**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

# Klasifikasi Tingkat Serangan pada Log Jaringan Siber dengan Komparasi Naive Bayes dan K-Nearest Neighbor

**Evinda Apriliani, Sri Winiarti*, Imam Riadi, Herman Yuliansyah**

Fakultas Teknologi Industri, Program Studi Informatika, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
Email: [1]2200018387@webmail.uad.ac.id, [2,*]sri.winiarti@tif.uad.ac.id, [3]imam.riadi@is.uad.ac.id,
[4]herman.yuliansyah@tif.uad.ac.id
Email Penulis Korespondensi: sri.winiarti@tif.uad.ac.id
Submitted: **21/12/2025**; Accepted: **26/12/2025**; Published**: 26/12/2025**

**Abstrak**−Meningkatnya ancaman keamanan siber menimbulkan dampak signifikan bagi organisasi maupun individu, sehingga diperlukan sistem yang mampu mendeteksi sekaligus mengklasifikasikan tingkat serangan secara akurat untuk mendukung prioritas penanganan. Penelitian ini bertujuan untuk menganalisis dan membandingkan performa dua algoritma *machine learning*, yaitu *Naive Bayes* dan *K-Nearest Neighbor* (KNN), dalam mengklasifikasikan tingkat serangan siber, serta mengevaluasi pengaruh *hyperparameter tuning* terhadap peningkatan akurasi model. Metode penelitian meliputi pemanfaatan dataset *cybersecurity_attacks*, pra-proses data, pelatihan model pada tiga rasio pembagian data (70:30, 80:20, dan 90:10), serta optimisasi parameter menggunakan *Randomized Search* dan *Grid Search*. Evaluasi performa dilakukan berdasarkan nilai akurasi, *precision*, *recall*, dan *f1-score*. Hasil menunjukkan bahwa KNN memberikan performa terbaik dengan akurasi tertinggi 0.96 pada rasio 80:20 setelah proses *tuning*, meningkat dari akurasi sebelum *tuning* sebesar 0.947, disertai nilai *precision*, *recall*, dan *f1-score* pada kisaran 0.95–0.96. Sementara itu, *Naive Bayes* hanya mencapai akurasi tertinggi 0.8485 pada rasio yang sama. Meskipun peningkatan setelah *hyperparameter tuning* tidak terlalu signifikan, proses ini tetap memberikan model yang lebih stabil dan konsisten. Penelitian selanjutnya disarankan mengeksplorasi metode *ensemble* dan pengujian pada dataset lain untuk menghasilkan model klasifikasi serangan siber yang lebih adaptif.

**Kata Kunci:** Naive Bayes; K-Nearest Neighbor; Klasifikasi; Tingkat Serangan Siber; Hyperparameter Tuning

**Abstract**−The increasing threat of cybersecurity poses a significant impact on both organizations and individuals, necessitating a system capable of accurately detecting and classifying attack levels to support prioritization of responses. This study aims to analyze and compare the performance of two machine learning algorithms, Naive Bayes and K-Nearest Neighbor (KNN), in classifying cyberattack levels, and to evaluate the effect of hyperparameter tuning on improving model accuracy. The research methods included utilizing the cybersecurity_attacks dataset, data preprocessing, model training at three data split ratios (70:30, 80:20, and 90:10), and parameter optimization using Randomized Search and Grid Search. Performance evaluation was based on accuracy, precision, recall, and F1-score values. The results showed that KNN performed best, with a peak accuracy of 0.96 at the 80:20 ratio after tuning, increased from an accuracy of 0.947 before tuning, with precision, recall, and F1-score values ranging from 0.95 to 0.96. Meanwhile, Naive Bayes only achieved a peak accuracy of 0.8485 at the same ratio. Although the improvement after hyperparameter tuning was not significant, this process still resulted in a more stable and consistent model. Future research is recommended to explore ensemble methods and test them on other datasets to produce more adaptive cyberattack classification models.

**Keywords**: Naive Bayes; K-Nearest Neighbor; Classification; Cyberattack Level; Hyperparameter Tuning

## 1. PENDAHULUAN

Kemajuan teknologi, salah satunya internet, telah membawa banyak manfaat di banyak aspek kehidupan. Namun, hal ini juga membawa tantangan serius berupa meningkatnya ancaman dan serangan terhadap keamanan siber [1]. Berbagai serangan siber yang terus bermunculan ini menciptakan berbagai dampak negatif yang signifikan, yaitu kerugian finansial, rusaknya reputasi, kehilangan hak kekayaan intelektual, gangguan dalam operasinal, ancaman bahaya bagi keamanan negara, serta dampak psikologis dan emosional pada korbannya [2]. Oleh karena itu, diperlukan strategi yang efektif, seperti pemahaman terhadap risiko yang ada agar dapat memberikan respon yang sesuai [3].

Dalam upaya menangani tantangan tersebut, dibutuhkan solusi yang tidak hanya mampu mengidentifikasi serangan, tetapi juga mengetahui tingkat serangannya. Dengan mengetahui tingkat serangan, kita dapat menentukan prioritas penanganan dan mengatur sumber daya dengan cara yang lebih efektif. Dalam hal ini, machine learning (ML) dapat diterapkan untuk mengidentifikasi pola dari data serangan dan langsung mengklasifikasinnya. Guna menemukan model klasifikasi ML yang optimal, diperlukan perbandingan antar algoritma ML agar mendapatkan pemahaman menyeluruh tentang performa mereka dalam memprediksi tingkat serangan siber dengan performa yang lebih baik [4].

Berbagai algoritma machine learning telah banyak diterapkan dalam penelitian terkait serangan siber. Di antaranya adalah K-Nearest Neighbor (KNN), Random Forest, dan Decision Tree dalam mendeteksi phising website [5], KNN, Naive Bayes, Support Vector Machine, dan Decision Tree untuk mendeteksi malware Android [6], serta Adaboost dan Random Forest pada klasifikasi DDoS [7]. Ini mengindikasikan bahwa algoritma machine learning sangat relevan untuk menangani permasalahan dalam bidang keamanan siber.

Di antara banyaknya algoritma yang dieksplorasi dalam domain serangan siber, Naive Bayes dan K-Nearest Neighbor (KNN) cukup sering digunakan dan dibandingkan dalam berbagai penelitian. Sebagai contoh, Penelitian yang dilakukan oleh K. B. Dasari dan N. Devarakonda [8] menerapkan tujuh algoritma klasifikasi machine learning, yaitu Logistic Regression, Decision Tree, Random Forest, AdaBoost, Gradient Boost, KNN, dan Naive Bayes untuk

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

mendeteksi serangan DDoS pada dataset CIC-DDoS2019. Hasil eksperimen menunjukkan bahwa Logistic Regression, Gradient Boost, dan Naive Bayes menghasilkan performa terbaik dengan akurasi sebesar 99.58%, f1-score tinggi, dan log loss rendah. AdaBoost juga memberikan performa sangat baik dengan recall dan ROC-AUC tertinggi. Sedangkan, KNN memperoleh akurasi tinggi sebesar 99.55%, namun memiliki kelemahan pada waktu eksekusi. Sementara itu, Decision Tree dan Random Forest memiliki performa paling rendah dibandingkan algoritma lainnya.

Penelitian lain oleh J. Naufal Semendawai et al. [9] menerapkan Decision Tree, KNN, dan Naive Bayes untuk mendeteksi serangan Shellcode melalui skema binary classification, menggunakan dataset UNSW-NB15 yang telah diseimbangkan dengan teknik oversampling. Ketiga model diuji pada berbagai rasio data (90:10, 80:20, 70:30, 60:40, 50:50), serta dioptimalkan menggunakan hyperparameter tuning. Didapatkan hasil bahwa Decision Tree dan KNN memberikan performa terbaik dengan akurasi mencapai 97%, sedangkan Naive Bayes memperoleh performa yang jauh lebih rendah dengan akurasi sekitar 63%.

Riset lain oleh Herman et al. [10] berfokus pada pencarian kerentanan SQL Injection di situs web www.bookplusapp.com. Setelah dilakukan analisis awal dengan menggunakan Vega, ditemukan kerentanan SQL Injection dengan tingkat keparahan tinggi terdeteksi di path product_detail.php?id=9. Kerentanan ini memungkinkan penyerang menyuntikkan kode dan mengungkap struktur database. Untuk mengidentifikasi kemungkinan kerentanan ini, penelitian ini membandingkan algoritma K-Nearest Neighbor (KNN) dan Naive Bayes. Model dilatih pada 50 dataset, KNN mencapai akurasi sebesar 94. 2%, sedangkan Naive Bayes 80%. Setelah deteksi, dilakukan penetration testing dengan SQL Map yang mengkonfirmasi temuan dan berhasil mengekstrak informasi dari 23 database. Maka, didapatkan bahwa KNN terbukti lebih akurat dalam mendeteksi kerentanan SQL Injection.

Selain itu, riset oleh D. Hindarto et al. [11] membandingkan performa KNN, Naive Bayes, dan Decision Tree dalam mendeteksi malware pada berkas APK Android. Menggunakan kumpulan data Malware yang diambil dari Reverse Engineering serta mengekstrak fitur Permission dan Intent melalui analisis statis. Dari dataset yang berisi 600 berkas APK Android, diperoleh 1277 fitur. Model dilatih pada split 60:40, 70:30, 80:20, dan 90:10, dengan hasil akurasi setiap model berurutan tiap split yaitu KNN sebesar 84%, 93%, 85%, dan 79%, Naive Bayes sebesar 94%, 98%, 99%, 100%, Decision Tree sebesar 100% pada semua split. Menunjukkan bahwa semua model sudah cukup baik dalam mengidentifikasi malware, meskipun Naive Bayes dan Decision Tree jauh lebih tinggi dari KNN.

A. D. Afifaturahman dan F. Maulana [12] juga mendeteksi anomali lalu lintas jaringan pada dataset Instrusion Detection System (IDS) menggunakan algoritma KNN dan Naive Bayes. Pemodelan dilakukan pada split 60%, 70%, dan 80%. Kedua model memberikan hasil yang baik pada setiap split data, meskipun KNN menunjukkan keunggulan dengan nilai akurasi 99.53% pada split 60%, 99.69% pada split 70%, dan 99.70% pada split 80%. Sedangkan Naive Bayes sebesar 88.55% pada split 60%, 88.43% pada split 70%, dan 88.21% pada split 88.21%. Nilai precision, recall, specifity, dan sensitivity pada KNN juga lebih tinggi daripada Naive Bayes.

Berdasarkan penelitian-penelitian tersebut, dapat disimpulkan bahwa Naive Bayes dan KNN efektif dalam melakukan klasifikasi pada data serangan siber. Meskipun demikian, sebagian besar penelitian sebelumnya hanya membandingkan performa algoritma tanpa hyperparameter tuning. Padahal, hyperparameter tuning sangat berpengaruh dalam meningkatkan akurasi dan stabilitas model, tetapi belum ada penelitian yang membandingkan Naive Bayes dan KNN sekaligus mengeksplorasi hyperparameter tuning dalam klasifikasi tingkat serangan siber.

Penelitian ini bertujuan mengatasi celah tersebut dengan membandingkan performa Naive Bayes dan KNN sekaligus melakukan hyperparameter tuning sehingga menemukan algoritma yang terbaik. Kedua algoritma ini dipilih karena kemampuannya dalam menangani klasifikasi multi-kelas, yang sangat relevan karena dataset yang digunakan, yaitu cybersecurity_attacks, yang merepresentasikan log kejadian siber pada jaringan, dengan tiga kelas label (Low, Medium, High). Naive Bayes, yang menggunakan probabilitas dan asumsi independensi fiturnya, akan menghitung seberapa besar kemungkinan sebuah data masuk ke suatu kelas, lalu memilih kelas dengan kemungkinan tertinggi [13]. Sementara KNN mengklasifikasikan data berdasarkan jarak tetangga terdekatnya, sehingga akan mengambil nilai kelas mayoritas dari tetangganya [14].

Dalam memaksimalkan performa dan mengidentifikasi algoritma terbaik, peningkatan kinerja model akan dilakukan melalui hyperparameter tuning menggunakan teknik Randomized Search dan Grid Search. Dengan demikian, penelitian ini bertujuan untuk menganalisis dan membandingkan performa Naive Bayes dan KNN dalam mengklasifikasikan tingkat serangan siber, serta mengevaluasi pengaruh hyperparameter tuning menggunakan Randomized Search dan Grid Search pada dataset cybersecurity_attacks dengan tiga rasio split data (70:30, 80:20, 90:10) guna menemukan model yang paling optimal untuk kasus ini.

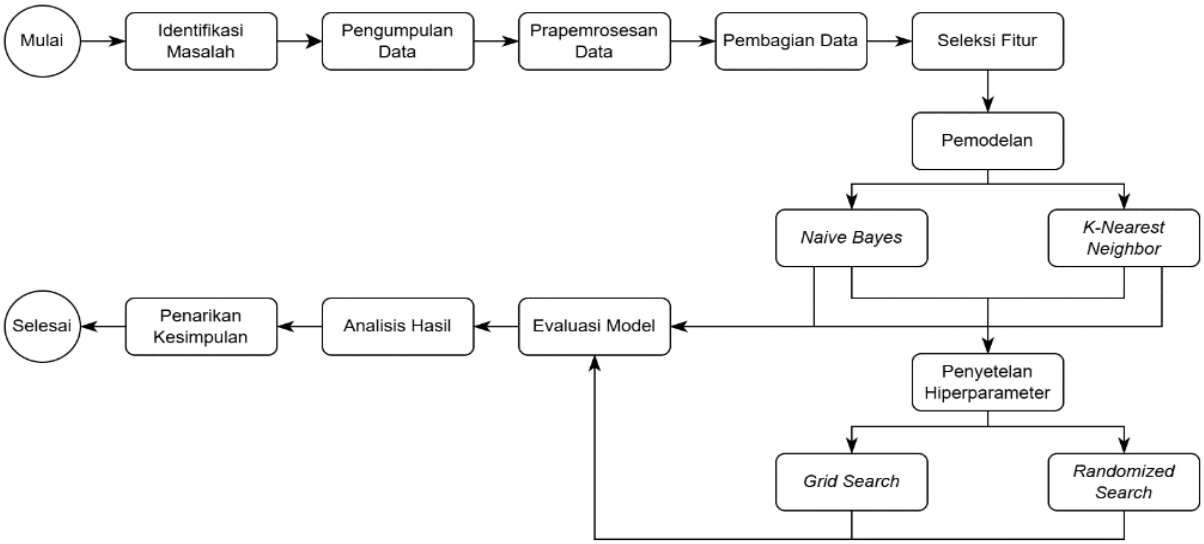
# 2. METODOLOGI PENELITIAN

## 2.1 Tahapan Penelitian

Dalam penelitian ini menggunakan metode metode penelitian eksperimen (experimental research) dengan pendekatan komputasional berbasis *machine learning*, karena seluruh tahapan yang ditunjukkan menggambarkan proses sistematis untuk menguji dan membandingkan performa dua algoritma yang berbeda. Diagram tersebut menampilkan langkah-langkah khas penelitian eksperimen, mulai dari identifikasi masalah, pengumpulan dan prapemrosesan data, pembagian dataset, seleksi fitur, hingga pemodelan menggunakan *Naive Bayes* dan *K-Nearest Neighbor*. Selanjutnya,

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

dilakukan penyetelan hiperparameter dengan *Grid Search* dan *Randomized Search* untuk melihat pengaruh optimasi model terhadap kinerja, diikuti evaluasi model serta analisis hasil untuk menarik kesimpulan. Seluruh proses ini menegaskan bahwa penelitian bertujuan memberikan perlakuan berbeda pada model dan mengukur dampaknya terhadap akurasi, *precision, recall*, dan *F1-score*, sehingga menjadikannya tepat dikategorikan sebagai penelitian eksperimen berbasis *machine learning*. Pada Gambar 1 menunjukkan tahapan penelitian yang diterapkan dalam klasifikasi tingkat serangan ini.




**Gambar 1.** Tahapan Penelitian

a. Identifikasi masalah
Tahap yang sangat fundamental, melibatkan pengamatan dan pemahaman masalah dalam keamanan siber, pengkajian teori dan penelitian terdahulu untuk menerapkan algoritma klasifikasi sebagai solusi, sekaligus perumusan ruang lingkup penelitian.

b. Pengumpulan data
Dataset yang digunakan adalah data serangan siber (*cybersecurity_attacks*) yang bersumber dari platform Kaggle. Dataset sintetis ini terdiri dari 40.000 baris dengan 25 kolom fitur bertipe data campuran, memuat informasi rekaman kejadian serangan siber pada lalu lintas dan aktivitas jaringan, serta memiliki label tingkat keparahan serangan (*severity level*), yaitu *Low*, *Medium*, dan *High*. Pemilihan dataset ini didasarkan pada relevansinya dengan permasalahan klasifikasi serangan siber yang diangkat dalam penelitian. Untuk keperluan eksperimen, digunakan sebagian data dari dataset untuk memastikan fokus dan efisiensi pengujian model, dengan tetap mempertahankan keragaman dan proporsi label aslinya.

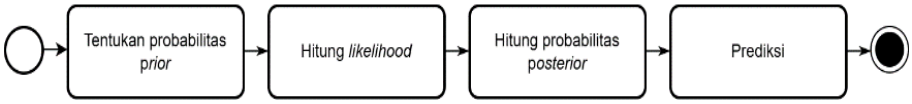
c. Prapemrosesan data
Prapemrosesan data (*data preprocessing*) dilakukan untuk menyiapkan data agar memiliki kualitas yang memadai sebelum digunakan dalam pelatihan model. Proses ini penting untuk menjamin efektivitas dan akurasi model akhir, serta mengatasi masalah mendasar pada data mentah. Tahap awal prapemrosesan adalah pembersihan data (*data cleaning*). Proses ini bertujuan untuk meningkatkan kualitas data, meliputi penanganan *missing values*, perbaikan inkonsistensi format, penghapusan data duplikat, serta eliminasi kolom yang tidak relevan atau tidak digunakan dalam analisis.
Langkah berikutnya adalah pelabelan ulang (*relabelling*). Tahap ini dilakukan karena label asli pada dataset bersifat sintetis dan tidak merefleksikan tingkat serangan yang sesungguhnya. Oleh karena itu, pelabelan ulang bertujuan untuk menyesuaikan dan memperbaiki klasifikasi data agar lebih relevan dengan tujuan penelitian. Sebagai contoh, sampel data yang semula berlabel '*Low*' diubah menjadi '*High*' setelah melalui evaluasi mendalam yang ternyata menunjukkan tingkat serangan aktual yang lebih tinggi. Proses sistematis ini memastikan setiap sampel data memiliki label yang konsisten dan bermakna untuk pelatihan model.
Tahap terakhir adalah transformasi data, yang dilakukan untuk memodifikasi nilai-nilai data agar sesuai dengan persyaratan algoritma *machine learning*, sehingga model dapat belajar dengan lebih efektif dan akurat. Transformasi yang umum diterapkan meliputi *encoding* data kategorikal menjadi representasi numerik, serta normalisasi atau standardisasi untuk menyamakan skala data numerik.

d. Pembagian data
Proses membagi dataset menjadi dua bagian, data latih untuk melatih model, dan data uji untuk menguji seberapa efektif model dalam memprediksi data yang belum pernah dipelajari. Ini dilakukan dengan berbagai rasio *split*, 70:30, 80:20, atau 90:10. Pemilihan rasio ini bertujuan agar mendapatkan akurasi yang terbaik, dengan memastikan bahwa model yang dibangun tidak sekadar mengingat data pelatihan (*overfitting*), tetapi juga dapat menggeneralisasi data baru secara efektif.

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

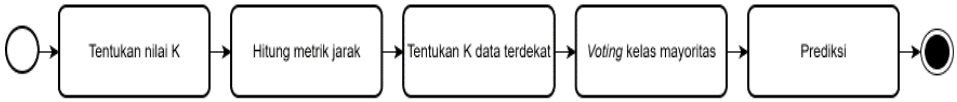
e. Seleksi Fitur menggunakan *SelectKBest*
Seleksi fitur bertujuan untuk menentukan sejumlah fitur terbaik yang paling relevan dengan variabel target. Proses ini membantu mengurangi ukuran data, yang tidak hanya mempercepat pelatihan model, tetapi juga meningkatkan akurasi dan mencegah *overfitting* karena model tidak dilatih dengan fitur-fitur yang penting. Metode seleksi fitur yang akan digunakan yaitu *SelectKBest*. *SelectKBest* adalah metode pemilihan fitur yang bertujuan untuk meningkatkan performa dan akurasi model pada dataset besar [15]. Metode ini menganalisis hubungan antara fitur dan variabel target, memilih k fitur dengan nilai tertinggi. Pada penelitian ini, nilai k fitur terbaik ditetapkan sebanyak 10. Dengan fungsi *f_classif*, *SelectKBest* dapat mengidentifikasi fitur penting secara statistik, sehingga mengurangi jumlah fitur tanpa kehilangan informasi. Ini dapat mempercepat pemrosesan dan menurunkan risiko *overfitting* [16].

f. Penerapan algoritma *Naive Bayes*
Selanjutnya, model *Naive Bayes* akan mempelajari pola dari data *training* untuk membuat prediksi. *Naive Bayes* adalah teknik klasifikasi berlandaskan *Teorema Bayes* dengan pendekatan probabilistik [17]. Model ini bekerja dengan menghitung probabilitas *prior* dan *likelihood*, yang kemudian digunakan untuk menghitung probabilitas *posterior* dengan menerapkan rumus *Teorema Bayes* untuk menentukan kelas dengan peluang tertinggi [18]. Penerapan Algoritma *bayes* ditunjukkan pada Gambar 2.




**Gambar 2.** Tahapan Penerapan Algoritma *Naive Bayes*

g. Penerapan algoritma *K-Nearest Neighbor*
Langkah selanjutnya membuat model *K-Nearest Neighbor* (KNN) akan dilatih menggunakan data *training* untuk membuat prediksi. KNN adalah metode klasifikasi yang mengkategorikan data baru berdasarkan kelas mayoritas dari k tetangga terdekat, di mana suatu data uji akan dikategorikan ke kelas yang paling banyak muncul di antara k data pelatihan yang memiliki jarak terdekat dengannya [21]. Algoritma KNN ditunjukkan pada Gambar 3.




**Gambar 3.** Tahapan Penerapan Algoritma *K-Nearest Neighbor*

h. Penyetelan Hiperparameter
Penyetelan hiperparameter atau *hyperparameter tuning* adalah proses mengatur *hyperparameter*, yaitu konfigurasi model yang tidak dipelajari dari data dan harus ditentukan sebelum pelatihan, seperti nilai k pada KNN. Konfigurasi ini memengaruhi cara model mempelajari pola dari data, sehingga pemilihan nilai yang tepat sangat penting agar model dapat mencapai kinerja optimal, meminimalkan kesalahan, dan tetap mampu melakukan generalisasi pada data baru [25]. Untuk melakukan ini, akan digunakan dua teknik, yaitu *Randomized Search*, yang secara acak memilih kombinasi *hyperparameter* dari distribusi yang ditentukan dan sering kali lebih efisien secara komputasi untuk ruang pencarian yang luas, serta *Grid Search*, yang secara sistematis mencoba semua kombinasi *hyperparameter* yang telah ditentukan dalam ruang pencarian [26].

i. Evaluasi model
Setelah model selesai dilatih, dilakukan evaluasi untuk mengukur kinerja model dalam memprediksi data baru. Evaluasi dilakukan dengan beberapa metrik klasifikasi seperti akurasi (persentase prediksi yang tepat), *precision* (proporsi data positif yang diprediksi dengan tepat), *recall* (jumlah data positif yang berhasil terdeteksi), *f1-score* (gabungan *precision* dan *recall*) [27]. Untuk mendapatkan gambaran performa yang menyeluruh, digunakan juga *confusion matrix* [15], yang akan menilai kinerja model dengan empat hasil prediksi, yaitu *True Positive, False Negative, False Positive,* dan *True Negative* [28]. Hasil dari evaluasi ini memberikan gambaran menyeluruh tentang kemampuan model dalam mengenali pola dan menghasilkan prediksi yang tepat sebelum dilakukan analisis efektivitas lebih lanjut.

j. Analisis hasil
Tahap ini membahas hasil evaluasi model, baik dari kinerja awal maupun setelah dilakukan proses *hyperparameter tuning*. Analisis mencakup perbandingan performa setiap algoritma berdasarkan metrik evaluasi yang digunakan. Dari hasil tersebut dapat terlihat bahwa algoritma tertentu menunjukkan peningkatan kinerja yang lebih signifikan, sehingga mengindikasikan kemampuan model yang lebih baik dalam menangani permasalahan yang diteliti.

k. Penarikan kesimpulan
Tahap terakhir adalah menarik kesimpulan dari analisis yang telah dilakukan. Fokus kesimpulan adalah menentukan algoritma yang memberikan performa terbaik dalam penelitian ini. Dengan mempertimbangkan hasil evaluasi dan pengaruh *hyperparameter tuning*, algoritma dengan nilai metrik tertinggi akan dianggap sebagai pilihan paling tepat, karena mampu memberikan prediksi yang lebih akurat dan konsisten.

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

### 2.2 Algoritma *Naive Bayes*

*Naive Bayes* adalah teknik klasifikasi berlandaskan *Teorema Bayes* dengan pendekatan probabilistik. *Teorema Bayes* akan memprediksi kemungkinan berdasarkan data yang telah ada dengan asumsi independensi antar atribut [17]. Algoritma ini menghitung probabilitas bersyarat suatu kelas berdasarkan nilai atribut dengan memperkirakan frekuensi kemunculan atribut di setiap kelas, kemudian menerapkan rumus *Teorema Bayes* untuk menentukan kelas dengan peluang tertinggi. Meskipun sederhana, *Naive Bayes* dikenal efisien dan cepat dalam menangani jumlah data besar [18].

Model ini bekerja dengan menghitung probabilitas *prior* dan *likelihood*, yang kemudian digunakan untuk memperoleh probabilitas *posterior*, di mana kelas dengan probabilitas tertinggi akan menjadi hasil prediksi. Hubungan antar probabilitas tersebut dapat dijelaskan melalui rumus berikut [19]:
Rumus probabilitas *posterior*:

$$P(C|F_1 \dots F_n) = \frac{P(C)\cdot P(F_1 \dots F_n|C)}{P(F_1 \dots F_n)} \quad (1)$$

Rumus probabilitas *prior*

$$P(C) \quad (2)$$

Rumus *likelihood*:

$$P(F_1 \dots F_n|C) \quad (3)$$

Di mana $P(C|F_1 \dots F_n)$ menghitung probabilitas suatu data masuk ke kelas $C$ setelah mempertimbangkan fitur-fiturnya. Nilai ini diperoleh dengan mengalikan probabilitas awal kelas ($P(C)$) dengan probabilitas kemunculan fitur-fitur tersebut pada kelas tersebut ($P(F_1 \dots F_n|C)$), kemudian membaginya dengan *evidence* ($P(F_1 \dots F_n)$) yang merepresentasikan probabilitas munculnya fitur-fitur tersebut secara keseluruhan. Adapun *evidence* pada kasus ini dapat diabaikan karena nilainya sama untuk semua kelas sehingga tidak memengaruhi proses pemilihan kelas dengan probabilitas *posterior* tertinggi [19].

### 2.3 Algoritma *K-Nearest Neighbor*

KNN adalah metode klasifikasi *supervised learning*, karena proses klasifikasinya bergantung pada data pelatihan yang telah memiliki label kelas [20]. Proses penerapannya diawali dengan menentukan parameter k (jumlah tetangga) yang sesuai dengan karakteristik dataset, di mana nilai k minimal 1 dan maksimal sebanyak jumlah data pelatihan. Setelah itu, dilakukan perhitungan jarak antara data uji dan data pelatihan dengan beberapa metrik, antara lain *Euclidean*, *Manhattan*, dan *Minkowski distance* [22]. Rumus *Euclidean distance* dirumuskan sebagai berikut [23]:

$$d_{(x,y)} = \sqrt{\Sigma_{i=1}^{n}(x_1 - y_1)^2} \quad (4)$$

Sedangkan *Manhattan distance* dinyatakan sebagai [23]:

$$d_{(x,y)} = \sum_{i=1}^{n}|x_i - y_i| \quad (5)$$

di mana $x_1$ merujuk pada sampel data, $y_1$ merupakan data pengujian, dan $d_{(x,y)}$ menunjukkan jarak yang ada antara $x_1$ dan $y_1$.

Setelah semua jarak dihitung, jarak-jarak tersebut diurutkan dari yang paling kecil hingga paling besar untuk menemukan k tetangga terdekat. Hasil *voting* mayoritas dari tetangga-tetangga terdekat tersebut yang akan menentukan kelas untuk data baru [22]. Meskipun tidak perlu proses pelatihan yang kompleks karena hanya menyimpan data pelatihan sebagai referensi, pemilihan nilai k sangat berpengaruh terhadap performa model, nilai k yang terlalu kecil dapat menyebabkan model sensitif terhadap *noise*, sedangkan nilai k yang terlalu besar dapat menimbulkan bias sehingga menurunkan akurasi prediksi [4], [24].

## 3. HASIL DAN PEMBAHASAN

### 3.1 Persiapan Data

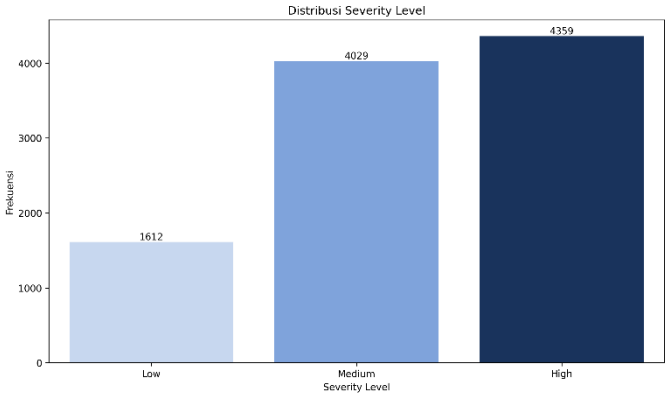
Dataset *cybersecurity_attacks* dipersiapkan melalui tahapan prapemrosesan data. Tahapan ini dimulai dengan proses pembersihan data. Dari hasil pengecekan *missing values*, ditemukan sejumlah nilai kosong pada kolom kategorikal, yang kemudian diimputasi dengan label '*Unknown*' untuk merepresentasikan data yang tidak tersedia, tidak terdeteksi, atau tidak tercatat dalam rekaman trafik jaringan serangan siber. Dengan demikian, pemberian label '*Unknown*' memungkinkan model untuk mengenali dan mengolah ketidakhadiran informasi sebagai sinyal potensial dalam pola serangan.

Selanjutnya, kolom yang tidak relevan terhadap analisis klasifikasi dihapus untuk menyederhanakan dataset dan memfokuskan pada kolom fitur yang lebih mudah diolah. Selain itu, meskipun beberapa fitur tampak tidak berkorelasi langsung dengan target, sebagian tetap dipertahankan karena dinilai masih berkontribusi pada model dan memudahkan proses transformasi data. Hasilnya, sebanyak 8 kolom dihapus dan tersisa 17 kolom untuk proses

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

selanjutnya. Dengan kolom-kolom numerik yaitu '*Source Port*', '*Destination Port*', '*Packet Length*', dan '*Anomaly Scores*', serta kolom-kolom kategorikal yaitu '*Protocol*', '*Packet Type*', '*Traffic Type*', '*Malware Indicators*', '*Alerts/Warnings*', '*Attack Type*', '*Attack Signature*', '*Network Segment*', '*Firewall Logs*', '*IDS/IPS Alerts*', '*Log Source*', '*Action Taken*', dan '*Severity Level*'.

Tahap berikutnya adalah pelabelan ulang (*relabelling*). Label asli pada dataset bersifat sintetis dan tidak secara eksplisit merepresentasikan tingkat serangan. Oleh karena itu, dibuat beberapa aturan pelabelan baru secara manual dengan mengacu pada informasi dan referensi yang diperoleh dari berbagai sumber, dengan tetap menggunakan label *Low*, *Medium*, dan *High*. Salah satu aturannya adalah menggunakan nilai '*Anomaly Scores*', di mana data dengan skor 0–50 diberi label *Low*, 51–75 diberi label *Medium*, dan 76-100 diberi label *High*, pengelompokkan rentang ini mengikuti distribusi persentil dalam dataset dan didasarkan pada prinsip bahwa semakin tinggi skor anomali, maka semakin menandakan terjadinya peristiwa abnormal atau kritis [29]. Setelah proses pelabelan selesai, didapatkan distribusi data pada setiap kelas, yaitu 1612 untuk *Low*, 4029 untuk *Medium*, serta 4359 untuk *High*. Visualisasi distribusi data setiap kelas tersebut ditampilkan pada Gambar 4.




**Gambar 4.** Distribusi Data Setelah *Relabelling*

Distribusi tersebut menunjukkan bahwa data menjadi tidak seimbang setelah proses *relabeling*, di mana kelas *Low* memiliki jumlah sampel yang paling sedikit dibandingkan dengan kelas lainnya. Ketidakseimbangan ini dianggap sebagai karateristik alami dari hasil pelabelan, sehingga tidak dilakukan upaya *balancing*. Karena selain dapat mengubah distribusi kelas yang terbentuk setelah *relabelling*, pengujian awal menunjukkan bahwa penerapan *balancing* tidak memberikan peningkatan performa secara signifikan. Model *Naive Bayes* bahkan mengalami penurunan performa yang besar di semua kelas setelah *balancing*, dan model KNN juga mengalami penurunan performa.

Proses berikutnya adalah *encoding* pada fitur kategorikal agar seluruh fitur dapat diolah oleh model *machine learning*. *Encoding* dilakukan menggunakan metode One Hot *Encoding* pada kolom '*Protocol*', '*Packet Type*', '*Traffic Type*', '*Malware Indicators*', '*Alerts/Warnings*', '*Attack Type*', '*Attack Signature*', '*Network Segment*', '*Firewall Logs*', '*IDS/IPS Alerts*', '*Log Source*', serta Label *Encoding* pada kolom '*Action Taken*' dan '*Severity Level*'. Setelah *encoding* selesai, semua data telah berada dalam format numerik yang sesuai dengan kebutuhan model klasifikasi.

Setelah semua fitur siap, dataset dibagi ke dalam tiga rasio, 70:30, 80:20, dan 90:10, untuk keperluan pelatihan dan evaluasi model. Seleksi fitur kemudian dilakukan pada data latih di setiap skenario rasio menggunakan metode *SelectKBest* untuk menghindari data *leakage* dan memastikan evaluasi model yang valid. Seleksi ini bertujuan mengurangi redundansi dan meningkatkan efisiensi pelatihan dengan memilih 10 fitur terbaik berdasarkan skor statistiknya terhadap label target (*Severity Level*). Setiap rasio pembagian data latih dan data uji menghasilkan kombinasi fitur terpilih yang berbeda dikarenakan proses seleksi fitur dilakukan secara terpisah pada setiap skenario pembagian data, namun jumlah fitur yang digunakan tetap sama, yaitu 10 fitur, sebagaimana ditunjukkan pada Tabel 1.

**Tabel 1.** Fitur Terpilih *SelectKBest*

| Rasio Data | Fitur Terpilih |
|---|---|
| 70:30 | '*Packet Length*', '*Anomaly Scores*', '*Protocol_UDP*', '*Malware Indicators_Unknown*', '*Alerts/Warnings_Unknown*', '*Attack Type_Intrusion*', '*Attack Type_Malware*', '*Attack Signature_Known Pattern B*', '*Network Segment_Segment B*', '*Action Taken Encoded*' |
| 80:20 | '*Packet Length*', '*Anomaly Scores*', '*Protocol_UDP*', '*Malware Indicators_Unknown*', '*Alerts/Warnings_Unknown*', '*Attack Type_Intrusion*', '*Attack Type_Malware*', '*Network Segment_Segment B*', '*Log Source_Server*', '*Action Taken Encoded*' |
| 90:10 | '*Packet Length*', '*Anomaly Scores*', '*Protocol_TCP*', '*Protocol_UDP*', '*Malware Indicators_Unknown*', '*Alerts/Warnings_Unknown*', '*Attack Type_Intrusion*', '*Attack Type_Malware*', '*Log Source_Server*', '*Action Taken Encoded*' |

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

Setelah fitur-fitur terbaik dipilih melalui metode *SelectKBest*, langkah selanjutnya adalah melakukan standardisasi data pada fitur-fitur tersebut. Namun, proses ini secara spesifik hanya diterapkan pada model KNN karena algoritma ini sangat peka terhadap perbedaan skala antar variabel yang dapat memengaruhi perhitungan jarak antar data. Standardisasi dilakukan menggunakan metode *Standard Scaler* agar seluruh fitur memiliki distribusi yang lebih merata dengan nilai rata-rata 0 dan standar deviasi 1. Tahap ini dilakukan setelah seleksi fitur untuk menjaga efisiensi komputasi dan memastikan bahwa hanya fitur-fitur yang benar-benar relevanlah yang melalui proses standardisasi tersebut.

Seluruh tahapan prapemrosesan data menghasilkan dataset yang bersih, lengkap, dan terstruktur dengan baik. Dataset ini kini terdiri atas 10 fitur utama dan tiga kelas target (*Low*, *Medium*, *High*), dan berada dalam format yang sepenuhnya siap digunakan untuk melatih model pada tahap selanjutnya.

### 3.2 Pemodelan dan Evaluasi

Setelah melalui tahap persiapan data, proses pelatihan model dilakukan menggunakan algoritma *Naive Bayes* dan *K-Nearest Neighbor* (KNN). Keduanya dipilih karena memiliki pendekatan yang berbeda dalam mengklasifikasikan data, sebagaimana telah dijelaskan sebelumnya di bab metodologi. Guna mendapatkan hasil klasifikasi yang optimal, proses pemodelan dilakukan dengan tiga konfigurasi berbeda, yaitu model dasar tanpa penyetelan *hyperparameter* (menggunakan parameter *default*), serta dengan penyetelan *hyperparameter* menggunakan metode *Randomized Search* dan *Grid Search*.

Untuk *Naive Bayes*, parameter yang diuji meliputi *priors*, dengan kombinasi nilai [0.1, 0.3, 0.6], [0.3, 0.3, 0.4], dan [0.2, 0.3, 0.5], serta *var_smoothing* yang diatur dalam rentang logaritmik antara 1e-2 hingga 1e-10. Sementara untuk *K-Nearest Neighbor* (KNN), parameter yang diuji mencakup *n_neighbors* dari 1 hingga 50, *weights* dengan pilihan '*uniform*' dan '*distance*', *metric* yang terdiri dari '*minkowski*', '*euclidean*', dan '*manhattan*', *p* dengan nilai 1 dan 2, *leaf_size* yang terdiri dari 10, 20, 30, 40, 50, serta *algorithm* dengan opsi '*auto*', '*ball_tree*', '*kd_tree*', dan '*brute*'. Setiap kombinasi model dan *hyperparameter* diuji dengan metode 10-*fold cross-validation* pada data latih untuk mendapatkan parameter yang menghasilkan performa terbaik.

Guna mengoptimalkan efisiensi waktu komputasi untuk model KNN, hasil terbaik yang ditemukan oleh *Randomized Search* (menggunakan parameter yang telah disebutkan sebelumnya) kemudian digunakan untuk menentukan rentang parameter yang lebih sempit saat melakukan *Grid Search*. Pendekatan ini dipilih karena *Grid Search* bekerja dengan menguji semua kombinasi parameter yang ditentukan dalam ruang pencarian, sehingga menerapkan *Grid Search* pada rentang parameter luas akan membutuhkan waktu komputasi yang sangat lama [26]. Dengan mempersempit rentang di sekitar hasil terbaik, *Grid Search* dapat lebih efektif dalam mencari solusi optimal tanpa menghabiskan waktu yang tidak perlu.

Pelatihan dilakukan dengan tiga rasio pembagian data latih dan uji: 70:30, 80:20, dan 90:10. Dengan total 18 kombinasi pelatihan model yang diujikan. Setiap rasio diuji dengan enam konfigurasi yaitu *Naive Bayes*, *K-Nearest Neighbor*, *Naive Bayes* dengan *Randomized Search*, *K-Nearest Neighbor* dengan *Randomized Search*, *Naive Bayes* dengan *Grid Search*, dan *K-Nearest Neighbor* dengan *Grid Search*. Hasil dari pelatihan tersebut dapat dilihat pada Tabel 2 yang menampilkan akurasi dari setiap skenario.

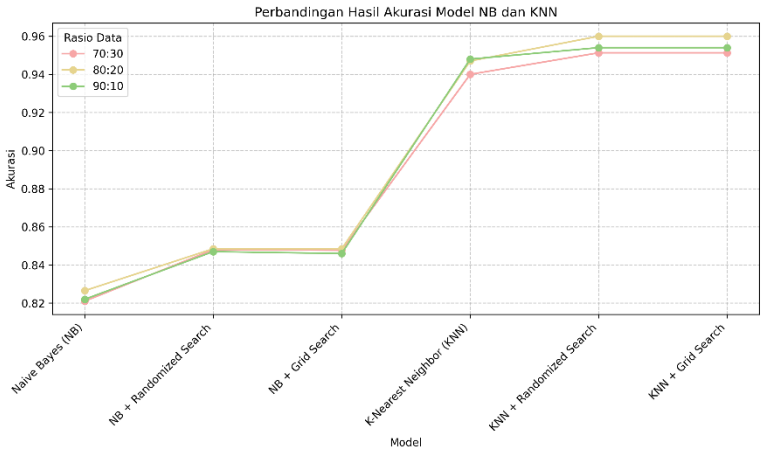
**Tabel 2.** Hasil Akurasi Model

| Model | Split Data | | |
|---|---|---|---|
| | 70:30 | 80:20 | 90:10 |
| *Naive Bayes* (NB) | 0.8210 | 0.8265 | 0.8220 |
| NB + *Randomized Search* | 0.8480 | 0.8485 | 0.8470 |
| NB + *Grid Search* | 0.8477 | 0.8485 | 0.8460 |
| *K-Nearest Neighbor* (KNN) | 0.9400 | 0.9470 | 0.9480 |
| KNN + *Randomized Search* | 0.9513 | 0.9600 | 0.9540 |
| KNN + *Grid Search* | 0.9513 | 0.9600 | 0.9540 |

Berdasarkan Tabel 2, terlihat bahwa model *K-Nearest Neighbor* (KNN) secara konsisten memberikan akurasi lebih tinggi dibandingkan *Naive Bayes* di seluruh rasio. Pada rasio 70:30, model *Naive Bayes* memperoleh akurasi sebesar 0.8210, meningkat menjadi 0.8480 setelah dilakukan *hyperparameter tuning* menggunakan *Randomized Search*, dan 0.8477 dengan *Grid Search*. Model KNN menunjukkan hasil yang lebih unggul dengan akurasi 0.9400 pada model tanpa *hyperparameter tuning* dan 0.9513 baik pada *Randomized Search* maupun *Grid Search*. Pola yang serupa juga ditemukan pada rasio 80:20, di mana akurasi *Naive Bayes* mencapai 0.8265 untuk model tanpa *hyperparameter tuning*, dan meningkat menjadi 0.8485 setelah *hyperparameter tuning* dengan *Randomized Search* dan *Grid Search*, serta Model KNN yang tetap menunjukkan performa tinggi dengan akurasi 0.9470 pada model tanpa *hyperparameter tuning* dan 0.9600 setelah dilakukan *hyperparameter tuning* dengan *Randomized Search* dan *Grid Search*. Pada rasio 90:10, akurasi model *Naive Bayes* tanpa *hyperparameter tuning* berada pada akurasi 0.8220, 0.8470 dengan *Randomized Search*, dan 0.8460 dengan *Grid Search*. Sementara itu, KNN memperoleh hasil akurasi 0.9480 tanpa *hyperparameter tuning*, serta 0.9540 baik dengan *Randomized Search* maupun *Grid Search*. Secara keseluruhan, peningkatan performa pada kedua algoritma setelah dilakukan *hyperparameter tuning* dengan *Randomized Search* dan *Grid Search* berkisar antara 1–2%, sedangkan selisih performa antara *Naive Bayes* dan KNN mencapai rata-rata

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

11% di seluruh rasio data. Untuk visualisasi yang lebih jelas, perbandingan hasil akurasi antar model dapat dilihat melalui *line chart* yang ditampilkan pada Gambar 5.




**Gambar 5.** Grafik Perbandingan Hasil Akurasi Model

Gambar 5 mempertegas pola yang ditampilkan pada Tabel 2, bahwa KNN konsisten memberikan akurasi lebih tinggi dibandingkan *Naive Bayes* di seluruh rasio, dengan nilai akurasi tertinggi sebesar 0.96 dihasilkan oleh model KNN dengan *hyperparameter tuning* pada rasio 80:20. Sedangkan, model *Naive Bayes* hanya mencapai akurasi tertinggi di sekitar 0.84. Gambar 5 juga memperjelas bahwa *hyperparameter tuning* berhasil meningkatkan performa model meskipun hanya sedikit, seperti yang terlihat bahwa sebelum *hyperparameter tuning*, model KNN mencapai akurasi sekitar 0.94 dan *Naive Bayes* sekitar 0.82. Selanjutnya, kinerja setiap model dievaluasi lebih lanjut menggunakan metrik klasifikasi, dengan rata-rata *precision*, *recall*, dan *f1-score* dihitung menggunakan *weighted average*. Hasil evaluasi ini ditampilkan pada Tabel 3.

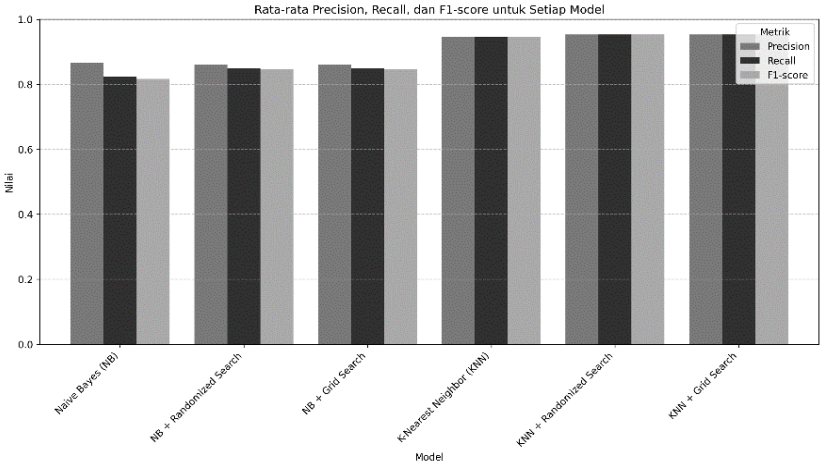
**Tabel 3.** Ringkasan *Precision*, *Recall*, dan *F1-score* (*Weighted Average*)

| Model | *Precision* | *Recall* | *F1-score* |
|---|---|---|---|
| *Naive Bayes* (NB) | 0.86, 0.87, 0.87 | 0.82, 0.83, 0.82 | 0.81, 0.82, 0.82 |
| NB + *Randomized Search* | 0.86, 0.86, 0.86 | 0.85, 0.85, 0.85 | 0.85, 0.85, 0.84 |
| NB + *Grid Search* | 0.86, 0.86, 0.86 | 0.85, 0.85, 0.85 | 0.85, 0.85, 0.84 |
| *K-Nearest Neighbor* (KNN) | 0.94, 0.95, 0.95 | 0.94, 0.95, 0.95 | 0.94, 0.95, 0.95 |
| KNN + *Randomized Search* | 0.95, 0.96, 0.95 | 0.95, 0.96, 0.95 | 0.95, 0.96, 0.95 |
| KNN + *Grid Search* | 0.95, 0.96, 0.95 | 0.95, 0.96, 0.95 | 0.95, 0.96, 0.95 |

Selain tingkat akurasi, nilai *precision*, *recall*, dan *f1-score* juga menunjukkan pola yang relatif stabil, seperti yang terlihat di Tabel 3. Pada seluruh rasio data, ketiga metrik tersebut menunjukkan nilai yang hampir sama antara model KNN sebelum dan sesudah *hyperparameter tuning*, yaitu berada di antara 0.94 hingga 0.96, sementara model *Naive Bayes* sebelum *hyperparameter tuning* memiliki nilai di antara 0.81 hingga 0.87. Setelah dilakukan *hyperparameter tuning*, nilai *precision* pada *Naive Bayes* tetap relatif stabil di sekitar 0.86 hingga 0.87, sedangkan *recall* dan *f1-score* mengalami peningkatan dari kisaran 0.81 hingga 0.83 menjadi sekitar 0.84 hingga 0.85, dengan selisih berkisar antara 0.02 hingga 0.04 dibandingkan model sebelum *hyperparameter tuning*. Nilai *precision*, *recall*, dan *f1-score* dari kedua metode *hyperparameter tuning* (*Grid Search* dan *Randomized Search*) identik pada semua percobaan, menandakan bahwa proses *hyperparameter tuning* tidak mengubah distribusi hasil prediksi secara signifikan, dan performa model relatif stabil terhadap variasi parameter yang diuji.

Didapatkan juga nilai per kelas yang memperkuat temuan bahwa kedua model menghasilkan performa yang konsisten pada seluruh kelas. Meskipun rincian nilai untuk setiap kelas ini tidak disajikan secara terpisah dalam bentuk tabel sebagaimana hasil-hasil sebelumnya, namun rincian berikut diuraikan secara deskriptif untuk menyoroti hasil klasifikasi pada setiap kelas yang distribusi datanya tidak seimbang. Pada model *Naive Bayes*, kelas *Low* menunjukkan kemampuan deteksi model yang baik dengan nilai *recall* yang tinggi sebesar 0.91, meskipun nilai *precision* lebih rendah dibandingkan kelas lainnya. Sementara itu, kelas *Medium* dan *High* menunjukkan keseimbangan antara *precision* dan *recall* dengan *f1-score* masing-masing sebesar 0.81 dan 0.91. Di sisi lain, pada Model KNN, seluruh kelas menunjukkan performa yang lebih unggul dengan nilai *precision* dan *recall* yang tinggi dan relatif seimbang. Hal ini tercermin dari *f1-score* pada masing-masing kelas yang mencapai 0.95, 0.95, 0.97. Nilai-nilai ini secara signifikan menunjukkan bahwa model tetap mampu melakukan prediksi pada seluruh kelas secara stabil dan akurat, meskipun dataset memiliki karakteristik yang tidak seimbang (*imbalanced dataset*). Untuk mempermudah perbandingan performa secara visual, Gambar 6 menyajikan *bar chart* yang merangkum rata-rata nilai *precision*, *recall*, dan *f1-score* antar model di seluruh skenario rasio.

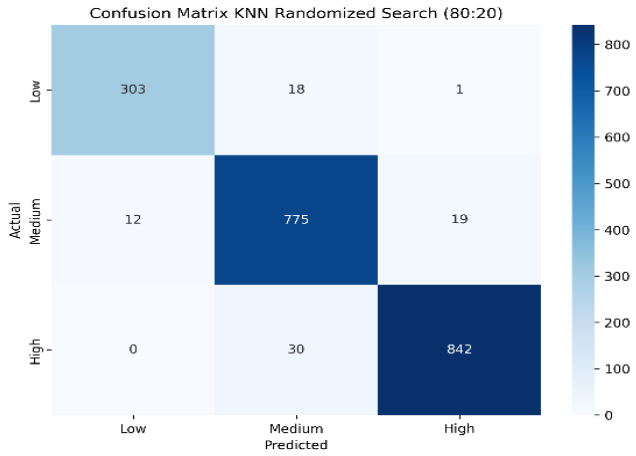
**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765




**Gambar 6.** *Bar Chart* Rata-rata *Precision*, *Recall*, dan *F1-score*

Visualisasi pada Gambar 6 semakin mempertegas hasil yang ditunjukkan dalam Tabel 3, yaitu bahwa nilai *precision*, *recall*, dan *f1-score* di setiap model menunjukkan pola kinerja yang relatif stabil. Untuk KNN, rata-rata ketiga metrik tersebut menghasilkan nilai yang sama persis pada model dengan *hyperparameter tuning*, dan sedikit lebih rendah pada model dasar. Sementara itu, pada *Naive Bayes*, perbedaan nilai ketiga metrik tersebut lebih terlihat namun tidak terlalu besar. Ini menunjukkan bahwa setiap model memiliki performa yang relatif stabil dalam memprediksi kelas data, terlepas dari kombinasi rasio data dan *hyperparameter* yang digunakan.

Berdasarkan hasil evaluasi sebelumnya, ditemukan dua model dengan performa terbaik dan stabil, yaitu KNN dengan *Randomized Search* dan KNN dengan *Grid Search* pada rasio 80:20, yang memiliki nilai metrik klasifikasi yang identik. Selain itu, kedua model ini juga menghasilkan prediksi yang sama persis untuk setiap kelas. Untuk memberikan gambaran mengenai pola prediksi yang dihasilkan kedua model tersebut, Gambar 7 menyajikan *confusion matrix* dari salah satu model.




**Gambar 7.** *Confusion Matrix* KNN dengan *Randomized Search* Rasio 80:20

Gambar 7 menunjukkan prediksi benar dan salah untuk setiap kelas (*Low, Medium, High*) yang diuji. Pada kelas *Low*, model memprediksi benar sebanyak 303 sampel dan salah sebanyak 19 sampel, dengan 18 diprediksi kelas *Medium* dan 1 diprediksi kelas *High*. Pada kelas *Medium*, model memprediksi benar sebanyak 775 sampel dan salah sebanyak 31 sampel (12 diprediksi *Low* dan 19 diprediksi *High*). Sementara itu, pada kelas *High* didapatkan 842 prediksi benar dengan prediksi salah sebanyak 30 sampel pada kelas *Medium*. Kelas *High* terlihat memiliki jumlah prediksi benar paling tinggi, ini sesuai dengan ekspektasi karena kelas ini memang lebih diharapkan untuk dikenali secara tepat. Jumlah prediksi benar yang dominan pada setiap kelas ini sejalan dengan nilai *f1-score* per kelas, yang memiliki nilai tinggi sebesar 0.95 hingga 0.97. Kelas *Low* sebagai kelas dengan jumlah data lebih sedikit tetap memiliki jumlah prediksi benar yang tinggi, kelas *Medium* dan *High* juga menunjukkan tingkat kesalahan yang relatif rendah. Hal ini menunjukkan bahwa model tetap menghasilkan performa yang stabil pada seluruh kelas meskipun dataset bersifat tidak seimbang.

### 3.3 Analisis Hasil

Berdasarkan hasil yang telah diperoleh, algoritma *K-Nearest Neighbor* (KNN) menunjukkan kinerja yang secara konsisten lebih baik daripada *Naive Bayes* pada seluruh rasio pembagian data pelatihan dan pengujian. Selisih akurasi yang cukup besar, yaitu sekitar 11%, mengindikasikan perbedaan mendasar pada cara kerja kedua algoritma tersebut.

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

Dari sisi teoritis, KNN bekerja dengan menghitung jarak antar data dan mengklasifikasikan data baru berdasarkan kelas mayoritas dari k tetangga terdekat [20], [21], [24]. Metode ini lebih mampu menangani pola data non-linear dengan baik, sehingga lebih adaptif terhadap variasi nilai atribut serta sebaran data yang kompleks tanpa memerlukan asumsi distribusi tertentu [24]. Sebaliknya, *Naive Bayes* membuat asumsi bahwa setiap fitur saling independen [17], [18], [30]. Ketika terdapat ketergantungan antar atribut atau variasi nilai fitur yang tinggi, asumsi tersebut tidak terpenuhi sehingga menurunkan akurasi prediksi [30]. Mengingat karakteristik dataset *cybersecurity_attacks* yang memiliki variasi fitur cukup tinggi, metode berbasis jarak seperti KNN lebih efektif dalam menangkap pola kompleks yang ada, sehingga lebih sesuai untuk melakukan klasifikasi pada dataset ini.

Hasil juga memperlihatkan bahwa *hyperparameter tuning* dengan *Randomized Search* dan *Grid Search* mampu meningkatkan performa sekitar 1–2% pada kedua algoritma, dengan akurasi sebesar 0.94-0.948 sebelum *tuning* dan 0.9513-0.96 setelah *tuning*. Meskipun secara angka peningkatan ini terlihat kecil, dari segi metodologi, temuan ini tetap penting karena menunjukkan bahwa parameter *default* sudah mendekati titik optimal. Proses *hyperparameter tuning* memberikan kontribusi pada peningkatan stabilitas model, sehingga akurasi, *precision*, *recall*, dan *F1-score* relatif konsisten di sebagian besar percobaan, menandakan bahwa ruang pencarian yang digunakan pada kedua model sudah cukup representatif. Dalam KNN, kesamaan hasil antara *Randomized Search* dan *Grid Search* lebih terlihat jelas karena *Grid Search* dilakukan di sekitar parameter terbaik yang ditemukan oleh *Randomized Search*, sehingga keduanya menghasilkan kombinasi parameter yang sama-sama optimal. Namun, jika dilihat dari segi efisiensi, *Randomized Search* dapat dipertimbangkan sebagai alternatif yang lebih baik karena mampu menghasilkan performa yang setara dengan *Grid Search* tetapi dalam waktu komputasi yang lebih singkat.

Hasil *confusion matrix* memperlihatkan kinerja model yang sangat baik. Kelas *High* dan *Medium* menunjukkan akurasi tertinggi dan seimbang, masing-masing memiliki 842 dan 775 prediksi benar dengan prediksi salah sebanyak 30 dan 31 sampel. Hal ini menunjukkan bahwa model KNN mampu memisahkan kelas data *High* dan *Medium* secara efektif dalam ruang fitur, sejalan dengan prinsipnya yang mengandalkan kedekatan jarak untuk menentukan keputusan [22]. Sementara itu, kelas *Low* mencatat prediksi benar terendah sebanyak 303 sampel dengan 19 kesalahan klasifikasi, tetapi ini disebabkan karena jumlah sampel kelas *Low* secara keseluruhan yang memang paling sedikit, bukan karena kegagalan model. Dominannya jumlah prediksi benar pada setiap kelas yang sejalan dengan nilai *f1-score* per kelas dan jumlah kesalahan total yang sangat minim, yaitu sebanyak 80 sampel, menunjukkan bahwa meskipun datanya tidak seimbang, model KNN mampu mengenali serangan berisiko tinggi secara cukup akurat dan bekerja secara stabil dalam mengidentifikasi pola untuk ketiga kelas tersebut.

Selain itu, hasil menunjukkan bahwa rasio pembagian data (70:30, 80:20, 90:10) tidak memberikan pengaruh yang signifikan terhadap performa akhir model, dengan perbedaan performa berada di bawah 2%. Kondisi ini mengindikasikan bahwa dataset *cybersecurity_attacks* memiliki tingkat konsistensi yang baik serta cukup representatif di setiap rasio data pelatihan dan pengujian. Meskipun demikian, rasio 80:20 menghasilkan performa tertinggi untuk model KNN dengan akurasi 0.96, yang dapat diartikan sebagai titik keseimbangan yang optimal antara proporsi data pelatihan dan pengujian dalam menilai kemampuan generalisasi model secara akurat.

Jika dibandingkan dengan penelitian terdahulu yang menggunakan model serupa, beberapa studi melaporkan akurasi yang lebih tinggi, dengan salah satu penelitian mencapai akurasi tertinggi sebesar 99,7% untuk KNN dan 88,55% untuk *Naive Bayes*, serta stabilitas akurasi yang konsisten pada kisaran 99% [12]. Namun, perbedaan ini tidak dapat dibandingkan secara langsung karena setiap penelitian menggunakan dataset, karakteristik fitur, dan tujuan klasifikasi yang berbeda. Penelitian sebelumnya umumnya memakai dataset yang lebih terstruktur dan seimbang [9], sedangkan penelitian ini menggunakan dataset dengan distribusi kelas yang tidak merata setelah proses *relabelling*. Selain itu, sebagian besar studi terdahulu berfokus pada deteksi serangan [8], [9], [10], [11], [12], sementara penelitian ini menargetkan klasifikasi tingkat serangan, yang merupakan tugas dengan tingkat kesulitan lebih tinggi. Dari sisi metodologi, penelitian ini juga menerapkan *hyperparameter tuning* melalui *GridSearchCV* dan *RandomizedSearchCV*. Dengan demikian, meskipun akurasi yang diperoleh berada di bawah beberapa studi sebelumnya, penelitian ini memberikan kontribusi melalui fokus klasifikasi yang lebih spesifik dan pendekatan optimasi model yang lebih terstruktur.

Secara keseluruhan, penelitian ini menunjukkan bahwa algoritma *K-Nearest Neighbor* (KNN) menghasilkan performa terbaik dan paling konsisten dibandingkan dengan model *Naive Bayes* dalam mengklasifikasikan tingkat serangan siber pada dataset *cybersecurity_attacks*. Keunggulan ini dapat dilihat dari nilai akurasi tertinggi yang dicapai, yaitu 0.96 pada rasio 80:20 setelah proses *tuning*, meningkat dari akurasi sebelum *tuning* sebesar 0.947, serta kestabilan nilai *precision*, *recall*, dan *f1-score* di berbagai rasio data pelatihan dan pengujian. Sementara Naive Bayes hanya mencapai akurasi tertinggi 0.8485 setelah *tuning* dan 0.8265 sebelum *tuning* pada rasio yang sama. Meskipun proses *hyperparameter tuning* tidak memberikan peningkatan performa yang besar, langkah tersebut tetap krusial untuk menjaga konsistensi hasil dan memastikan bahwa model bekerja dengan konfigurasi parameter yang efisien.

Berdasarkan hasil yang diperoleh, KNN dapat dijadikan pilihan yang ideal untuk diterapkan dalam sistem deteksi atau klasifikasi serangan siber, terutama ketika dibutuhkan model yang sederhana, efisien, dan mudah diimplementasikan. Dalam penerapannya, model KNN yang telah dilakukan *hyperparameter tuning* dapat digunakan untuk membantu proses pengelompokan dan penentuan prioritas penanganan serangan siber berdasarkan tingkat serangannya, serta menjadi *baseline* kuat untuk pengembangan model klasifikasi lainnya di bidang keamanan siber, khususnya dalam mendeteksi atau mengklasifikasikan berbagai jenis serangan. Dengan demikian, dapat disimpulkan

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

bahwa KNN dengan *hyperparameter tuning* memberikan performa yang lebih optimal dibanding *Naive Bayes* dalam penelitian ini.

## 4. KESIMPULAN

Penelitian ini menunjukkan bahwa algoritma *K-Nearest Neighbor* (KNN) memberikan performa lebih baik dibandingkan *Naive Bayes* dalam klasifikasi tingkat serangan siber pada dataset *cybersecurity_attacks*. Di mana model KNN mencapai akurasi tertinggi sebesar 0.96 pada rasio data latih 80:20 setelah proses *tuning*, meningkat dari akurasi sebelum *tuning* sebesar 0.947, dengan nilai *precision*, *recall*, dan *f1-score* di kisaran 0.95 hingga 0.96. Berdasarkan *confusion matrix*, model dapat memprediksi kelas data dengan jumlah kesalahan yang relatif kecil pada semua kelas, menunjukkan bahwa model mampu mengenali serangan berisiko tinggi secara cukup akurat. Meskipun peningkatan performa setelah *hyperparameter tuning* melalui *Randomized Search* dan *Grid Search* tidak terlalu besar, proses tersebut tetap berkontribusi dalam menghasilkan model yang lebih baik dan konsisten. Selain itu, rasio pembagian data yang tidak berpengaruh besar terhadap performa menunjukkan bahwa model mampu menyesuaikan diri dalam menangani variasi proporsi data. Dengan tingkat akurasi sebesar 0.96, model dapat digunakan sebagai langkah awal untuk memprioritaskan penanganan serangan siber berdasarkan tingkat serangannya, sehingga sumber daya dapat dialokasikan secara lebih efektif dan waktu respons terhadap insiden kritis dapat dipercepat. Untuk pengembangan penelitian di masa mendatang, beberapa pendekatan dapat dipertimbangkan untuk meningkatkan performa dan stabilitas model. Salah satu yang berpotensi memberikan hasil lebih baik adalah melakukan kombinasi model KNN dengan model *ensemble* seperti *bagging* atau *boosting*, misalnya *Random Forest*, *XGBoost*, atau *LightGBM*, yang umumnya mampu menangani variasi distribusi kelas dan menghasilkan generalisasi yang lebih kuat dibandingkan model tunggal. Selain itu, pengembangan lain yang dapat dilakukan adalah memperluas ruang pencarian *hyperparameter* serta menguji model pada dataset berbeda untuk melihat konsistensi kinerjanya pada karakteristik data yang lebih beragam. Dengan temuan yang telah diperoleh dari penelitian ini, diharapkan pendekatan lanjutan tersebut dapat menghasilkan model klasifikasi tingkat serangan siber yang lebih adaptif dan dapat diimplementasikan secara lebih efektif pada skenario keamanan siber yang nyata.

## REFERENCES

[1] E. S. P. Sinlae, I. F. Syahda, and A. U. Hosnah, "Kriminalitas Cyber Bjorka: Ancaman dan Tantangan di Era Digital," *Jurnal Justitia: Jurnal Ilmu Hukum dan Humaniora*, vol. 7, no. 1, pp. 208–289, 2024, doi: http://dx.doi.org/10.31604/justitia.v7i1.280-289.

[2] E. M. Kala, "The Impact of Cyber Security on Business: How to Protect Your Business," *Open Journal of Safety Science and Technology*, vol. 13, no. 02, pp. 51–65, 2023, doi: 10.4236/ojsst.2023.132003.

[3] T. G. Laksana and S. Mulyani, "Pengetahuan Dasar Identifikasi Dini Deteksi Serangan Kejahatan Siber Untuk Mencegah Pembobolan Data Perusahaan," *Jurnal Ilmiah Multidisiplin (JUKIM)*, vol. 3, no. 1, pp. 109–122, Jan. 2024, doi: 10.56127/jukim.v3i01.1143.

[4] I. P. Putri, Terttiaavini, and N. Arminarahmah, "Analisis Perbandingan Algoritma Machine Learning untuk Prediksi Stunting pada Anak," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 1, pp. 257–265, Jan. 2024, doi: 10.57152/malcom.v4i1.1078.

[5] A. F. Mahmud and S. Wirawan, "Deteksi Phishing Website menggunakan Machine Learning Metode Klasifikasi," *Sistemasi: Jurnal Sistem Informasi*, vol. 13, no. 4, pp. 1368–1380, 2024, doi: 10.32520/stmsi.v13i4.3456.

[6] R. B. Hadiprakoso, W. R. Aditya, and F. N. Pramitha, "Analisis Statis Deteksi Malware Android Menggunakan Algoritma Supervised Machine Learning," *Cyber Security dan Forensik Digital*, vol. 5, no. 1, pp. 1–5, May 2022, doi: 10.14421/csecurity.. 2022.5.1.3116.

[7] A. Fauzi *et al.*, "Penerapan Random Forest dan Adaboost untuk Klasifikasi Serangan DDoS," *Journal on Education*, vol. 05, no. 03, pp. 7925–7937, 2023, doi: 10.31004/joe.v5i3.1920.

[8] K. B. Dasari and N. Devarakonda, "Detection of DDoS Attacks Using Machine Learning Classification Algorithms," *International Journal of Computer Network and Information Security*, vol. 14, no. 6, pp. 89–97, Dec. 2022, doi: 10.5815/ijcnis.. 2022.06.07.

[9] J. Naufal Semendawai, D. Stiawan, and I. Pahendra, "Shellcode Classification with Machine Learning Based on Binary Classification," *Indonesia Journal of Social Technology*, vol. 6, no. 2, pp. 833–844, Feb. 2025, doi: 10.59141/jist.v6i2.3233.

[10] Herman, I. Riadi, and Y. Kurniawan, "Vulnerability Detection With K-Nearest Neighbor and Naïve Bayes Method using Machine Learning," *International Journal of Artificial Intelligence Research*, vol. 7, no. 1, pp. 10–18, Jun. 2023, doi: 10.29099/ijair.v7i1.795.

[11] D. Hindarto, R. E. Indrajit, and E. Dazki, "Perbandingan Kinerja Akurasi Klasifikasi K-NN, NB dan DT Pada APK Android," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 9, no. 1, pp. 486–503, Mar. 2022, doi: 10.35957/jatisi.v9i1.1542.

[12] A. D. Afifaturahman and F. Maulana, "Perbandingan Algoritma K-Nearest Neighbour (KNN) dan Naive Bayes pada Intrusion Detection System (IDS)," *Innovation in Research of Informatics (INNOVATICS)*, vol. 3, no. 1, pp. 17–25, 2021, doi: 10.37058/innovatics.v3i1.2852.

[13] S. Junaidi, R. V. Anggela, and D. Kariman, "Klasifikasi Metode Data Mining untuk Prediksi Kelulusan Tepat Waktu Mahasiswa dengan Algoritma Naïve Bayes, Random Forest, Support Vector Machine (SVM) dan Artificial Neural Nerwork (ANN)," *Journal of Applied Computer Science and Technology (JACOST)*, vol. 5, no. 1, pp. 109–119, Jun. 2024, doi: 10.52158/jacost.v5i1.489.

**Building of Informatics, Technology and Science (BITS)**
Volume 7, No 3, December 2025 Page: 1974−1985
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i3.8765

[14] S. R. Cholil, T. Handayani, R. Prathivi, and T. Ardianita, "Implementasi Algoritma Klasifikasi K-Nearest Neighbor (KNN) Untuk Klasifikasi Seleksi Penerima Beasiswa," *IJCIT (Indonesian Journal on Computer and Information Technology)*, vol. 6, no. 2, pp. 118–127, 2021, doi: 10.31294/ijcit.v6i2.10438.

[15] G. L. Pritalia, "Analisis Komparatif Algoritme Machine Learning pada Klasifikasi Kualitas Air Layak Minum," *KONSTELASI: Konvergensi Teknologi dan Sistem Informasi*, vol. 2, no. 1, pp. 43–55, Apr. 2022, doi: 10.24002/konstelasi.v2i1.5630.

[16] Y. E. J. Putra, I. Imelda, and S. Suryadih, "Seleksi Fitur SelectKBest Dalam Prediksi Kelulusan Mahasiswa Tepat Waktu dengan Decision Tree," *Building of Informatics, Technology and Science (BITS)*, vol. 6, no. 4, pp. 2776–2784, Mar. 2025, doi: 10.47065/bits.v6i4.7086.

[17] R. M. Sari *et al.*, *Klasifikasi Data Mining*. Serasi Media Teknologi, 2024. [Online]. Available: https://www.google.co.id/books/edition/Klasifikasi_Data_Mining/xTwXEQAAQBAJ?hl=id&gbpv=0

[18] N. Febriyanti and A. F. Rozi, "Komparasi Algoritma Naïve Bayes, Support Vector Machine, dan Random Forest Untuk Analisis Sentimen Ulasan Pengguna Aplikasi CGV Cinemas Indonesia," *Building of Informatics, Technology and Science (BITS)*, vol. 7, no. 1, pp. 453–464, Jun. 2025, doi: 10.47065/bits.v7i1.7459.

[19] Mustika *et al.*, *Data Mining dan Aplikasinya*. Bandung: Widina Bhakti Persada Bandung, 2021. [Online]. Available: https://www.google.co.id/books/edition/DATA_MINING_DAN_APLIKASINYA/53FXEAAAQBAJ?hl=id&gbpv=0

[20] R. A. Nolly, A. Fitria, and K. Saputra S, "Penerapan Algoritma K-Nearest Neighbors untuk Klasifikasi Fragmen Metagenom Berdasarkan Ekstraksi Fitur K-Mers," *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*, vol. 17, no. 1, pp. 52–56, Feb. 2023, doi: 10.30872/jim.v17i1.5779.

[21] V. Sianipar, D. Irmayani, and B. Bangun, "Analisis Faktor-Faktor yang Mempengaruhi Tingkat Kelulusan Siswa Menggunakan Algoritma KNN," *Building of Informatics, Technology and Science (BITS)*, vol. 7, no. 1, pp. 626–637, Jun. 2025, doi: 10.47065/bits.v7i1.7386.

[22] N. M. Putry and B. N. Sari, "Komparasi Algoritma KNN dan Naïve Bayes Untuk Klasifikasi Diagnosis Penyakit Diabetes Melitus," *Evolusi: Jurnal Sains dan Manajemen*, vol. 10, no. 1, pp. 45–57, Sep. 2022, doi: 10.31294/evolusi.v10i1.12514.

[23] R. A. Putra, *Langkah Mudah Belajar Machine Learning dengan Python untuk Pemula*. Yogyakarta: Anak Hebat Indonesia, 2024. [Online]. Available: https://www.google.co.id/books/edition/Langkah_Mudah_Belajar_Machine_Learning_D/JulyEQAAQBAJ?hl=id&gbpv=0&kptab=overview

[24] A. J. Himawan, A. M. K. Sari, N. A. Parsa, K. S. P. Hermansyah, and E. S. D. Rizki, "Penerapan Metode K-Nearest Neighbors dalam Mendeteksi Website Phishing," *Jurnal Kecerdasan Buatan, Komputasi dan Teknologi Informasi*, vol. 5, no. 2, pp. 167–173, Dec. 2024, doi: 10.33650/coreai.v5i2.10484.

[25] P. Padman, *Learn Data Science from Scratch: Mastering ML and NLP with Python in a Step-by-step Approach (English Edition)*, 1st ed. BPB Publications, 2024. [Online]. Available: https://www.google.co.id/books/edition/Learn_Data_Science_from_Scratch/rhX1EAAAQBAJ?hl=id&gbpv=0

[26] L. Owen, *Hyperparameter Tuning with Python: Boost Your Machine Learning Model's Performance Via Hyperparameter Tuning*, 1st Edition. Packt Publishing, 2022. [Online]. Available: https://www.google.co.id/books/edition/_/CqF-EAAAQBAJ?hl=id&gbpv=1

[27] A. R. Arsyan and W. F. Al Maki, "Classification of Glaucoma Using Invariant Moment Methods on K-Nearest Neighbor and Random Forest Models," *Building of Informatics, Technology and Science (BITS)*, vol. 3, no. 4, pp. 466–472, Mar. 2022, doi: 10.47065/bits.v3i4.1244.

[28] A. M. Halim, M. Dwifebri, and F. Nhita, "Handling Imbalanced Data Sets Using SMOTE and ADASYN to Improve Classification Performance of Ecoli Data Sets," *Building of Informatics, Technology and Science (BITS)*, vol. 5, no. 1, pp. 246–253, Jun. 2023, doi: 10.47065/bits.v5i1.3647.

[29] N. Soprala, D. Sinnreich, N. Kumar, and S. Furqhan, "Anomaly Detection in Streaming Time Series Data with Online Learning using Amazon Managed Service for Apache Flink," aws.amazon.com. Accessed: Sep. 22, 2025. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/anomaly-detection-in-streaming-time-series-data-with-online-learning-using-amazon-managed-service-for-apache-flink/

[30] R. Blanquero, E. Carrizosa, P. Ramírez-Cobo, and M. R. Sillero-Denamiel, "Variable Selection for Naïve Bayes Classification," *Computers and Operations Research 135*, Jul. 2021, doi: 10.1016/j.cor.2021.105456.