted average menjadi 0.2475. Beberapa kelas mulai bisa dideteksi, seperti Bot (label 0) dengan F1-score 0.3726 dan SSH-Bruteforce (label 2) dengan 0.5751. Meskipun masih rendah, ini menunjukkan bahwa SMOTE membantu SVM dalam mengatasi masalah ketidakseimbangan data(*imbalance*).

### 3.5.2 Skenario 2: Normalisasi dengan SMOTE

SVM Performance with Normalisasi					
Accuracy : 0.8706					
Precision: 0.8892					
Recall : 0.8706					
F1-score : 0.8707					
Detail per kelas:					
	precision	recall	f1-score	support	
0	0.9904	0.9872	0.9888	313	
1	0.8085	0.6507	0.7211	292	
2	0.9937	0.9937	0.9937	316	
3	0.9878	1.0000	0.9939	325	
4	0.7604	0.9735	0.8538	339	
5	0.8069	0.9879	0.8883	330	
6	0.7843	0.7668	0.7754	313	
7	0.9837	0.7562	0.8551	320	
8	0.9969	0.9969	0.9969	325	
9	0.9550	0.6709	0.7881	316	
10	0.8421	0.6667	0.7442	120	
11	0.3500	0.9545	0.5122	44	
12	0.5217	0.7500	0.6154	16	
13	0.0000	0.0000	0.0000	15	
14	0.4333	0.8667	0.5778	15	
accuracy			0.8706	3399	
macro avg	0.7477	0.8014	0.7536	3399	
weighted avg	0.8892	0.8706	0.8707	3399	

Gambar 3.23 Report Klasifikasi SVM Normalisasi

Normalisasi memberikan dampak pada performa SVM dengan F1-score weighted average melonjak drastis menjadi 0.8707. Hampir semua kelas menunjukkan peningkatan yang luar biasa, dengan mayoritas kelas mencapai F1-score di atas 0.8. Ini membuktikan bahwa SVM sangat memerlukan normalisasi data untuk dapat bekerja optimal, karena algoritma ini sangat sensitif terhadap skala fitur.

### 3.5.3 Skenario 3: Selection Fitur dan Normalisasi dengan SMOTE

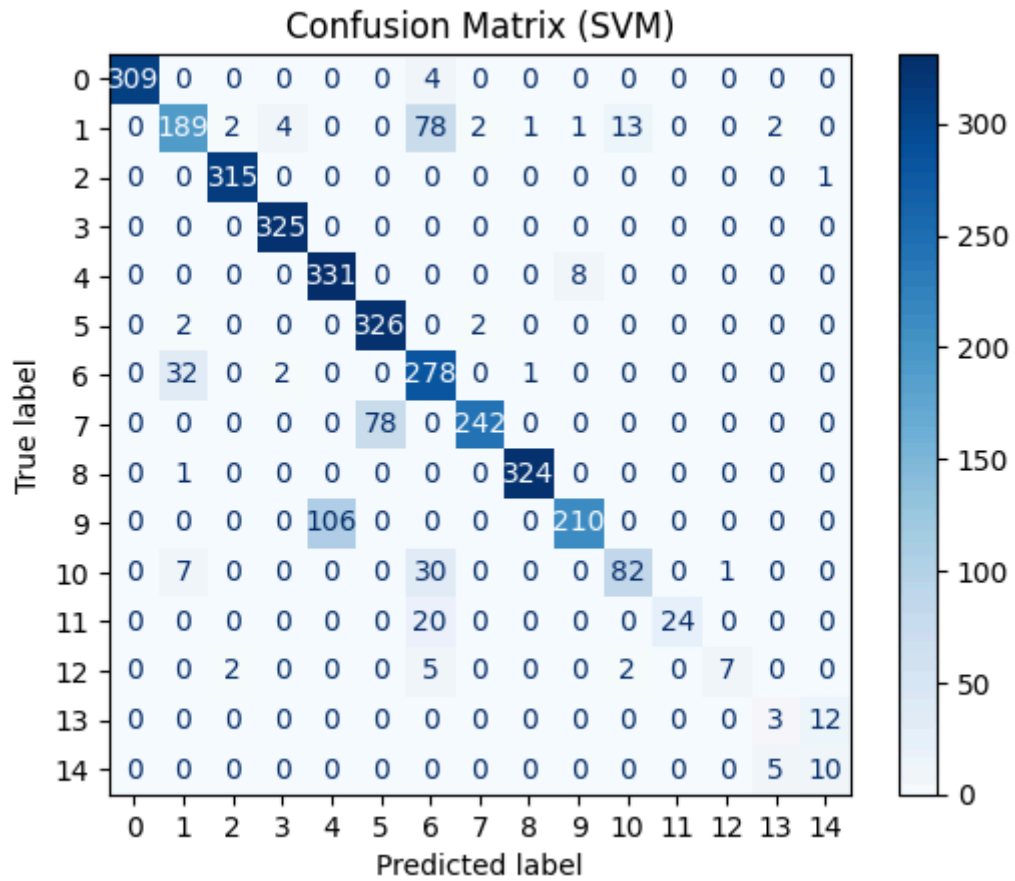
SVM Performance with Normalisasi and k = 60				
Accuracy : 0.8720				
Precision: 0.8900				
Recall : 0.8720				
F1-score : 0.8724				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	0.9872	0.9936	313
1	0.8085	0.6507	0.7211	292
2	0.9937	0.9937	0.9937	316
3	0.9878	1.0000	0.9939	325
4	0.7676	0.9646	0.8549	339
5	0.8069	0.9879	0.8883	330
6	0.7864	0.7764	0.7814	313
7	0.9837	0.7562	0.8551	320
8	0.9969	0.9969	0.9969	325
9	0.9435	0.6867	0.7949	316
10	0.8421	0.6667	0.7442	120
11	0.3500	0.9545	0.5122	44
12	0.5217	0.7500	0.6154	16
13	0.0000	0.0000	0.0000	15
14	0.4333	0.8667	0.5778	15
accuracy			0.8720	3399
macro avg	0.7482	0.8025	0.7549	3399
weighted avg	0.8900	0.8720	0.8724	3399

Gambar 3.24 Report Klasifikasi SVM Selection Fitur

Dengan seleksi fitur k=60, F1-score weighted average sedikit meningkat menjadi 0.8724. Peningkatan ini menunjukkan bahwa eliminasi fitur yang tidak relevan atau noise dapat membantu SVM dalam membuat keputusan klasifikasi yang lebih baik, meskipun peningkatannya tidak sebesar dampak normalisasi.

### 3.5.4 Kesimpulan

Model *Support Vector Machine* (SVM) mencapai *F1-score weighted average* terbaik sebesar pada Skenario 2 (Normalisasi dengan SMOTE) dan Skenario 3 (Seleksi Fitur dan Normalisasi dengan SMOTE). Peningkatan ini sangat drastis dibandingkan performa awal pada data imbalance (0.0136). Kombinasi *data balancing* (SMOTE), normalisasi, dan seleksi fitur berhasil meningkatkan kemampuan model ini dalam mendeteksi kelas minoritas secara signifikan, menjadikannya pilihan yang lebih seimbang untuk deteksi berbagai jenis serangan.



Gambar 3.25 Report Klasifikasi SVM Confusion Matrix

### 3.6 KNN

#### 3.6.1 Skenario 1: Imbalance vs. SMOTE

KNN Performance:				
Accuracy : 0.9053				
Precision: 0.9038				
Recall : 0.9053				
F1-score : 0.9041				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8129	0.7740	0.7930	292
2	0.9873	0.9842	0.9857	316
3	1.0000	0.9938	0.9969	325
4	0.8343	0.8466	0.8404	339
5	0.9326	0.9636	0.9478	330
6	0.7918	0.8019	0.7968	313
7	0.9697	1.0000	0.9846	320
8	0.9817	0.9877	0.9847	325
9	0.8247	0.8038	0.8141	316
10	0.8783	0.8417	0.8596	120
11	0.7143	0.7955	0.7527	44
12	0.7273	0.5000	0.5926	16
13	0.3636	0.5333	0.4324	15
14	0.1667	0.0667	0.0952	15
accuracy			0.9053	3399
macro avg	0.7990	0.7929	0.7918	3399
weighted avg	0.9038	0.9053	0.9041	3399

Gambar 3.26 Report Klasifikasi KNN Imbalance

Model KNN menunjukkan performa yang sudah cukup baik pada data tidak seimbang dengan F1-score weighted average 0.9041. Mayoritas kelas memiliki F1-score yang tinggi, meskipun beberapa kelas minoritas seperti DoS attacks-SlowHTTPTest (label 13) masih rendah dengan 0.4324. Ini menunjukkan bahwa KNN lebih robust terhadap ketidakseimbangan data dibandingkan SVM.

KNN Performance with SMOTE:					
Accuracy : 0.9050					
Precision: 0.9080					
Recall : 0.9050					
F1-score : 0.9057					
Detail per kelas:					
	precision	recall	f1-score	support	
0	1.0000	1.0000	1.0000	313	
1	0.8278	0.7740	0.8000	292	
2	0.9968	0.9810	0.9888	316	
3	1.0000	0.9938	0.9969	325	
4	0.8348	0.8496	0.8421	339	
5	0.9461	0.9576	0.9518	330	
6	0.7950	0.8051	0.8000	313	
7	0.9756	1.0000	0.9877	320	
8	0.9847	0.9877	0.9862	325	
9	0.8274	0.8038	0.8154	316	
10	0.9307	0.7833	0.8507	120	
11	0.6452	0.9091	0.7547	44	
12	0.5652	0.8125	0.6667	16	
13	0.1739	0.2667	0.2105	15	
14	0.1538	0.1333	0.1429	15	
accuracy			0.9050	3399	
macro avg	0.7771	0.8038	0.7863	3399	
weighted avg	0.9080	0.9050	0.9057	3399	

Gambar 3.27 Report Klasifikasi KNN SMOTE

### 3.6.2 Skenario 2: Normalisasi dengan SMOTE

Penerapan SMOTE memberikan sedikit peningkatan dengan F1-score weighted average menjadi 0.9057. Beberapa kelas minoritas mengalami perbaikan, seperti DoS attacks-SlowHTTPTest yang meningkat menjadi 0.2105, meskipun secara keseluruhan peningkatannya tidak terlalu signifikan.

KNN Performance dengan SMOTE dan Normalisasi:				
Accuracy : 0.9176				
Precision: 0.9205				
Recall : 0.9176				
F1-score : 0.9182				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8303	0.7877	0.8084	292
2	0.9937	0.9968	0.9953	316
3	1.0000	1.0000	1.0000	325
4	0.8162	0.8643	0.8395	339
5	0.9970	0.9909	0.9939	330
6	0.8360	0.8466	0.8413	313
7	0.9969	1.0000	0.9984	320
8	0.9969	0.9969	0.9969	325
9	0.8441	0.7880	0.8151	316
10	0.9412	0.8000	0.8649	120
11	0.6508	0.9318	0.7664	44
12	0.6522	0.9375	0.7692	16
13	0.2000	0.2667	0.2286	15
14	0.1429	0.1333	0.1379	15
accuracy			0.9176	3399
macro avg	0.7932	0.8227	0.8037	3399
weighted avg	0.9205	0.9176	0.9182	3399

Gambar 3.28 Report Klasifikasi KNN Normalisasi

### 3.6.3 Skenario 3: Selection Fitur dan Normalisasi dengan SMOTE

Normalisasi memberikan dampak positif pada KNN dengan F1-score weighted average meningkat menjadi 0.9182. Peningkatan ini cukup signifikan karena KNN adalah algoritma berbasis jarak, sehingga normalisasi membantu dalam menghitung jarak antar data point dengan lebih akurat dan mengurangi bias dari fitur dengan skala yang berbeda.

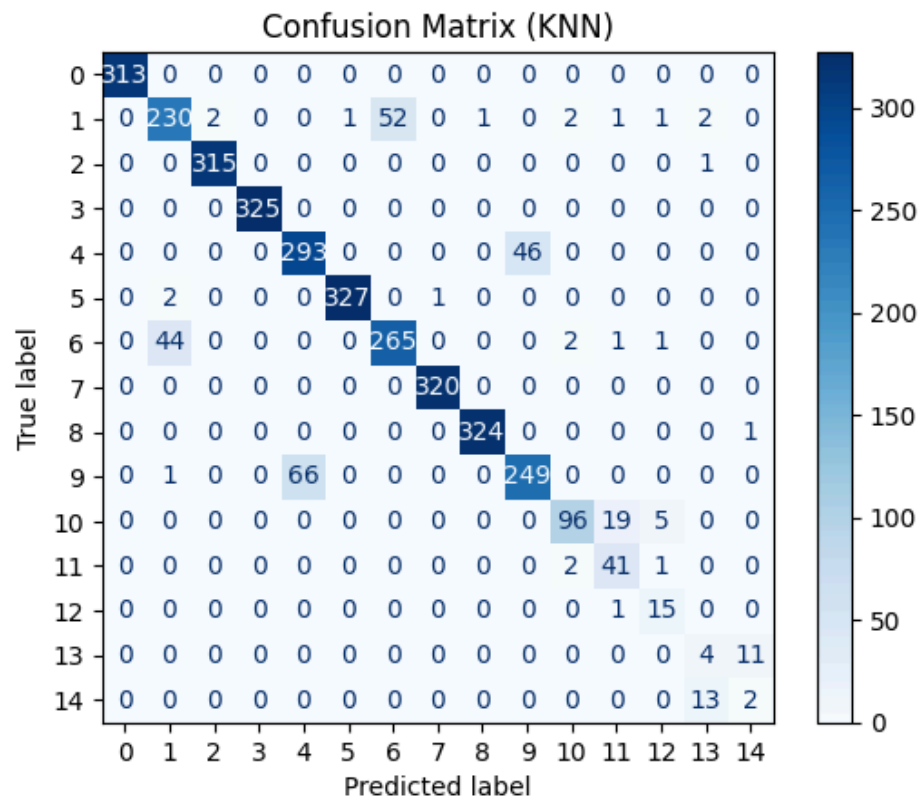


KNN Performance with Normalization and Feature Selection k = 70:				
Accuracy : 0.9176				
Precision: 0.9205				
Recall : 0.9176				
F1-score : 0.9182				
Detail per kelas:				
	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	313
1	0.8303	0.7877	0.8084	292
2	0.9937	0.9968	0.9953	316
3	1.0000	1.0000	1.0000	325
4	0.8162	0.8643	0.8395	339
5	0.9970	0.9909	0.9939	330
6	0.8360	0.8466	0.8413	313
7	0.9969	1.0000	0.9984	320
8	0.9969	0.9969	0.9969	325
9	0.8441	0.7880	0.8151	316
10	0.9412	0.8000	0.8649	120
11	0.6508	0.9318	0.7664	44
12	0.6522	0.9375	0.7692	16
13	0.2000	0.2667	0.2286	15
14	0.1429	0.1333	0.1379	15
accuracy			0.9176	3399
macro avg	0.7932	0.8227	0.8037	3399
weighted avg	0.9205	0.9176	0.9182	3399

Gambar 3.29 Report Klasifikasi KNN Selection Fitur

Dengan seleksi fitur k=70, F1-score weighted average tetap stabil di 0.9182. Meskipun tidak ada peningkatan, stabilitas performa ini menunjukkan bahwa seleksi fitur berhasil mempertahankan informasi penting sambil mengurangi dimensi data, yang dapat bermanfaat untuk efisiensi komputasi tanpa mengorbankan akurasi.

### 3.6.4 Kesimpulan



Gambar 3.30 Report Klasifikasi KNN Confusion Matrix

Secara keseluruhan, confusion matrix ini mengkonfirmasi bahwa KNN efektif untuk deteksi sebagian besar jenis serangan jaringan, dengan kelemahan utama pada deteksi serangan yang memiliki representasi data sangat terbatas dalam dataset. Misklasifikasi yang terjadi umumnya masih dalam konteks yang logis, dimana serangan dengan karakteristik serupa cenderung saling tertukar dalam prediksi.

KNN efektif untuk klasifikasi tetapi kurang pada deteksi serangan yang datanya terbatas di dataset. Terjadi salah klasifikasi dimana karakteristik saling tertukar dalam prediksi. Untuk kelas minoritas terutama DoS attacks-SlowHTTPTest (label 13) hanya memprediksi 4 dari 15 sampel yang benar.

Tetapi DoS attacks-Slowloris (label 8) diprediksi hampir sempurna

## BAB IV KESIMPULAN DAN SARAN

### 4.1 Kesimpulan

Berdasarkan hasil eksperimen dan analisis perbandingan enam algoritma machine learning untuk deteksi serangan jaringan menggunakan dataset CICIDS2018 :

#### 4.1.1 Performa Model Klasifikasi

Model	Skenario Terbaik	F1-Score	Urutan
Catboost	Imbalance	0.9481	1
Random Forest	Imbalance	0.9470	2
Gradient Boosting	Selection Fitur (k=60)	0.9469	3
KNN	SMOTE + Normalisasi + Selection Fitur	0.9182	4
SVM	SMOTE + Normalisasi + Selection Fitur	0.8724	5
Logistic Regression	SMOTE + Normalisasi + Selection Fitur	0.8654	6

CatBoost menunjukkan performa terbaik dengan F1-score 0.9481, yang dapat dicapai melalui dua pendekatan berbeda, yaitu data Imbalance asli tanpa preprocessing maupun kombinasi standarisasi dan seleksi fitur. Hal ini menunjukkan *robustness* dan *flexibility* dari algoritma CatBoost.

#### 4.1.2 Pengaruh Preprocessing Data

##### 4.1.2.1. SMOTE (Synthetic Minority Over-sampling Technique):

- A. Memberikan dampak positif signifikan pada model linear seperti Logistic Regression dan SVM
- B. Meningkatkan kemampuan deteksi kelas minoritas secara substansial
- C. Tidak memberikan peningkatan pada model tree-based (Random Forest, CatBoost, Gradient Boosting)

#### 4.1.2.2. Normalisasi (Standard Scaler):

- A. Esensial untuk model berbasis gradien dan jarak (Logistic Regression, SVM, KNN)
- B. Tidak memberikan benefit pada model tree-based, bahkan dapat menurunkan performa
- C. Logistic Regression menunjukkan peningkatan dari F1-score 0.2783 menjadi 0.8654 setelah normalisasi

#### 4.1.2.3. Selection Fitur (One-Way ANOVA):

- A. Memberikan peningkatan marginal pada sebagian besar model
- B. Membantu mengurangi dimensi data dan computational cost
- C. Optimal pada k=60-70 fitur untuk dataset CICIDS2018
- D. CatBoost menunjukkan remarkable stability: dapat mempertahankan performa optimal (0.9481) baik dengan maupun tanpa feature selection

#### 4.1.3 Kemampuan Deteksi Kelas minoritas

Semua model menunjukkan kesulitan dalam mendeteksi kelas minoritas ekstrem pada DoS attacks-SlowHTTPTest (label 13) dan FTP-BruteForce (label 14). Model yang menunjukkan kemampuan terbaik untuk kelas minoritas adalah CatBoost dan Logistic Regression.

#### 4.14 Rekomendasi Model

CatBoost adalah pilihan terbaik untuk deployment karena performa tinggi dengan preprocessing yang minimum. Random Forest dapat digunakan sebagai alternatif terbaik kedua dengan interpretasi yang baik. Gradient Boosting berada pada urutan ketiga dengan performa stabil. Jika ingin dilakukan riset, Logistic Regression dapat diteliti lebih dalam karena model tersebut menunjukkan perkembangan terbesar setelah preprosesi.

#### 4.2 Saran

Untuk mengoptimalkan performa model lebih lanjut, penelitian selanjutnya dapat fokus pada optimasi hyperparameter dengan mengimplementasikan *Grid Search* atau *Random Search* untuk *fine-tuning* parameter secara sistematis. Penggunaan *Bayesian Optimization* juga direkomendasikan untuk efisiensi biaya komputasi yang lebih baik, dikombinasikan dengan cross-validation yang lebih kokoh untuk validasi model.

Eksplorasi pendekatan *deep learning* dapat memberi peluang untuk perbandingan komprehensif dengan traditional machine learning. Implementasi LSTM atau CNN dapat mengeksplorasi time-series pattern detection yang lebih mendalam, sedangkan autoencoder dapat dimanfaatkan untuk pendeteksian anomali yang lebih sensitif terhadap pola penyerangan yang tidak dikenal sebelumnya.

Diversifikasi dataset dan *feature engineering* yang lebih mendalam akan memperkuat generalisasi model. Eksplorasi dataset IDS lainnya seperti NSL-KDD, UNSW-NB15, dan CIC-IDS2017 dapat memberikan validasi cross-dataset yang lebih kokoh. Implementasi domain knowledge untuk *feature creation* dan *feature engineering* yang lebih mendalam dapat meningkatkan kemampuan deteksi secara signifikan.

Penelitian ini menunjukkan bahwa pemilihan algoritma dan preprocessing yang tepat sangat crucial untuk efektivitas sistem deteksi serangan siber. CatBoost terbukti sebagai solusi yang *robust* dan praktis, sementara preprocessing yang *comprehensive* dapat meningkatkan performa model yang sebelumnya kurang perform.

## DAFTAR PUSTAKA

- [1] H.-H. Chen, “Understanding Gradient Boosting Classifier: Training, Prediction, and the Role of  $\gamma_j$ ,” 2024.
- [2] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, “CatBoost: unbiased boosting with categorical features.” [Online]. Available: <https://github.com/catboost/catboost>
- [3] L. Breiman, “Random Forests,” 2001.
- [4] T. Evgeniou and M. Pontil, “Support vector machines: Theory and applications,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2049 LNAI, pp. 249–257, 2001, doi: 10.1007/3-540-44673-7\_12.
- [5] T. Mladenova and I. Valova, “Comparative analysis between the traditional K-Nearest Neighbor and Modifications with Weight-Calculation,” *ISMSIT 2022 - 6th International Symposium on Multidisciplinary Studies and Innovative Technologies, Proceedings*, pp. 961–965, 2022, doi: 10.1109/ISMSIT56059.2022.9932693.
- [6] M. K. Chung, “Introduction to logistic regression,” Aug. 2020, Accessed: Jun. 20, 2025. [Online]. Available: <https://arxiv.org/pdf/2008.13567>
- [7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal Of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002, doi: 10.1613/jair.953.
- [8] L. B. V. de Amorim, G. D. C. Cavalcanti, and R. M. O. Cruz, “The choice of scaling technique matters for classification performance,” *Appl Soft Comput*, vol. 133, Jan. 2023, doi: 10.1016/j.asoc.2022.109924.
- [9] J. M. H. Pinheiro *et al.*, “The Impact of Feature Scaling In Machine Learning: Effects on Regression and Classification Tasks,” *IEEE Trans Knowl Data Eng*, vol. XX, p. 1, Jun. 2025, Accessed: Jun. 20, 2025. [Online]. Available: <https://arxiv.org/pdf/2506.08274>
- [10] “(PDF) A Feature Selection Based on One Way Anova For Microarray Data Classification.” Accessed: Jun. 20, 2025. [Online]. Available: [https://www.researchgate.net/publication/328146496\\_A\\_Feature\\_Selection\\_Based\\_on\\_One\\_Way\\_Anova\\_For\\_Microarray\\_Data\\_Classification](https://www.researchgate.net/publication/328146496_A_Feature_Selection_Based_on_One_Way_Anova_For_Microarray_Data_Classification)
- [11] “(PDF) Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation.” Accessed: Jun. 20, 2025. [Online]. Available: [https://www.researchgate.net/publication/276412348\\_Evaluation\\_From\\_precision\\_recall\\_and\\_F-measure\\_to\\_ROC\\_informedness\\_markedness\\_correlation](https://www.researchgate.net/publication/276412348_Evaluation_From_precision_recall_and_F-measure_to_ROC_informedness_markedness_correlation)

- [12] P. Christen, D. J. Hand, and N. Kirielle, "A Review of the F-Measure: Its History, Properties, Criticism, and Alternatives," Mar. 31, 2023, *Association for Computing Machinery*. doi: 10.1145/3606367.
- [13] E. Mortaz, "Imbalance accuracy metric for model selection in multi-class imbalance classification problems," *Knowl Based Syst*, vol. 210, p. 106490, Dec. 2020, doi: 10.1016/J.KNOSYS.2020.106490.