BAB I
PENDAHULUAN

1.1 Latar Belakang

Berkembangnya penggunaan internet di era modern ini telah merambah ke berbagai aspek kehidupan. Pada tahun 2022, perkiraan jumlah pengguna internet di seluruh dunia adalah 5,3 miliar, naik dari 4,9 miliar pada tahun sebelumnya (Petrosyan, 2023). Dengan banyaknya pengguna internet tersebut, konektivitas jaringan internet akan semakin luas sehingga resiko terjadinya serangan siber juga akan semakin tinggi. Serangan siber tentunya akan merugikan banyak pihak, baik secara individual atau kelompok seperti perusahaan. Serangan dunia maya dapat merusak reputasi perusahaan yang mengakibatkan hilangnya kepercayaan dari pelanggan, mitra, dan investor. Oleh karena itu, keamanan digital perlu ditingkatkan seiring dengan peningkatan penggunaan internet dari kejahatan siber.

Network Intrusion Detection System (NIDS) memiliki beragam pendekatan yang bertujuan untuk memantau aktivitas jaringan dan mendeteksi atau mengklasifikasikan serangan (Farrukh Y. et al., 2024). Dengan memanfaatkan teknologi *machine learning* (ML), penggunaan menjadi lebih optimal dan efisien dalam mengidentifikasi anomali jaringan. Akan tetapi, penggunaan NIDS membutuhkan banyak sumber daya dan proses komputasi yang besar dalam menganalisis jumlah data yang besar. NIDS bekerja dengan cara menganalisis informasi-informasi teknologi jaringan yang ada, seperti *packet headers*, *packets/transmission*, *packet protocols*, dan informasi jaringan lainnya. Dengan memanfaatkan ML, mempunyai dataset NIDS yang berkualitas dan terbaru menjadi sangat penting. Dikutip dari (Layeghy S. et al., 2024) contoh dataset yang sudah tidak relevan adalah KDD99 yang dibuat lebih dari 20 tahun yang lalu. Dataset ini tidak optimal apabila digunakan sebagai dataset untuk NIDS, karena berkembangnya serangan pada jaringan mengakibatkan perlunya informasi tambahan mengenai trafik jaringan. Dengan menggunakan Zeek, informasi jaringan yang ditangkap dapat diekstrak menjadi sebuah dataset NIDS yang relevan untuk diterapkan dalam pengembangan anomaly detection. Dikutip dari (Ferriyan A., 2022), jumlah fitur dari hasil ekstraksi trafik jaringan menggunakan Zeek adalah sebanyak 86 fitur. Dengan banyaknya fitur tersebut, tentu akan menggunakan sumber daya yang besar dan memiliki komputasi yang tinggi saat melatih sebuah model ML. Permasalahan ini disebut sebagai "*curse of dimensionality*" yaitu ketika data yang besar dan kompleks mengandung banyak informasi tetapi juga meningkatkan kesulitan dalam menggunakannya secara efisien (Jia W. et al., 2022).

Pengembangan pembuatan model ML untuk mendeteksi serangan siber telah mengalami berbagai penelitian. Penelitian yang dilakukan memfokuskan untuk mengurangi fitur yang ada dengan cara melakukan fitur seleksi atau fitur reduksi yang bertujuan untuk mengurangi penggunaan sumber daya mesin, sehingga mempercepat komputasi waktu dan mengurangi penggunaan memori. Penelitian terkait reduksi fitur (Ngo V. et al., 2023) dengan melakukan fitur seleksi dengan menggunakan metode *Correlation Feature* dan fitur reduksi menggunakan metode *Principal Component Analysis* (PCA) menggunakan dataset UNSW-NB15 dengan 41 fitur. Hasil dari penelitian

tersebut dengan fitur seleksi sebanyak 8 hasil dari *Feature Correlation* adalah model Decision Tree dengan akurasi 95.5%, precision 87.41%, recall 86.61%, F1 score 87.46% dan waktu pembuatan model adalah 14.57 detik. Adapun hasil dari fitur reduksi sebanyak 4 hasil dari PCA adalah model KNeighbours adalah precision 86.39%, recall 84.99%, F1 score 85.69% dan waktu pembuatan model adalah 23.41 detik. Dalam penelitian lain (Seth S. et al., 2021) peneliti menggunakan metode Hybrid Feature Selection dengan tujuan mengurangi waktu prediksi tanpa mengurangi informasi dari data serangan yang akan digunakan model dengan menggunakan data CICIDS-2018. Hasil dari penelitiannya yaitu dengan model *Light Gradient Boosting Machine* mendapatkan akurasi 97.73%, sensitivity 96% dengan waktu training model 5.062 detik dan waktu prediksi 0.138 detik.

Tujuan penelitian ini adalah untuk mereduksi fitur jaringan terenkripsi, sehingga menemukan fitur-fitur yang optimal dengan menerapkan metode seleksi fitur, serta mengevaluasi performa algoritma klasifikasi dengan mengukur waktu, penggunaan memori, dan penggunaan CPU selama proses training dan prediksi.

1.2 Rumusan Masalah

Adapun rumusan masalah dari penelitian ini adalah sebagai berikut:

a. Bagaimana mengimplementasikan reduksi fitur pada dataset jaringan terenkripsi yang akan digunakan untuk model klasifikasi?
b. Bagaimana kinerja model terhadap dataset hasil reduksi fitur dalam mendeteksi jaringan berbahaya?

1.3 Tujuan

Berdasarkan permasalahan yang telah dirumuskan sebelumnya, tujuan dari penelitian yang akan dilakukan adalah:

a. Mengimplementasikan metode reduksi fitur pada dataset jaringan terenkripsi.
b. Menganalisis kinerja model klasifikasi terhadap fitur-fitur yang telah direduksi.

1.4 Manfaat

Dengan dilakukannya penelitian ini, diharapkan manfaat yang didapatkan antara lain:

a. Meningkatkan kinerja model klasifikasi dalam mendeteksi jaringan berbahaya, mengurangi waktu dan penggunaan sumber daya.
b. Mengoptimalkan performa model klasifikasi dengan mengurangi jumlah fitur.

1.5 Ruang Lingkup

Adapun ruang lingkup dalam penelitian ini adalah sebagai berikut:

a. Data yang diambil adalah hanya jaringan yang terenskripsi.
b. Pengambilan data dilakukan selama 100 jam.
c. Pengambilan data dilakukan dengan topologi jaringan yang sudah ada di Fakultas Teknik Universitas Hasanuddin.
d. Data ekstraksi dan labeling menggunakan Zeek.

e. Mengevaluasi model klasifikasi dari penggunaan memori, waktu, dan peforma model.

1.6 Tinjauan Pustaka

1.6.1 Fitur Seleksi

Fitur Seleksi adalah teknik dalam ML untuk memilih fitur-fitur penting dari dataset yang dapat mempengaruhi kinerja ML. Teknik ini dapat membantu mengurangi dimensi dataset, sehingga dapat meningkatkan performa klasifikasi algoritma pembelajaran mesin (Fatima M. et al., 2024). Dengan fitur seleksi, fitur-fitur yang tidak relevan dapat dihilangkan, sehingga kinerja model lebih optimal dan menghindari penggunaan komputasi seperti penggunaan *Central Process Unit* (CPU) dan memori. Hal ini mendorong pengembangan NIDS yang efisien agar model ML dapat beroperasi pada perangkat yang terbatas secara optimal (Rbah Y. et al., 2022).

Berbagai metode fitur seleksi seperti *Mutual Information* (MI) (Wang X. et al., 2023), *Random Forest* (RF) (Hasan M. et al., 2016) dan *Analysis of Variance* (ANOVA) (Sheikhan M. et al., 2013) dengan mengambil fitur yang paling relevan sehingga dapat meningkatkan kinerja model ML. Adapun penelitian terkait fitur seleksi yaitu menggunakan *Subset Combination Strategy* (SCS) dengan cara mengambil fitur dari frekuensi kemunculan fitur-fitur yang telah diseleksi sebelumnya (Kshirsagar & Kumar, 2021).

1.6.2 Mutual Information

Mutual Information (MI) adalah ukuran ketidakbergantungan statistik yang mengukur berbagai jenis hubungan antara variabel acak yang menjadikannya efektif dalam menganalisis data yang kompleks. MI memberikan ketahanan terhadap modifikasi data yang tidak memengaruhi hubungan mendasar antara variabel. Dengan kata lain, MI dapat digunakan untuk mengukur seberapa banyak informasi yang satu variabel miliki tentang variabel lainnya tanpa terpengaruh oleh cara kita merepresentasikan data tersebut (Vergara & Estévez, 2014). Dikutip dari Wang X. et al., (2023) MI secara matematis dituliskan seperti:

$$I(R;S) = \sum_{r\in R}\sum_{s\in S} p(r,s)\log\frac{p(r,s)}{p(r)P(s)} \quad (1)$$

Dengan I(R;S) adalah MI antara dua variabel acak diskrit, R dan S adalah contoh data atau variabel, p(r, s) adalah fungsi massa probabilitas gabungan, dan p(r) dan p(s) adalah probabilitas marginal.

1.6.3 Analysis of Variance

Analysis of Variance (ANOVA) adalah metode statistik yang digunakan untuk mengukur perbedaan antara rata-rata kelompok atau kelas dalam suatu dataset yang bertujuan untuk menentukan perbedaan signifikan antara rata-rata kelompok tersebut. ANOVA

dapat mengevaluasi kontribusi setiap fitur terhadap labelnya diberi peringkat berdasarkan relevansinya (Sawyer S., 2009).

$$\sigma^2 = \frac{\sum(\bar{x}_i - \bar{x})^2 n_i}{(k-1)} \quad (2)$$

σ adalah varians antar kelompok, $\bar{x}$ adalah rata-rata keseluruhan dari semua kelompok, $\bar{x}_i$ adalah rata-rata dari kelompok ke-*i*, n_i adalah jumlah data dalam kelompok ke-*i*, dan k adalah total kelompok yang dibandingkan dalam analisis varians.

Skor setiap fitur dihitung dengan merepresentasikan jumlah kemunculan kelas i dalam dataset, rata-rata untuk kelas, rata-rata untuk fitur, dan k menunjukkan jumlah kelas yang ada. Proses ini menghasilkan skor atribut yang diperoleh dari pembagian antara jarak antar kelas dengan jarak di dalam kelas. Nilai yang lebih tinggi menunjukkan bahwa fitur tersebut memiliki relevansi yang lebih besar terhadap kelas yang ada (Siraj M. et al., 2022). Adapun *p-value*, nilai yang mengukur probabilitas bahwa perbedaan ini terjadi secara acak atau tidak. Nilai *p-value* yang rendah (di bawah 0.05) menunjukkan bahwa perbedaan tersebut memiliki signifikansi statistik atau tidak terjadi secara kebetulan (Zhou & Skidmore, 2018).

1.6.4 Subset Combination Strategy

Subset Combination Strategy (SCS) adalah metode yang digunakan dalam pemilihan fitur untuk mengidentifikasi fitur-fitur penting berdasarkan frekuensi kemunculannya dalam beberapa hasil metode fitur seleksi lainnya (Kshirsagar & Kumar, 2021). Dalam penerapannya, SCS menghasilkan beberapa himpunan baru yang menunjukkan kemunculan fitur dalam berbagai kombinasi metode seleksi fitur. Misalnya, SCS-1 mencakup fitur yang muncul setidaknya dalam satu himpunan, SCS-2 mencakup fitur yang muncul di dua himpunan, dan seterusnya. Dengan cara ini, SCS memungkinkan proses seleksi fitur yang lebih terfokus dan relevan, sehingga membantu mengidentifikasi fitur-fitur paling signifikan untuk analisis lebih lanjut dalam model ML.

1.6.5 Machine Learning

Machine learning (ML) merupakan bagian dari bidan ilmu komputer yang bertujuan untuk mengidentifikasi pola dalam data untuk meningkatkan kinerja dalam berbagai aktivitas (Mobarak M. et al., 2023). Algoritma ML dapat berkembang secara otomatis melalui pengalaman yang diperoleh dari data pelatihan yang diberikan untuk membuat prediksi atau keputusan tanpa perlu diprogram secara langsung (Hua T., 2022).

Secara umum, algoritma ML terbagi menjadi tiga jenis, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. *Supervised learning*, yaitu metode ketika dataset memiliki label untuk dilatih sehingga model ML dapat menghasilkan keluaran berupa prediksi. *Unsupervised learning*, yang digunakan ketika data yang dimasukkan dan yang dikelu tidak memiliki label sehingga model ML mempelajari pola data. *Reinforcement learning*, yang berfungsi dengan menetapkan perilaku optimal dalam sistem dan mengambil langkah secara otomatis untuk meningkatkan kinerja sistem tersebut (Akman T. et al., 2023).

1.6.6 Supervised Learning

Supervised learning bertujuan untuk menyediakan platform fleksibel yang mampu menghasilkan prediksi akurat dan mengidentifikasi masalah kompleks, sekaligus mengurangi potensi bias dalam penilaian manusia (Harikrishnakumar R. et al., 2019). *Supervised learning* digunakan untuk mempelajari hubungan antara data yang masuk dan keluar yang sudah terlabel. Adapun algoritma-algoritma klasifikasi, yaitu *eXtreme Gradient Boosting* (XGBoost), *Light Gradient Boosting Machine* (LightGBM), *K-Nearest Neighbors* (KNN), *Random Forest* (RF), *Decision Tree* (DT), *Extra Trees* (ET), dan masih banyak lagi.

1.6.7 Classification

Classification atau klasifikasi dalam *supervised learning* adalah proses ketika model ML dilatih untuk mengelompokkan data yang dimasukkan ke dalam kelas atau kategori tertentu berdasarkan data yang sudah dilabeli (Rojas C. et al., 2023). Model ini mempelajari pola-pola dari data yang sudah diketahui kelasnya untuk kemudian digunakan memprediksi kelas pada data baru yang tidak diketahui labelnya. Dalam proses ini, algoritma berusaha membangun fungsi yang memisahkan kelas-kelas yang ada agar model dapat menempatkan kelas dari data baru yang belum diketahui kelasnya. Adapun contoh dari tugas klasifikasi seperti pengenalan wajah, klasifikasi email sebagai spam atau bukan spam, deteksi penyakit berdasarkan data medis, dan deteksi aktivitas jaringan berbahaya.

1.6.8 K-Nearest Neighbors

Algoritma KNN merupakan metode *supervised learning* dengan hanya menyimpan semua sampel pelatihan tanpa menghitung fungsi spesifik untuk ruang data, sehingga KNN juga disebut sebagai “metode pembelajaran malas” (Wang A. et al., 2023). KNN bekerja dengan cara menemukan tetangga terdekat pada dataset pelatihan berdasarkan jarak tertentu, lalu memberikan label kelas yang paling umum dari tetangga tersebut. KNN juga digunakan untuk mengenali pola di berbagai bidang, seperti pengenalan wajah, analisis graf, dan peramalan deret waktu.

KNN memiliki beberapa keunggulan, konsep yang sederhana, seperti landasan teori yang kuat, beradaptasi secara baik dengan berbagai jenis dataset, dan tidak mengharuskan data mengikuti distribusi tertentu (Abu-Aisheh Z. et al., 2020). KNN juga mudah dikombinasikan dengan algoritma lain dan mudah untuk dimodifikasi untuk menyelesaikan masalah tertentu, seperti penanganan data minoritas, prediksi banyak label, dan melakukan klasterisasi. (Wang A. et al., 2023). Misalkan untuk mencari jarak garis lurus dari dua data tersebut adalah:

$$\mathrm{dist(X,Y)} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2} \quad (3)$$

Dengan $dist(X,Y)$ adalah jarak dua titik garis lurus, x dan y adalah vektor yang mewakili titik vektor dalam ruang dimensi, dan n adalah banyak dimensi .

Jarak dari setiap item dalam set sampel dihitung terlebih dahulu. Dengan melihat k instance terdekat, data baru tersebut akan diklasifikasikan ke dalam kelas yang

memiliki jumlah item terbanyak (Su M., 2011). Adapun *time complexity* dan *space complexity* algoritma KNN adalah:

$$train\ time\ complexity = O(1) \quad (4)$$

$$test\ time\ complexity = O(n \times d) \quad (5)$$

$$space\ complexity = O(n \times d) \quad (6)$$

Dengan *n* adalah banyaknya sample data, dan *d* adalah banyaknya dimensi data (Kumar P., 2021). KNN hanya menyimpan data *training*, sehingga memiliki *train time complexity* O(1).

1.6.9 Decision Tree

Decision Tree (DT) bekerja dengan cara membangun model pelatihan berdasarkan aturan keputusan sederhana dari data pelatihan yang ada. DT mengidentifikasi kategori data uji dengan bergerak dari simpul awal, memeriksa nilai atribut terhadap simpul awal, dan mengikuti cabang yang sesuai hingga ke simpul berikutnya (Sahoo T. et al., 2023).

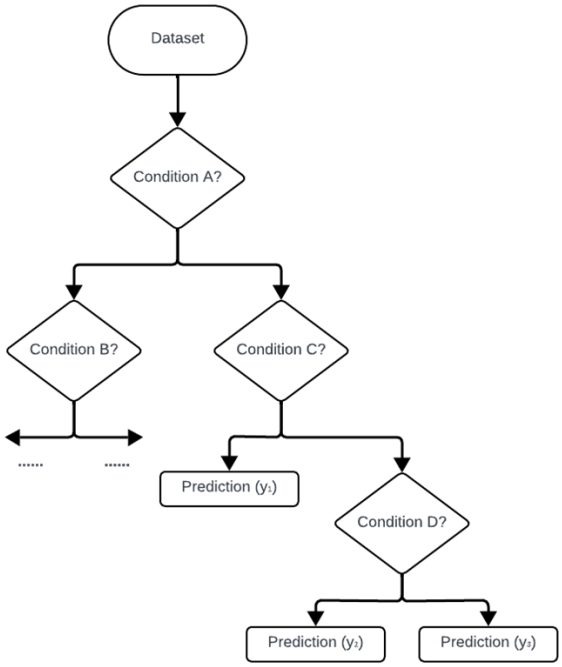
Simpul awal mewakili keseluruhan populasi atau sampel dan dipecah menjadi dua atau lebih kelompok serupa. DT menentukan proses cabang menjadi bagian simpul hingga menjadi simpul daun dan tidak lagi dapat bercabang. Adapun *time complexity* dan *space complexity* adalah sebagai berikut:

$$train\ time\ complexity = O(n \times \log(n) \times d) \quad (7)$$

$$test\ time\ complexity = O(\log(n)) \quad (8)$$

$$space\ complexity = O(n \times d) \quad (9)$$

Dengan *n* adalah banyaknya sample data, dan *d* adalah banyaknya dimensi data (Kumar P., 2021).



Gambar 1. Skema model Decision Tree

DT adalah salah satu metode dalam *supervised learning* yang berarti algoritma ini bekerja dengan data yang sudah diberi label, sehingga model dapat "belajar" dari contoh yang ada untuk membuat prediksi atau klasifikasi pada data baru. DT juga merupakan metode yang banyak digunakan karena mudah dipahami baik dari cara kerja ataupun hasil akhir yang berupa struktur pohon, mudah diinterpretasikan, dan efisien secara komputasi.

1.6.10 Random Forest

RF adalah kumpulan dari pohon-pohon keputusan individu DT yang sering digunakan untuk klasifikasi dan regresi. Simpul akan melakukan pengambilan keputusan dan pembagian data hingga simpul tersebut hanya mengandung satu jenis data. Seperti metode berbasis data lainnya, pohon keputusan mengambil data pelatihan menggunakan proses bagging yang disebut bootstrapping sekitar 2/3 (*in-bag data*) secara acak yang digunakan untuk klasifikasi. Data sisanya yang disebut *out-of-bag* (OOB), digunakan untuk validasi (Behnia P. et al., 2023). Jumlah pohon dalam model ditentukan oleh operator. Peningkatan jumlah pohon akan mengurangi kesalahan OOB hingga mencapai titik stabil.

RF sangat efektif karena sering memiliki akurasi prediksi yang baik, sedikit risiko *overfitting*, dan mudah diinterpretasi. RF juga sering digunakan untuk menentukan fitur yang akan digunakan dengan melihat nilai *feature importance* atau variabel yang

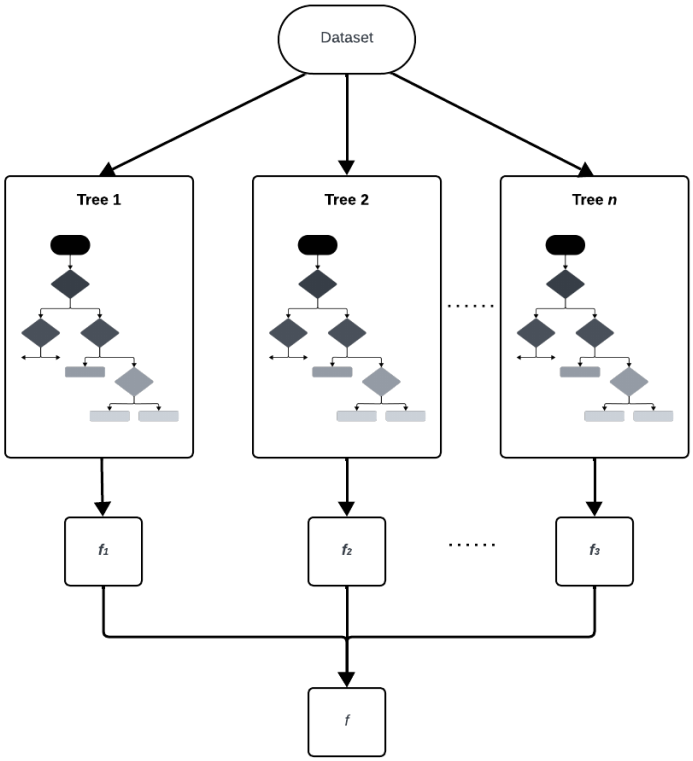
berkontribusi secara baik terhadap model RF. Dengan *time complexity* dan *space complexity* adalah sebagai berikut:

$$train\ time\ complexity = O(k \times n \times d \times \log(n)) \quad (10)$$

$$test\ time\ complexity = O(k \times \log(n)) \quad (11)$$

$$space\ complexity = O(k \times n \times d) \quad (12)$$

Dengan *k* adalah banyaknya pohon keputusan yang dibuat, *n* adalah banyaknya sample data, dan *d* adalah banyaknya dimensi data (Kumar P., 2021).



Gambar 2. Skema model Radom Forest

1.6.11 Network Intrusion Detection System

Network Intrusion Detection Systems (NIDS) merupakan mekanisme untuk menjaga keamanan jaringan dengan cara mendeteksi jaringan yang dianggap melakukan aktivitas berbahaya. NIDS dapat dipasang secara terpusat menyebar di berbagai titik jaringan, dan umumnya ditempatkan setelah firewall untuk memeriksa seluruh jaringan lalu lintas yang masuk (Verma J. et al., 2022). Dengan ini, NIDS dapat mendeteksi pola serangan, perubahan perilaku jaringan, dan membedakan antara aktivitas yang aman dan yang mencurigakan.

NIDS mencakup beberapa teknik utama, yaitu mendeteksi tanda-tanda serangan, mengidentifikasi anomali, dan memantau perubahan perilaku jaringan. Dalam metode berbasis signature-based, NIDS mencari pola serangan yang telah dikenal dan sesuai dengan basis data tanda tangan serangan yang ada, sedangkan dalam metode berbasis

anomali (*anomaly-based*), NIDS menganalisis jaringan lalu lintas untuk mendeteksi perilaku yang tidak biasa atau menyimpang dari pola normal (Assy A. et al., 2023). NIDS berperan penting dalam menjaga keamanan jaringan dengan memberikan peringatan terhadap potensi serangan dan membantu melindungi infrastruktur komputasi dari akses berbahaya.

ML berperan dalam NIDS untuk meningkatkan efektivitas dengan kemampuan analisis dan deteksi otomatis yang lebih adaptif. Dengan algoritma ML, NIDS dapat belajar dari pola dan data jaringan lalu lintas untuk mengenali tanda-tanda serangan yang kompleks dan variatif. Penggunaan ML dalam NIDS memungkinkan sistem untuk mendeteksi aktivitas mencurigakan secara lebih proaktif, tidak hanya berdasarkan pola serangan yang telah dikenal, tetapi juga dengan mengenali anomali atau perilaku yang tidak biasa yang mungkin tidak termasuk dalam basis data serangan.

1.6.12 TCPDUMP

Tcpdump adalah alat analisis jaringan berbasis command line yang digunakan untuk menangkap dan menganalisis lalu lintas jaringan dalam jaringan komputer (Chapman C., 2016). Tcpdump dapat melihat detail setiap paket, seperti alamat IP asal, tujuan, port, dan protokol yang digunakan.

Tcpdump dapat menyimpan tangkapan paket jaringan lalu lintas dalam bentuk format file .pcap yang dapat dianalisis lebih lanjut. File .pcap (Packet Capture) adalah format file yang digunakan untuk menyimpan data tangkapan jaringan lalu lintas yang berisi informasi paket yang ditangkap dari jaringan secara *real time*. *File* .pcap menyimpan data paket dalam format biner yang memungkinkan data untuk disimpan dan dianalisis secara mendalam di kemudian hari (Chapman C., 2016). *File* ini sangat berguna karena berisi informasi lengkap mengenai tiap paket yang melewati jaringan, termasuk *header* dan *payload*, yang dapat digunakan untuk menginvestigasi aktivitas jaringan lalu lintas lebih lanjut.

1.6.13 Zeek Flowmeter

Zeek adalah alat NIDS *open source* yang dirancang untuk memberikan analisis mendalam terhadap paket jaringan lalu lintas menjadi *flow* kemudian menganalisis pola komunikasi yang mencurigakan atau anomali yang mungkin menunjukkan serangan atau pelanggaran keamanan (Waleed A. et al., 2022).

Salah satu fitur utama Zeek adalah kemampuannya untuk mendeteksi anomali dalam lalu lintas jaringan. Zeek dapat mendeteksi anomali dalam jaringan lalu lintas dengan menerapkan aturan dan skrip untuk mengidentifikasi perilaku yang tidak biasa dan membandingkan pola jaringan lalu lintas yang aktual dengan pola yang telah ditentukan sebagai normal (Waleed A. et al., 2022).

1.6.14 Metrics Score

Metrics score berfungsi sebagai ukuran untuk menilai seberapa baik performa model dalam melakukan prediksi, baik pada masalah klasifikasi maupun regresi. Dalam evaluasi model ML klasifikasi, perhitungan *metrics score* dilakukan dengan

menggunakan confusion matrix, yang merangkum kinerja model klasifikasi dengan menunjukkan jumlah prediksi yang benar dan salah untuk masing-masing kelas. Adapun struktur *confusion matrix* adalah sebagai berikut:

	predict positive	**predict negative**
actual positive	True Positive (TP)	False Negative (FN)
actual negative	False Positive (FP)	True Negative (TN)

Gambar 3. Confision matrix

Dengan keterangan TP adalah jumlah prediksi positif benar, TN adalah jumlah prediksi negatif yang benar, FP adalah jumlah prediksi positif yang salah, dan FN adalah jumlah prediksi negatif yang salah.

Metrics score seperti akurasi, presisi, recall, dan *F1-score* dihitung menggunakan nilai-nilai dari *confusion matrix* tersebut. *Confusion matrix* menyediakan data yang diperlukan untuk menghitung metrik evaluasi. Melalui analisis ini, performa model terlihat lebih jelas dan mengambil keputusan untuk mengevaluasi model lebih lanjut.

Accuracy. A*ccuracy* merupakan rasio prediksi benar dari keseluruhan data.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (13)$$

Precision. *Precision* mengukur seberapa banyak prediksi positif yang benar dibandingkan dengan total prediksi positif.

$$Precision = \frac{TP}{TP + FP} \quad (14)$$

Recall. *Recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif.

$$Recall = \frac{TP}{TP + FN} \quad (15)$$

F1-score. Metrik ini menggabungkan *precision* dan *recall* untuk memberikan satu nilai tunggal yang mencerminkan keseimbangan antara keduanya.

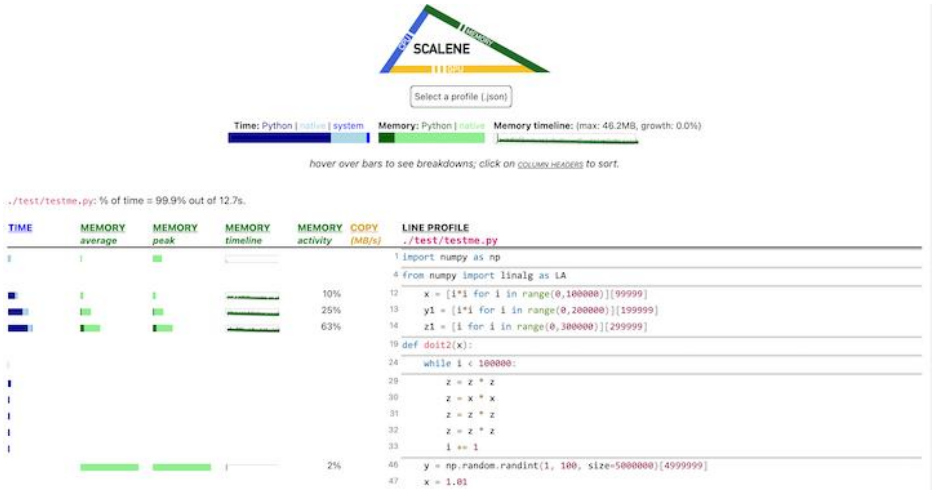
$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (16)$$

1.6.15 Metrics Score

Scalene adalah pustaka dari Python yang dirancang untuk memonitoring performa dari pengunaan sumber daya pada perangkat keras seperti CPU, memori dan lama waktu yang dibutuhkan (Berger E., 2024). Scalene memberikan kinerja yang jauh lebih cepat dengan *overhead* minimal sekitar 10-20%. Keunggulan ini memungkinkan pengembang untuk melakukan pengawasan pada aplikasi tanpa mengganggu kecepatan eksekusi secara signifikan.

Berbeda dari profiler lain yang hanya menyediakan data pada tingkat fungsi, scalene melakukan profiling di tingkat baris kode, sehingga hasil dari analisa lebih spesifik dan optimal. Selain itu, scalene mampu memisahkan penggunaan sumber daya antara kode Python dan kode asli (*native*), sehingga memudahkan pengguna untuk mengidentifikasi masalah performa dari programnya.

Scalene menghasilkan keluaran file dalam format .json yang berisi informasi hasil profiling seperti penggunaan CPU, memori, dan waktu eksekusi untuk bagian kode yang ingin diukur. Selain file .json, scalene juga menyediakan file .html yang memungkinkan pengguna untuk melihat data secara visual dan mendetail, dengan mengambil langsung dari file .json yang dihasilkan. Tampilan interaktif ini membantu pengguna untuk memahami kode dengan lebih mudah.



Gambar 4. Hasil scalene dalam format .html (Berger E., 2024)

1.6.16 Standar Scaler

Standard scaler bertujuan untuk menormalisasi data dengan memusatkannya di sekitar nilai rata-rata (*mean*) dan berdasarkan standar deviasi, sehingga menghasilkan data dengan rata-rata nol dan variansi satu (Siddiqi & Pak, 2021). Secara matematis dapat ditulis seperti:

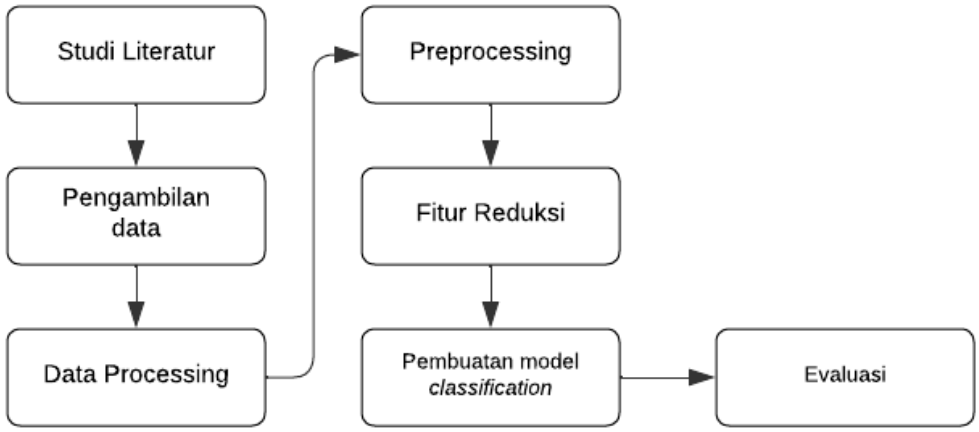
$$x_{scaler} = \frac{x - \mu_{mean}}{s_{std}} \quad (17)$$

x adalah data asli, μ_{mean} adalah rata-rata data, dan s_{std} adalah standar deviasi. Rumus ini memastikan bahwa setiap fitur memiliki nilai rata-rata nol dan standar deviasi satu, sehingga data menjadi lebih cocok untuk banyak algoritma ML yang mengasumsikan input yang terstandarisasi.

BAB II
METODE PENELITIAN

2.1 Tahapan Penelitian

Tahapan-tahapan penelitian akan dilakukan seperti pada gambar 5.



Gambar 5. Alur penelitian

Penjelasan tahapan penelitian secara garis besar yaitu sebagai berikut:

2.1.1 Studi Literatur

Peneliti mencari dan menganalisis buku, jurnal, serta penelitian-penelitian terdahulu yang relevan dengan topik NIDS, fitur reduksi, dan model klasifikasi ML. Kajian ini bertujuan untuk mengumpulkan referensi yang akan menjadi teori dasar dan acuan saat melakukan penelitian.

2.1.2 Pengambilan Data

Penelitian ini akan memanfaatkan topologi jaringan yang sudah ada di Lab Ubiquitos. Peneliti akan menangkap lalu lintas jaringan menggunakan sniffer, dengan bantuan tcpdump. Proses ini akan menghasilkan file dalam format .pcap.

2.1.3 Data Processing

Pada Pada tahap ini, akan dilakukan fitur ektraksi menggunakan Zeek dengan mengubah paket jaringan menjadi flow. Setelah itu, langkah selanjutnya adalah pelabelan dengan mengkategorikan jenis aktivitas, seperti benign atau malicious. Dengan ini, data siap dianalisis lebih lanjut.

2.1.4 Preprocessing

Pada tahap ini, data yang sudah dilabeling akan diolah sebelum digunakan dalam model. Adapun proses yang dilakukan di tahap ini seperti pembersihan data untuk menghapus *noise*, menghapus kolom atau fitur dianggap tidak digunakan, melakukan *encoding* terhadap label dan melakukan *normalization* dengan teknik *standar scaler*.

2.1.5 Fitur Reduksi

Tahap fitur reduksi dalam melibatkan penggunaan teknik fitur seleksi untuk mengurangi jumlah fitur yang digunakan dalam model, tanpa kehilangan informasi penting. Sehingga, kinerja model akan lebih optimal dan meminimalkan risiko overfitting. Adapun teknik fitur seleksi yang akan digunakan yaitu ANOVA, MI, dan RF dengan menganalisis bobot setiap fiturnya yang kemudian melakukan SCS.

2.1.6 Pembuatan Model Klasifikasi

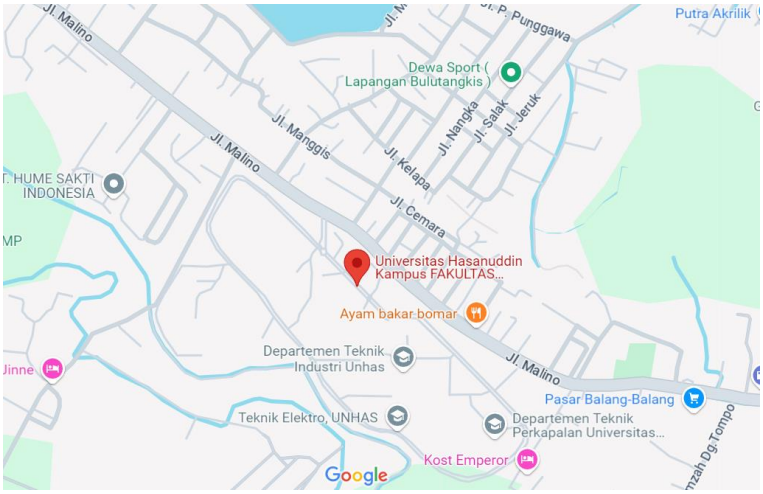
Pada tahap ini, data yang sudah dilakukan fitur reduksi akan dilatih oleh model klasifikasi antara lain K-Nearest Neighbour (KNN), Decision Tree (DT), dan Random Forest Classifier (RFC).

2.1.7 Evaluasi

Tahap ini akan melakukan evaluasi hasil terhadap setiap model yang dilatih dengan menganalisis menghasilkan skor metrik dan performa kinerja, termasuk penggunaan CPU, memori, serta waktu yang dibutuhkan selama proses pelatihan dan saat melakukan prediksi.

2.2 Waktu dan Lokasi Penelitian

Penelitian dilakukan mulai sejak disetujuinya proposal penelitian pada Maret 2024 sampai November 2024. Penulis melakukan penelitian ini di Laboratorium Penelitian ini dilakukan di Laboratorium Ubiquitos Computing, Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin seperti ditunjukkan pada Gambar 6.



Gambar 6. Lokasi penelitian

2.3 Instrumen Penelitian

Instrumen penelitian yang digunakan dalam penelitian ini yaitu:

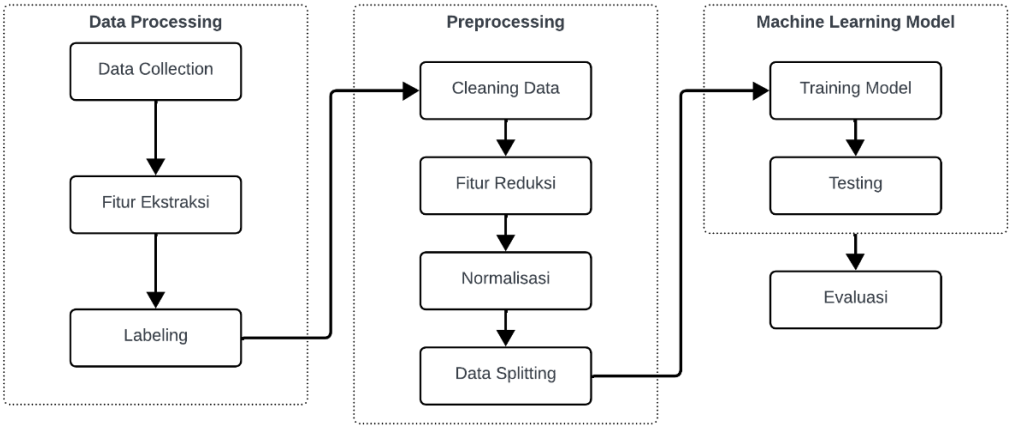
a. Perangkat lunak
   1. Sistem operasi Linux
   2. Python
   3. Scalene
   4. Tcpdump
   5. Zeek

b. Perangkat keras
   1. Laptop Asus ROG Strix G531T
      Laptop yang digunakan memiliki spesifikasi sebagai berikut:
      Sistem Operasi : Windows 11 Home 64-bit
      Layar : 15.6 inci FHD IPS
      Processor : Inter(R) Core(TM) i7-9750H
      RAM : 8 GB
      Storage : 512 SSD
   2. Mini-PC *Victim*
      Sistem Operasi : Linux / Ubuntu LTS 22.0
      GPU : Intel HD
      Processor : Intel Core i5 Gen 8
      RAM : 8 GB
      SSD : 256 GB
   3. Laptom Attacker
      Sistem Operasi : Linux / Ubuntu LTS 22.0
      GPU : Nvidia Geforce CUDA M GT 720 2gb
      Processor : Intel Core i3 Gen 3
      RAM : 16 gb
      SSD : 1 TB

4. Komputer Sniffer
   Sistem Operasi : Linux / Ubuntu LTS 22.0
   GPU : Intel HD
   Processor : Intel Core i5 Gen 8
   RAM : 16 GB
   SSD : 1 TB
   External LAN Card dengan 6 interfaces
5. Router
6. Switch

2.4 Rancangan Sistem

Rancangan sistem bertujuan untuk memberikan ilustrasi mengenai perancangan sistem yang akan dibangun, sekaligus untuk memahami alur proses yang terdapat dalam sistem tersebut.



Gambar 7. Alur rancangan system

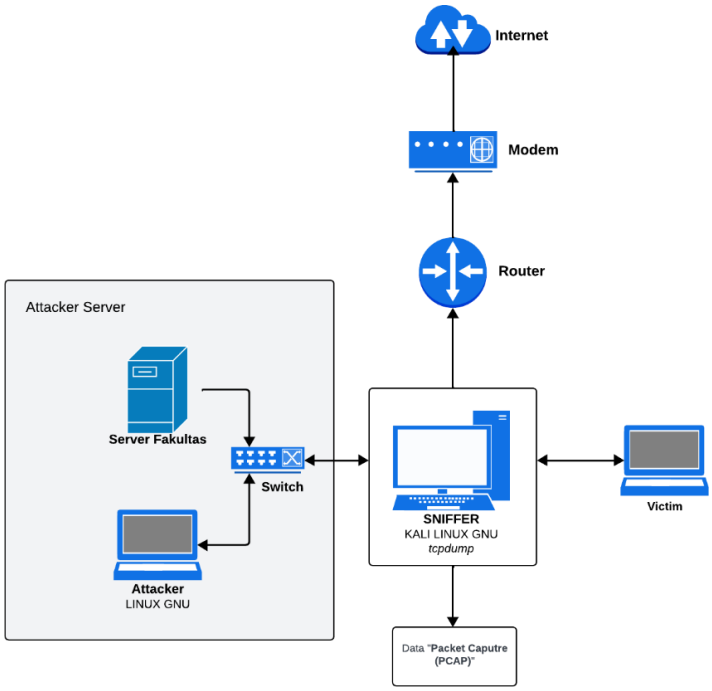
Berdasarkan gambar 7 perancangan sistem terdiri dari beberapa tahapan yang diuraikan sebagai berikut:

2.4.1 Data Processing

Pengambilan data dilakukan secara di Lab Ubiquitos pada tanggal 1 April 2024 – 31 April 2024 menggunakan tcpdump dengan memasang sniffer pada topologi jaringan yang sudah ada.

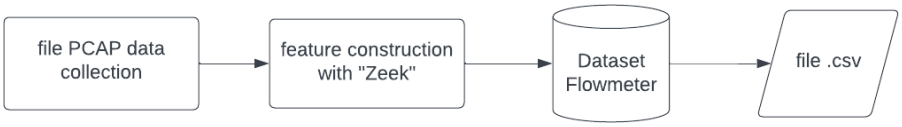
Topologi jaringan. Gambar 8 menunjukkan proses penangkapan data jaringan lalu lintas yang terhubung dengan jaringan lokal di Lab Ubiquitos serta jaringan luar di Fakultas Teknik Universitas Hasanuddin. Dalam penelitian ini, sniffing dilakukan dengan menggunakan tcpdump untuk memantau aktivitas yang dilakukan oleh *victim* dan *attacker* yang akan menghasilkan format file .pcap yang dapat digunakan untuk menganalisis paket-paket yang ada di dalamnya. Adapun filter yang dilakukan yaitu

hanya mengambil jaringan TLS trafik. Alur pengambilan data ini terdiri dari dua tahap utama, yaitu perancangan topologi jaringan dan dumping data.



Gambar 8. Rancangan topologi

Fitur Ekstraksi. Fitur ekstraksi pada tahap ini akan dilakukan dengan Zeek. Data paket jaringan hasil tcpdump diubah menjadi flow dengan menggunakan Zeek. Data yang diperoleh selanjutnya dikonversi menjadi format .csv, sehingga memudahkan dalam mencari pola serangan dan mengidentifikasi jaringan yang berbahaya.



Gambar 9. Tahapan fitur ektraksi menggunakan Zeek

Labeling. Pada tahap ini, data yang telah dikonversi ke format .csv akan dilabel sebagai benign atau malicious agar model klasifikasi dapat dilatih menggunakan data tersebut. Dengan menganalisis file log atau mengidentifikasi alamat IP proxy dari *attacker*. Selain memberikan label biner, peneliti juga akan menetapkan label yang menunjukkan jenis aktivitas yang terjadi dalam data tersebut.

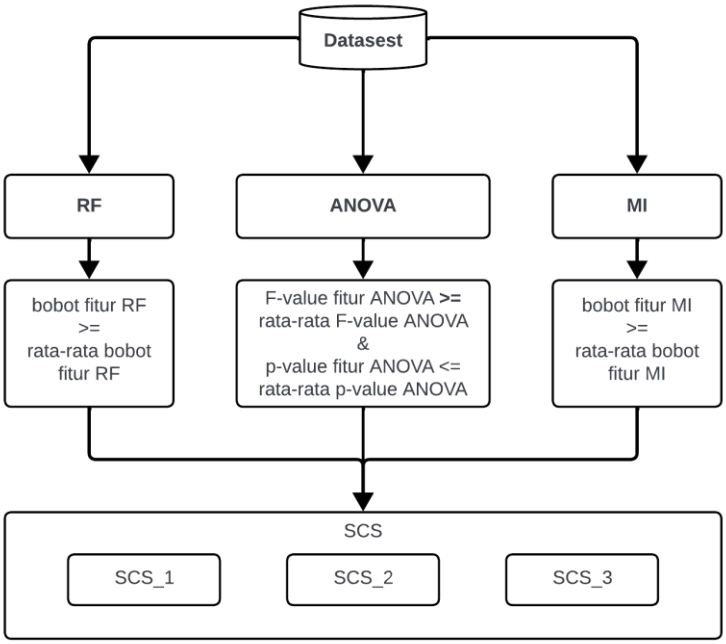
2.4.2 Preprocessing

Data yang sudah diekstraksi dan dilabeli selanjutnya akan di *preprocessing* melalui langkah-langkah berikut:

Cleaning Data. Langkah awal dalam proses ini adalah mengecek dan menangani nilai yang hilang. Jika jumlah data dengan nilai yang hilang sedikit, data tersebut dapat dihapus. Selain itu, perlu juga menghapus data yang dianggap *outliner*.

Langkah selanjutnya yaitu menghilangkan kolom yang dianggap tidak relevan, seperti kolom hasil fitur ekstraksi menggunakan Zeek 'uid', '*Unnamed*: 0', dan '*Unnamed*: 0.1' dengan jumlah nilai unik di setiap baris dihilangkan. Hal tersebut dilakukan karena kolom tersebut tidak memberikan informasi yang bermanfaat untuk dianalisis atau prediksi. Adapun fitur lainnya yang dihapus seperti *'traffic_category*', '*originh*', '*originp*', '*responh*', dan '*responp*' yang memberikan informasi tentang asal dan tujuan koneksi. Hal ini dilakukan karena model klasifikasi difokuskan untuk mempelajari pola perilaku atau aktivitas jaringan yang lebih umum. Sehingga, fitur-fitur tersebut tidak diperlukan agar model klasifikasi dapat lebih tergeneralisasi.

Fitur reduksi. Fitur reduksi yang dilakukan yaitu dengan teknik fitur seleksi. Adapun metode fitur seleksi yang akan digunakan adalah ANOVA, MI, dan RF dengan melihat bobot atau nilai *importance* pada setiap fitur. Seperti pada penelitian Kshirsagar & Kumar, (2021), fitur yang akan dipilih adalah dilihat yang bobotnya lebih dari rata–rata dari setiap hasil dari fitur seleksi tersebut. Dari hasil tersebut, kemudian akan dilakukan SCS. Tiga kelompok fitur baru akan dibuat berdasarkan frekuensi kemunculan fitur pada hasil seleksi fitur sebelumnya, yaitu fitur yang muncul setidaknya satu kali (SCS_1), fitur yang muncul dua kali akan (SCS_2), dan fitur yang muncul di setiap metode seleksi (SCS_3). Berikut gambar 10 untuk langkah-langkah penerapan fitur reduksi.



Gambar 10. Langkah-langkah penerapan fitur reduksi

Normalisasi. Pada tahap ini menormalisasi *dataset* menggunakan *standard scaler* dilakukan dengan mengonversi setiap fitur dalam *dataset* agar memiliki distribusi

dengan rata-rata 0 dan standar deviasi 1. Proses ini memastikan bahwa setiap data berada pada skala yang sama, sehingga kinerja model ML lebih efisien, terutama pada algoritma yang sensitif terhadap skala data seperti KNN.

Data Splitting. Pada tahap ini, *dataset* dibagi menjadi dua bagian, yaitu *train* data untuk melatih model dan *testing* untuk mengevaluasi performa model klasifikasi.

2.4.3 Machine Learning Model

Selanjutnya, untuk mengembangkan model klasifikasi, akan dilakukan pelatihan model dan pengujian model.

Training. Pada tahap ini, model klasifikasi DT, RFC, dan KNN yang akan dilatih menggunakan data latih. Setiap model akan dilatih menggunakan masing-masing kelompok fitur yang dihasilkan dari proses fitur seleksi.

Testing Model. Pada tahap ini, model akan ditest menggunakan data uji dan menghasilkan prediksi yang akan dibandingkan dengan label sebenarnya untuk mengukur akurasi dan performa model.

Dengan menggunakan scalene, setiap model akan dievaluasi kinerjanya saat melakukan *training* dan *testing* dengan melihat penggunaan CPU, memori, dan waktu.

2.4.4 Evaluasi

Evaluasi akan dilakukan dengan menganalisis *metrics score* seperti akurasi, presisi, *recall*, dan *F1-score*. Dengan menganalisis metrik tersebut, model dengan performa terbaik dapat ditinjau lebih dalam. Melalui evaluasi metrik-metrik ini, performa model klasifikasi dapat diukur secara menyeluruh, sehingga memberikan gambaran tentang keakuratan dan efektivitas model dalam melakukan klasifikasi.