

Tugas Besar 2 IF3170 Intelegensi Artifisial
Semester I tahun 2024/2025
Intelegensi Artifisial
Implementasi Algoritma Pembelajaran Mesin



Oleh:
-duoenjoyer-

Hugo Sabam Augusto (13522129)

Muhammad Zaki (13522136)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

DAFTAR ISI	2
1. Implementasi Algoritma	3
1.1. Implementasi KNN	3
1.2. Implementasi Naive Bayes	3
1.3. Implementasi ID3	3
2. Data Cleaning And Preprocessing	3
2.1. Data Cleaning	3
2.1.1. Handling Missing Data	3
2.1.2. Dealing with Outliers	4
2.1.3. Remove Duplicates	4
2.1.4. Feature Engineering	4
2.2. Preprocessing	7
2.2.1. Feature Scaling	7
2.2.2. Feature Encoding	7
2.2.3. Handling Imbalanced Dataset	8
2.2.4. Dimensionality Reduction	9
2.2.5. Data Normalization	9
3. Analisis	10
3.1. KNN	10
3.2. Naive-Bayes	10
3.3. ID3	11
PEMBAGIAN TUGAS	12
REFERENSI	13
LAMPIRAN	14

1. Implementasi Algoritma

1.1. Implementasi KNN

Algoritma KNN yang diimplementasikan oleh kelompok kami memiliki beberapa tahap, pertama hitung jarak perbedaan dari data yang ingin diuji dengan data yang lama menggunakan beberapa algoritma seperti Euclidean distances, Manhattan distance dan juga minkowski. Lalu setelahnya mengambil data sejumlah K yang memiliki jarak terdekat untuk menentukan label dari data yang baru. Setelah itu kita melihat dari K data tersebut label mana yang banyak muncul, nah itu yang akan kita ambil sebagai label untuk data yang baru.

1.2. Implementasi Naive Bayes

Algoritma naive-bayes yang kami implementasi cukup sederhana yaitu cari berapa banyak jumlah kelas di target (attack_cat). Dilanjutkan dengan menghitung probabilitas awal setiap kelas dan hitung rata-rata nilai fitur untuk masing masing kelas. Ketika prediksi, dibandingkan sampel baru dengan rata rata fitur tiap kelas untuk hitung kemungkinan bahwa sampel baru itu berasal dari kelas apa.

1.3. Implementasi ID3

Implementasi algoritma ID3 yang kami buat cukup sederhana, yaitu dengan menghitung entropy dan information gain dari fitur yang ada. Kemudian, kami memilih information gain terbesar untuk digunakan pada node pohon keputusan. Proses ini diulang secara rekursif untuk membangun pohon keputusan hingga seluruh data terbagi dengan sempurna. Pohon keputusan yang terbentuk nantinya dapat digunakan untuk melakukan prediksi.

2. Data Cleaning And Preprocessing

2.1. Data Cleaning

2.1.1. Handling Missing Data

Sebelum data cleaning, kami pertama melakukan EDA terlebih dahulu yaitu melihat seberapa banyak nilai yang missing / NaN, duplikat data, dan outlier data. Untuk missing data, didapatkan informasi yaitu

mayoritas feature yang ada pada dataset ini memiliki Missing value dengan rata rata 8000 hingga 9000 missing value untuk masing masing feature kecuali feature target yaitu attack_cat

state	8805	ct_src_dport_ltm	8775
dur	8722	ct_dst_sport_ltm	8788
sbytes	8561	ct_dst_src_ltm	8895
dbytes	8869	swin	8740
sttl	8825	dwin	8779
dttl	8654	stcpb	8672
sloss	8794	dtcpb	8803
dloss	8978	smean	8788
service	8791	dmean	8855
sload	8786	trans_depth	8785
dload	8837	response_body_len	8791
spkts	8654	proto	8826
dpkts	8686	sjit	8738
is_sm_ips_ports	8746	djit	8846
ct_state_ttl	8635	sinpkt	8707
ct_flw_http_mthd	8647	dinpkt	8734
is_ftp_login	8647	tcprrt	8836
ct_ftp_cmd	8842	synack	8736
ct_srv_src	8851	ackdat	8595
ct_srv_dst	8774	attack_cat	0
ct_dst_ltm	8738	label	0
ct_src_ltm	8823		

Jadi, solusi kami untuk menghandle missing value ini dengan mengganti semua missing value dengan median untuk feature numerical dan modus untuk feature kategorikal. Kami memilih median untuk fitur numerikal karena dalam EDA kami, mayoritas distribusi feature tidak normal dan cenderung memiliki skewness yang cukup besar.

2.1.2. Dealing with Outliers

Untuk outlier sendiri, didapatkan informasi hampir semua fitur aman dari outlier kecuali fitur 'sbytes' dengan kondisi outlier yang cukup tinggi, visualisasi kami menggunakan box plot. Untuk handling outlier, kami menggunakan metode z-score dengan threshold 3.



2.1.3. Remove Duplicates

Selanjutnya, untuk duplikat data kami melakukan di awal sekali sebelum data splitting, kami melakukan ini agar mempermudah dan konsistensi data ketika preprocessing. Untuk cleaning sebelum split hanyalah **cleaning duplikasi data**. Karena kalau kami clean outlier dan handle missing value di awal sebelum split table kemungkinan terjadi data leakage. Kami mendapatkan jumlah baris yang terduplikat sebanyak 5756 baris.

2.1.4. Feature Engineering

Selanjutnya, kami juga menganalisis mengenai korelasi antar fitur numerikal (karena dalam dataset ini mayoritas fitur bertipe numerik) sehingga kami bisa memilih salah satu fitur dari pasangan fitur yang berkorelasi (kami memilih threshold korelasi > 0.8). Dengan ini dipastikan mengurangi model untuk overfit karena tidak ada pasangan fitur dengan korelasi tinggi yang digunakan ketika preprocessing.

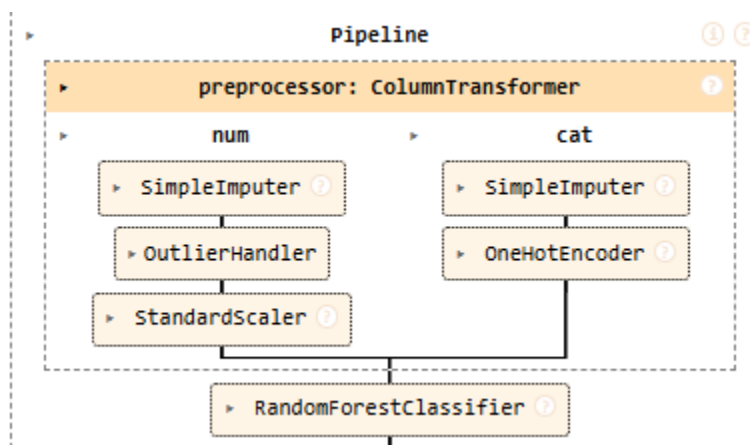
```
Fitur 'sloss' dan 'sbytes' memiliki korelasi: 1.00
Fitur 'dloss' dan 'dbytes' memiliki korelasi: 1.00
Fitur 'spkts' dan 'sbytes' memiliki korelasi: 0.96
Fitur 'spkts' dan 'sloss' memiliki korelasi: 0.97
Fitur 'dpkts' dan 'dbytes' memiliki korelasi: 0.97
Fitur 'dpkts' dan 'dloss' memiliki korelasi: 0.98
Fitur 'ct_ftp_cmd' dan 'is_ftp_login' memiliki korelasi:
1.00
Fitur 'ct_srv_dst' dan 'ct_srv_src' memiliki korelasi:
0.98
Fitur 'ct_dst_ltm' dan 'ct_srv_src' memiliki korelasi:
0.84
Fitur 'ct_dst_ltm' dan 'ct_srv_dst' memiliki korelasi:
0.85
Fitur 'ct_src_ltm' dan 'ct_dst_ltm' memiliki korelasi:
0.89
Fitur 'ct_src_dport_ltm' dan 'ct_srv_src' memiliki
korelasi: 0.87
Fitur 'ct_src_dport_ltm' dan 'ct_srv_dst' memiliki
korelasi: 0.87
Fitur 'ct_src_dport_ltm' dan 'ct_dst_ltm' memiliki
korelasi: 0.96
Fitur 'ct_src_dport_ltm' dan 'ct_src_ltm' memiliki
korelasi: 0.90
Fitur 'ct_dst_sport_ltm' dan 'ct_srv_src' memiliki
```

```

korelasi: 0.82
Fitur 'ct_dst_sport_ltm' dan 'ct_srv_dst' memiliki
korelasi: 0.83
Fitur 'ct_dst_sport_ltm' dan 'ct_dst_ltm' memiliki
korelasi: 0.87
Fitur 'ct_dst_sport_ltm' dan 'ct_src_ltm' memiliki
korelasi: 0.80
Fitur 'ct_dst_sport_ltm' dan 'ct_src_dport_ltm' memiliki
korelasi: 0.91
Fitur 'ct_dst_src_ltm' dan 'ct_srv_src' memiliki
korelasi: 0.97
Fitur 'ct_dst_src_ltm' dan 'ct_srv_dst' memiliki
korelasi: 0.97
Fitur 'ct_dst_src_ltm' dan 'ct_dst_ltm' memiliki
korelasi: 0.85
Fitur 'ct_dst_src_ltm' dan 'ct_src_dport_ltm' memiliki
korelasi: 0.87
Fitur 'ct_dst_src_ltm' dan 'ct_dst_sport_ltm' memiliki
korelasi: 0.84
Fitur 'dwin' dan 'swin' memiliki korelasi: 0.99
Fitur 'tcprrt' dan 'dtl' memiliki korelasi: 0.81
Fitur 'synack' dan 'tcprrt' memiliki korelasi: 0.95
Fitur 'ackdat' dan 'tcprrt' memiliki korelasi: 0.94

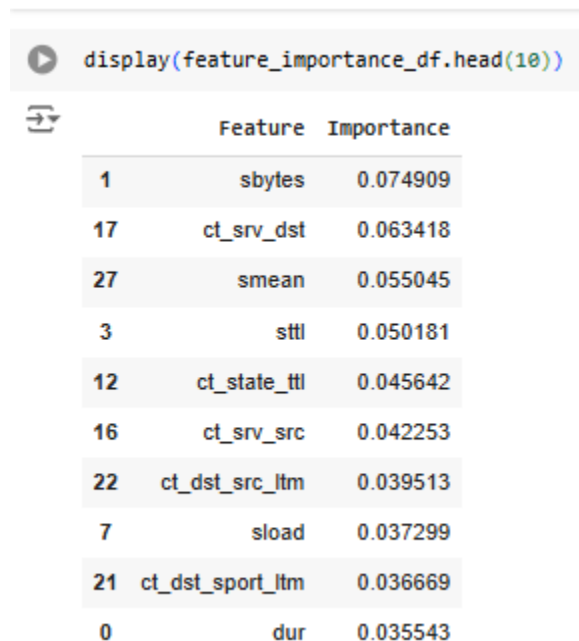
```

Terakhir dalam data cleaning, kami juga melakukan feature engineering berupa feature selection dengan menghitung importance dari setiap feature, karena menggunakan semua feature untuk modelling kemungkinan besar membuat model overfit.



Untuk mendapatkan importance, kami membuat sebuah flow pipeline dengan Imputer, Outlier dan Scaler handler untuk fitur numerik dan Imputer serta One Hot Encoder untuk Kategorikal sebagai fit preprocessing dan menggunakan model algoritma Random Forest

Classifier dari library Scikit untuk mendapatkan data frame importances nya. Berikut hasil 10 feature dengan importance tertinggi



```
display(feature_importance_df.head(10))
```

	Feature	Importance
1	sbytes	0.074909
17	ct_srv_dst	0.063418
27	smean	0.055045
3	sttl	0.050181
12	ct_state_ttl	0.045642
16	ct_srv_src	0.042253
22	ct_dst_src_ltm	0.039513
7	sload	0.037299
21	ct_dst_sport_ltm	0.036669
0	dur	0.035543

Dengan informasi ini, kami bisa memilih fitur apa yang bisa digunakan untuk modeling data dengan algoritma scratch kami, karena algoritma scratch kami cukup berat kalau semua fitur digunakan, memakan waktu yang lama.

2.2. Preprocessing

2.2.1. Feature Scaling

Feature scaling kami gunakan karena dalam algoritma machine learning terutama KNN dan Naive-Bayes, bisa sangat sensitif kalau data untuk training dan test memiliki skala yang tidak tentu dan berbeda beda sehingga tidak optimal. KNN menggunakan jarak euclidean untuk mengukur kedekatan antar titik data. Naive-bayes menghitung probabilitas dengan distribusi data, jadi kalau skala fitur berbeda-beda, fitur dengan nilai yang besar sekali bisa merusak distribusi probabilitas sehingga mempengaruhi predict. Jadi untuk kedua algoritma diatas kami

menggunakan standarisasi dengan library StandardScaler dari Scikit. ID3 tidak memerlukan scaling karena pengukuran based on distribusi data (Tree-based model), bukan skala sensitif seperti jarak atau probabilitas

2.2.2. Feature Encoding

Feature Encoding kami gunakan karena umumnya algoritma machine learning bekerja dalam bentuk numerik, sehingga feature yang berbentuk kategorik harus diubah dahulu bentuknya menjadi angka/binary. Untuk KNN, digunakan One-Hot Encoding dari library Scikit. KNN bekerja dengan metrik distance jadi representasi numerik yang eksplisit sangat membantu dan diperlukan. Naive-Bayes juga menggunakan One-Hot Encoding karena fitur biner membantu menghitung probabilitas kategori menjadi lebih mudah. Untuk ID3 menggunakan Label Encoding karena ID3 merupakan pohon keputusan sehingga bekerja dengan distribusi kategori, tidak perlu kategorial yang eksplisit seperti one hot encoding.

2.2.3. Handling Imbalanced Dataset

attack_cat	
Normal	37389
Generic	24321
Exploits	21739
Fuzzers	12132
DoS	7877
Reconnaissance	6786
Analysis	1341
Backdoor	1177
Shellcode	769
Worms	90

Untuk handling imbalanced dataset, pertama kami testing dengan Undersampling untuk kelas yang dominan dari target yaitu 'Normal'

, 'Generic' , 'Exploits' dan 'Fuzzers' menjadi masing masing 5000. Untuk algoritma KNN dan ID3 (masih menggunakan Scikit),

KNN				
Tanpa Undersample				
KNN Accuracy: 78.50%				
	precision	recall	f1-score	support
Analysis	0.23	0.10	0.14	643
Backdoor	0.16	0.10	0.13	563
DoS	0.33	0.48	0.39	3835
Exploits	0.66	0.68	0.67	10769
Fuzzers	0.67	0.63	0.65	5994
Generic	0.99	0.98	0.98	12064
Normal	0.91	0.90	0.91	18345
Reconnaissance	0.87	0.69	0.77	3347
Shellcode	0.61	0.52	0.56	364
Worms	0.57	0.42	0.49	40
accuracy			0.78	55964
macro avg	0.60	0.55	0.57	55964
weighted avg	0.80	0.78	0.79	55964
Undersampling				
KNN Accuracy: 71.07%				
	precision	recall	f1-score	support
Analysis	0.52	0.20	0.29	683
Backdoor	0.35	0.08	0.13	593
DoS	0.55	0.85	0.67	3900
Exploits	0.56	0.35	0.43	1606
Fuzzers	0.72	0.69	0.70	1648
Generic	0.99	0.97	0.98	1637
Normal	0.88	0.82	0.85	1623
Reconnaissance	0.90	0.76	0.83	3352
Shellcode	0.68	0.73	0.70	371
Worms	0.58	0.60	0.59	42
accuracy			0.71	15455
macro avg	0.67	0.60	0.62	15455
weighted avg	0.72	0.71	0.70	15455

Dari tabel diatas, meskipun akurasi menurun ketika undersampling, informasi F1-score, Precision, dan Recall lebih baik dibanding tidak di undersampling. Kelas minoritas memiliki kenaikan yang cukup besar

dalam presisi (Analysis,Backdoor,Dos). weighted average menunjukkan penurunan setelah undersampling (dari 0.79 menjadi 0.72), namun ini bisa dimaklumi karena penurunan pada kelas mayoritas.

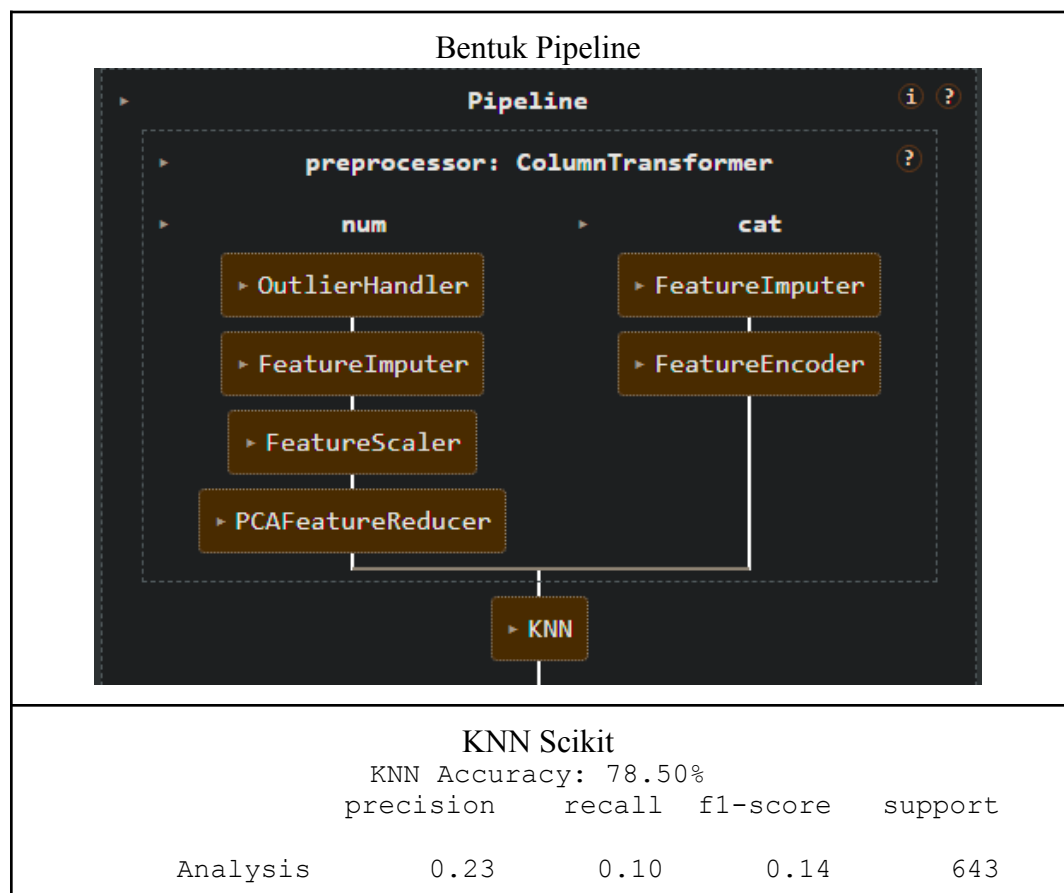
Kalau fokus akurasi, lebih baik tidak usah pakai undersampling.

2.2.4. Dimensionality Reduction

Seperti yang dilihat dari informasi korelasi di pembahasan Data cleaning di [feature engineering](#), dalam dataset ini banyak sekali fitur pasangan numerik yang memiliki korelasi tinggi. Karena itu kami menggunakan PCA dari library Scikit agar dihasilkan dimensi fitur yang lebih sederhana dan tidak ada korelasi.

3. Analisis

3.1. KNN



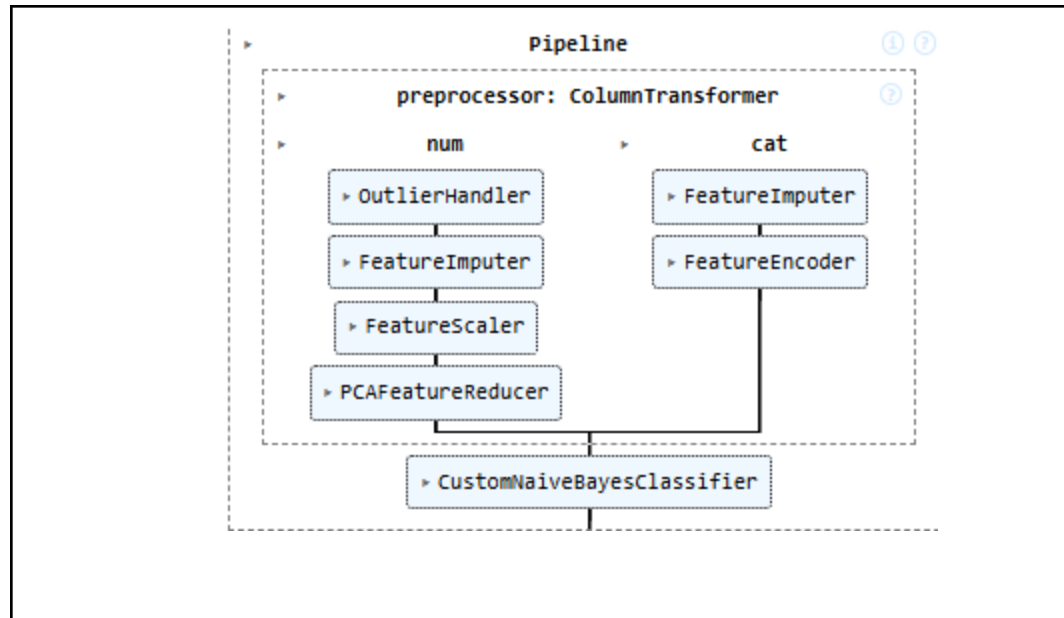
Backdoor	0.16	0.10	0.13	563
DoS	0.33	0.48	0.39	3835
Exploits	0.66	0.68	0.67	10769
Fuzzers	0.67	0.63	0.65	5994
Generic	0.99	0.98	0.98	12064
Normal	0.91	0.90	0.91	18345
Reconnaissance	0.87	0.69	0.77	3347
Shellcode	0.61	0.52	0.56	364
Worms	0.57	0.42	0.49	40
accuracy			0.78	55964
macro avg	0.60	0.55	0.57	55964
weighted avg	0.80	0.78	0.79	55964

KNN From scratch				
KNN Accuracy: 78.75%				
	precision	recall	f1-score	support
Analysis	0.31	0.06	0.10	643
Backdoor	0.29	0.09	0.14	563
DoS	0.33	0.45	0.38	3835
Exploits	0.66	0.69	0.68	10769
Fuzzers	0.67	0.64	0.66	5994
Generic	0.99	0.98	0.98	12064
Normal	0.91	0.90	0.91	18345
Reconnaissance	0.82	0.71	0.76	3347
Shellcode	0.59	0.52	0.55	364
Worms	0.50	0.47	0.49	40
accuracy			0.79	55964
macro avg	0.61	0.55	0.56	55964
weighted avg	0.79	0.79	0.79	55964

Dari tabel penghitungan score diatas implementasi KNN menggunakan scikit dan juga scratch memiliki score accuracy yang tidak jauh berbeda. Hal ini mungkin terjadi karena implementasi dari KNN scratch yang menyerupai dengan KNN library.

3.2. Naive-Bayes

Bentuk Pipeline



Naive-Bayes Scikit

Naive Bayes Accuracy: 11.86%

	precision	recall	f1-score	support
Analysis	0.03	0.13	0.05	643
Backdoor	0.03	0.09	0.05	563
DoS	0.32	0.02	0.03	3835
Exploits	0.88	0.03	0.06	10769
Fuzzers	0.04	0.00	0.00	5994
Generic	0.01	0.00	0.01	12064
Normal	0.28	0.31	0.29	18345
Reconnaissance	0.21	0.01	0.02	3347
Shellcode	0.02	1.00	0.04	364
Worms	0.00	0.80	0.01	40
accuracy			0.12	55964
macro avg	0.18	0.24	0.06	55964
weighted avg	0.30	0.12	0.11	55964

Naive-Bayes Scratch

Naive Bayes Accuracy: 60.61%

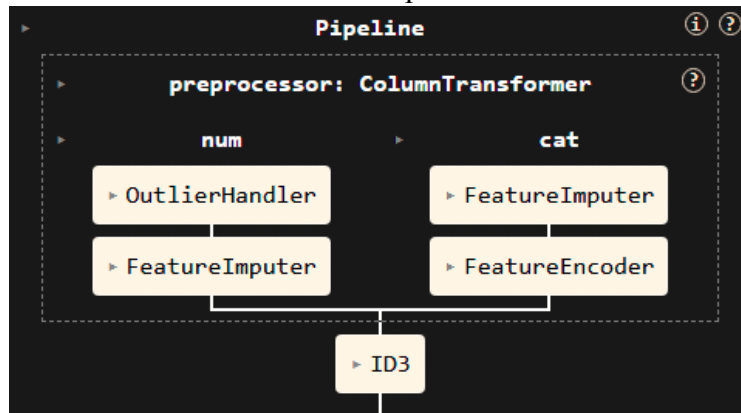
	precision	recall	f1-score	support
Analysis	0.00	0.00	0.00	643
Backdoor	0.00	0.00	0.00	563
DoS	0.27	0.66	0.38	3835
Exploits	0.50	0.13	0.21	10769
Fuzzers	0.37	0.63	0.47	5994
Generic	0.86	0.97	0.92	12064
Normal	0.73	0.79	0.76	18345
Reconnaissance	0.19	0.02	0.03	3347

Shellcode	0.00	0.00	0.00	364
Worms	0.00	0.00	0.00	40
accuracy			0.61	55964
macro avg	0.29	0.32	0.28	55964
weighted avg	0.59	0.61	0.56	55964

Naive-Bayes implementasi dari scratch jauh lebih tinggi dari faktor apapun(Presisi,Accuracy,Recall). Hal ini terjadi karena dalam implementasi Naive-Bayes di scikit memperhitungkan variansi dan dihitung berdasarkan distribusi gaussian untuk tiap fitur sedangkan implementasi kami hanyalah menggunakan mean dari fitur. Hal ini sangatlah mempengaruhi hasil karena dalam EDA, distribusi fitur pada dataset ini tidaklah normal (distribusi non gaussian).

3.3. ID3

Bentuk Pipeline



ID3 Scikit

ID3 (Decision Tree) Accuracy: 80.09%

	precision	recall	f1-score	support
Analysis	0.33	0.10	0.15	643
Backdoor	0.39	0.08	0.14	563
DoS	0.33	0.13	0.18	3835
Exploits	0.60	0.85	0.71	10769
Fuzzers	0.69	0.69	0.69	5994
Generic	0.99	0.98	0.98	12064
Normal	0.92	0.90	0.91	18345
Reconnaissance	0.88	0.72	0.79	3347
Shellcode	0.64	0.54	0.58	364
Worms	0.63	0.60	0.62	40
accuracy			0.80	55964
macro avg	0.64	0.56	0.57	55964
weighted avg	0.79	0.80	0.79	55964

ID3 From scratch

ID3 (Decision Tree) Accuracy: 77.57%

	precision	recall	f1-score	support
Analysis	0.36	0.11	0.17	643
Backdoor	0.43	0.09	0.15	563
DoS	0.32	0.11	0.17	3835
Exploits	0.56	0.75	0.64	10769
Fuzzers	0.69	0.65	0.67	5994
Generic	0.99	0.97	0.98	12064
Normal	0.85	0.90	0.87	18345
Reconnaissance	0.87	0.72	0.79	3347
Shellcode	0.59	0.57	0.58	364
Worms	0.50	0.53	0.51	40
accuracy			0.78	55964
macro avg	0.62	0.54	0.55	55964
weighted avg	0.76	0.78	0.76	55964

Implementasi ID3 dari scikit dan juga scratch memiliki Hasil F1 score yang tidak terlalu jauh mungkin hanya berbeda sekitar 0.0x +-. Keduanya sama-sama sulit untuk menganalisis kelas seperti Analysis, Backdoor, DoS, hal ini bisa dilihat dari hasil keduanya yang relatif buruk yaitu dibawah 0.2

PEMBAGIAN TUGAS

NIM	Tugas
13522129	Semua
13522136	Semua

REFERENSI

- [The UNSW-NB15 Dataset | UNSW Research](#)
- [UNSW-NB15: a comprehensive data set for network intrusion detection systems \(UNSW-NB15 network data set\) | IEEE Conference Publication | IEEE Xplore](#)
- [K-Nearest Neighbor\(KNN\) Algorithm - GeeksforGeeks](#)
- [What Are Naïve Bayes Classifiers? | IBM](#)
- [Decision Trees: ID3 Algorithm Explained | Towards Data Science](#)

LAMPIRAN

Pranala GitHub: https://github.com/mzaki9/InteligensiArtifisial_ML

Pranala Jupyter Notebook:

https://colab.research.google.com/drive/1vmjBiPaGw2YUDUUbps_j-m6NYrIpUr7Q?usp=sharing