BAB 2
LANDASAN TEORI

2.1 Tinjauan Teori

2.1.1 Media Sosial dan Remaja

Media sosial merupakan platform daring yang memungkinkan penggunanya untuk menciptakan, membagikan, dan berinteraksi dengan berbagai jenis konten, seperti teks, gambar, video, dan audio [1]. Selain berfungsi sebagai sarana berbagi informasi, media sosial juga menjadi ruang interaksi sosial yang mempertemukan individu dari berbagai latar belakang dalam lingkungan digital. Beberapa platform media sosial yang paling sering digunakan antara lain X (sebelumnya dikenal sebagai Twitter), Facebook, Instagram, TikTok, dan YouTube [3]. Saat ini, media sosial tersebut telah menjadi bagian integral dalam kehidupan masyarakat, terutama di kalangan remaja [1].

Kalangan remaja sendiri dikategorikan ke dalam dua kelompok usia, yaitu remaja awal (10–14 tahun) dan remaja akhir (15–19 tahun) [6]. Pada fase perkembangan ini, individu cenderung memiliki keterikatan yang tinggi terhadap media sosial karena berbagai manfaat yang diperoleh, seperti kemudahan dalam berkomunikasi dengan teman sebaya, memperluas jaringan sosial, mengekspresikan diri, serta memperoleh dan menyebarkan informasi secara cepat [1]. Namun, media sosial juga membawa sejumlah dampak negatif yang dapat mempengaruhi kesejahteraan psikologis remaja akibat informasi maupun komentar negatif yang tersebar. Diantaranya seperti peningkatan kecemasan, depresi, bahkan munculnya pemikiran untuk bunuh diri [8][13].

Media sosial X menjadi salah satu platform yang rentan menjadi ruang bagi penyebaran komentar negatif, termasuk pelecehan seksual verbal, yang dapat berdampak serius pada kesehatan mental remaja [7]. Oleh karena itu, memahami hubungan antara remaja dan media sosial X menjadi penting dalam mendukung upaya deteksi serta pencegahan pelecehan seksual berbasis *online*.

2.1.2 Kekerasan Berbasis Gender Online

Kekerasan Berbasis Gender *Online* (KBGO) adalah bentuk kekerasan yang dilakukan melalui platform digital dan menargetkan individu berdasarkan

gender atau seksualitas mereka. KBGO mencakup berbagai bentuk tindakan seperti pelecehan daring (*cyber harassment*), eksploitasi seksual berbasis teknologi, penguntitan daring (*cyberstalking*), serta penyebaran informasi pribadi tanpa izin (*doxing*) [8].

Menurut laporan Komnas Perempuan, KBGO mengalami peningkatan signifikan dalam beberapa tahun terakhir. Data mencatat bahwa pada tahun 2020, terdapat lonjakan laporan KBGO sebesar 48% dibandingkan tahun sebelumnya, dengan kasus utama berupa ancaman penyebaran materi seksual korban serta pelecehan seksual daring [8]. KBGO juga sering kali dilakukan oleh orang-orang terdekat korban, seperti pasangan atau mantan pasangan, yang memanfaatkan internet sebagai sarana untuk melanjutkan kontrol dan pelecehan [5].

Selain berdampak pada kesehatan mental korban, KBGO juga memiliki dampak sosial yang luas. Studi menemukan bahwa korban KBGO cenderung mengalami isolasi sosial akibat stigma dan rasa takut untuk berbicara mengenai pengalaman mereka [8]. Keadaan ini diperburuk oleh kurangnya perlindungan hukum yang efektif bagi korban, karena regulasi yang ada sering kali lebih menitikberatkan pada hukuman bagi pelaku daripada pemulihan korban [5]. Oleh karena itu, diperlukan upaya bersama dalam bentuk kebijakan yang lebih kuat dan kampanye kesadaran publik untuk mengurangi prevalensi KBGO di ruang digital [5].

2.1.3 Pelecehan Seksual Berbasis Online

Pelecehan seksual berbasis *online* merupakan salah satu bentuk kekerasan berbasis gender yang terjadi di ruang digital, di mana pelaku melakukan tindakan seperti pengiriman pesan seksual tanpa persetujuan, penyebaran konten intim tanpa izin, serta ancaman atau pemerasan berbasis teknologi [5]. Karakteristik utama dari pelecehan ini adalah anonimitas dan aksesibilitas yang disediakan oleh platform digital, yang memungkinkan pelaku bertindak dengan risiko rendah untuk dikenali atau dihukum [5].

Dalam konteks media sosial, pelecehan seksual berbasis *online* dapat muncul dalam berbagai bentuk, termasuk komentar bernada seksual yang tidak diinginkan, ajakan yang bersifat melecehkan, hingga ancaman kekerasan seksual secara eksplisit. Dampak dari tindakan ini tidak hanya merugikan korban secara emosional dan psikologis, tetapi juga berpotensi menyebabkan trauma jangka panjang, perasaan tidak aman, dan penarikan diri dari ruang digital [5].

Untuk mengatasi fenomena ini, diperlukan pendekatan yang komprehensif melalui penguatan regulasi, peningkatan literasi digital, serta pengembangan teknologi deteksi otomatis yang mampu mengidentifikasi komentar bermuatan pelecehan seksual secara akurat [9]. Deteksi dini melalui algoritma *machine learning* menjadi salah satu upaya penting dalam menciptakan lingkungan digital yang lebih aman dan inklusif [10].

2.1.4 Data Collection

Data collection atau pengumpulan data merupakan tahap penting dalam penelitian berbasis analisis teks dan *machine learning*, yang bertujuan untuk memperoleh data yang valid dan relevan dalam mendeteksi pelecehan seksual verbal [23]. Dalam konteks media sosial, metode *web scraping* sering digunakan untuk mengekstrak teks dari berbagai platform daring secara otomatis, memungkinkan pengumpulan data dalam jumlah besar dengan efisien.

Dalam penelitian ini, alat yang digunakan untuk mengumpulkan data dari media sosial X adalah *tweet harvest*, sebuah *command-line tool* berbasis *playwright* yang memungkinkan pengambilan *tweet* berdasarkan kata kunci tertentu dan rentang waktu yang ditentukan [24]. *Tweet harvest* dijalankan di lingkungan komputasi berbasis *cloud* seperti Google Colab, yang memudahkan proses *scraping* tanpa memerlukan perangkat keras lokal. Selain itu, penggunaan Google Colab mendukung berbagai pustaka *data science* seperti *pandas* untuk manipulasi data setelah proses pengumpulan selesai [25].

Selama proses *scraping*, *tweet harvest* memanfaatkan *authorization token* yang diperoleh melalui sesi *login* pengguna untuk mengakses data. Data yang berhasil dikumpulkan akan disimpan dalam format CSV, memudahkan proses analisis lebih lanjut. Setelah pengumpulan, dilakukan proses *filtering* untuk memastikan hanya data yang relevan yang digunakan dalam analisis. *Filtering* ini meliputi penerapan kata kunci spesifik, penghapusan *tweet* duplikat, serta seleksi berdasarkan metadata seperti tanggal dan jumlah interaksi *tweet* [24].

Metode ini mendukung penelitian dalam membangun *dataset* yang lebih bersih dan terfokus, sehingga meningkatkan akurasi model dalam mendeteksi komentar pelecehan seksual verbal. Dengan memanfaatkan lingkungan berbasis *cloud* seperti Google Colab, proses pengumpulan data menjadi lebih efisien, terutama dalam menangani *dataset* berukuran besar, serta mempermudah integrasi dengan penyimpanan awan seperti Google Drive [25].

2.1.5 Data Labelling

Data labelling merupakan tahapan penting dalam proses persiapan data, di mana setiap entri teks diberi label atau kategori tertentu sesuai dengan tujuan analisis. Proses ini banyak diterapkan dalam berbagai bidang seperti analisis sentimen, klasifikasi topik, hingga deteksi konten sensitif di media sosial (Sebastian, 2002; [26]). Untuk memastikan kualitas dan keandalan *dataset*, pelabelan sering dilakukan secara manual dengan melibatkan lebih dari satu *annotator*. Keterlibatan beberapa anotator ini bertujuan untuk meningkatkan konsistensi, serta memastikan validitas dan objektivitas dalam proses pelabelan [23].

2.1.6 Resampling

Hasil dari proses *data labelling* umumnya menunjukkan distribusi label yang tidak merata, sehingga menimbulkan permasalahan *imbalanced data* atau ketidakseimbangan kelas. Ketidakseimbangan ini terjadi ketika salah satu kelas memiliki jumlah data yang jauh lebih besar dibandingkan kelas lainnya. Kondisi ini dapat berdampak negatif terhadap performa model klasifikasi, karena algoritma cenderung mempelajari pola dari kelas mayoritas dan mengabaikan kelas minoritas [22]. Ketidakseimbangan kelas merupakan tantangan umum dalam tugas klasifikasi dan dapat memengaruhi akurasi prediksi secara keseluruhan, terutama ketika perbandingan antar kelas terlalu jauh [22]. Untuk mengatasi masalah tersebut, teknik *resampling* sering digunakan, yang terbagi menjadi dua pendekatan utama: *random undersampling* dan *random oversampling*. *Random undersampling* dilakukan dengan mengurangi jumlah data pada kelas mayoritas secara acak, sedangkan *random oversampling* menambahkan data pada kelas minoritas dengan cara yang serupa [27]. Dengan penerapan teknik resampling, distribusi kelas dalam *dataset* menjadi lebih seimbang, sehingga membantu meningkatkan performa model klasifikasi yang akan dibangun [27].

2.1.7 Text Preprocessing

Text preprocessing merupakan langkah awal dalam pemrosesan teks yang bertujuan untuk mengubah data mentah menjadi format yang lebih terstruktur dan bersih sebelum digunakan dalam analisis lebih lanjut atau diterapkan dalam model *machine learning* [14]. Proses ini sangat penting dalam meningkatkan

akurasi serta efisiensi analisis teks karena membantu menghilangkan *noise* dan mengurangi dimensi fitur yang tidak relevan. Setiap tahapan *preprocessing* saling berkaitan dalam mempersiapkan data agar lebih mudah diolah oleh algoritma *machine learning* [14].

A Cleaning Text dan Case Folding

Cleaning text adalah tahap awal untuk membersihkan teks dari karakter yang tidak diperlukan seperti tanda baca, angka, simbol, URL, emoji, *mention* (*username*), dan *hashtag* (#tag) guna mengurangi *noise* dalam data [9]. Selain itu, *case folding* dilakukan dengan mengubah seluruh huruf menjadi huruf kecil agar proses analisis tidak terpengaruh oleh perbedaan kapitalisasi huruf [9]. Langkah ini bertujuan untuk memastikan data teks lebih seragam, sehingga dapat diproses lebih efektif dalam tahapan selanjutnya.

B Tokenization

Tokenization adalah proses memecah teks menjadi unit-unit yang lebih kecil seperti kata atau frasa, yang disebut sebagai token [16]. Teknik ini memungkinkan sistem untuk memahami dan mengelola teks dalam format yang dapat diproses oleh algoritma *machine learning*. *Tokenization* sangat penting karena model *machine learning* bekerja dengan data yang telah dipisahkan menjadi elemen-elemen kecil yang dapat dianalisis secara individual [16].

C Normalization

Normalization adalah proses standarisasi teks dengan mengubah kata-kata tidak baku atau kata singkatan menjadi bentuk baku yang memiliki makna jelas [14]. Dalam konteks analisis komentar di media sosial, normalisasi sangat penting karena banyak pengguna yang menggunakan bahasa tidak formal, seperti singkatan, slang, atau *typo*. Contohnya, kata ”gk” diubah menjadi ”tidak”, atau ”bgt” menjadi ”banget” [14]. Proses normalisasi dibutuhkan *dictionary* yang sudah divalidasi sebelumnya untuk memperoleh proses analisis lebih akurat dan valid [14].

D Stopwords Removal

Stopwords removal adalah proses menghapus kata-kata yang tidak memiliki makna penting dalam analisis teks, seperti ”dan”, ”di”, ”ke”, ”dari”, ”yang” dalam bahasa Indonesia. Kata-kata ini dihapus karena tidak memberikan kontribusi signifikan dalam proses klasifikasi teks atau analisis sentimen [14]. Dengan menghilangkan *stopwords*, jumlah fitur dalam teks dapat dikurangi tanpa menghilangkan informasi penting.

E Stemming

Stemming adalah proses mengubah kata ke bentuk dasarnya dengan menghilangkan imbuhan seperti awalan (me-, ber-, di-, ke-) dan akhiran (-kan, -i, -an) [28]. Teknik *stemming* bertujuan untuk mengurangi jumlah kata unik dalam *dataset*, sehingga model *machine learning* dapat bekerja lebih efisien dengan fitur yang lebih sedikit tetapi tetap mewakili makna dari teks asli. Salah satu alat yang sering digunakan dalam *stemming* bahasa Indonesia adalah pustaka Sastrawi yang menyediakan algoritma *stemming* berbasis aturan bahasa Indonesia [14].

2.1.8 Feature Extraction

Feature extraction adalah proses esensial dalam analisis teks yang bertujuan untuk mengubah data teks mentah menjadi representasi numerik yang dapat diolah oleh algoritma *machine learning* [29]. Terdapat berbagai pendekatan untuk melakukan ini, yang secara umum dapat dibedakan berdasarkan fokusnya: pendekatan statistik dan pendekatan semantik [29]. Metode statistik seperti *Term Frequency - Inverse Document Frequency* (TF-IDF) mengukur tingkat kepentingan sebuah kata dengan mengevaluasi frekuensinya dalam satu dokumen relatif terhadap keseluruhan korpus, sehingga sangat efektif dalam menonjolkan istilah-istilah pembeda [9][23][29]. Sebaliknya, pendekatan *word embedding* seperti *Word2Vec* dan *FastText* dirancang untuk menangkap hubungan semantik, di mana makna sebuah kata dipelajari dari konteks kata-kata lain di sekitarnya [29]. Pilihan antara kedua pendekatan ini bergantung pada apakah tugas klasifikasi lebih memerlukan penekanan pada bobot kata kunci spesifik atau pemahaman makna kontekstual [29].

2.1.9 Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) adalah metode pembobotan kata yang digunakan dalam analisis teks untuk mengukur seberapa penting suatu kata dalam dokumen relatif terhadap kumpulan dokumen lainnya [24]. Sebelum melakukan pembobotan, TF-IDF terlebih dahulu membangun sebuah kamus global (*vocabulary*) yang berisi semua kata unik dari seluruh korpus dokumen [30][31]. Ukuran dari kamus inilah yang akan menentukan dimensi atau jumlah fitur dari vektor numerik yang dihasilkan untuk setiap dokumen [28].

Setelah kamus terbentuk, TF-IDF menggabungkan dua komponen utama untuk menghitung bobot setiap kata:

1. *Term Frequency* (TF): Mengukur frekuensi kemunculan suatu kata dalam sebuah dokumen. Kata yang lebih sering muncul dalam dokumen memiliki nilai TF yang lebih tinggi [24].
2. *Inverse Document Frequency* (IDF): Mengukur seberapa jarang suatu kata muncul dalam keseluruhan korpus dokumen. Kata yang jarang muncul di banyak dokumen akan memiliki bobot IDF yang lebih tinggi [24].

Rumus 2.1 menunjukkan cara perhitungan *TF-IDF*.

$$W_{dt} = tf_{dt} \times idf_t = tf_{dt} \times \log\left(\frac{N}{df_t}\right) \tag{2.1}$$

Keterangan dari persamaan di atas adalah sebagai berikut [24]:

- W_{dt} : Bobot *term t* pada dokumen *d*
- tf_{dt} : Frekuensi kemunculan *term t* dalam dokumen *d*
- idf_t : *Invers* frekuensi dokumen dari *term t*
- N : Jumlah total dokumen dalam korpus
- df_t : Jumlah dokumen yang mengandung *term t*

Nilai bobot yang dihasilkan memiliki makna sebagai berikut :

- Nilai bobot 0 menunjukkan bahwa sebuah kata tidak ada dalam dokumen tersebut [30].

- Semakin tinggi nilai bobot TF-IDF, maka kata tersebut dianggap semakin penting atau signifikan untuk dokumen tersebut, karena sering muncul di dalamnya namun jarang ditemukan di dokumen-dokumen lain [30].

Dengan pembobotan ini, TF-IDF membantu dalam menurunkan bobot kata-kata umum dan menaikkan bobot kata-kata yang lebih informatif dan diskriminatif [24].

2.1.10 Data Splitting

Data splitting adalah proses pemisahan *dataset* menjadi beberapa bagian untuk keperluan pelatihan dan pengujian model dalam *machine learning*. Tujuan utama dari proses ini adalah untuk mengevaluasi performa model secara obyektif dan memastikan bahwa model mampu melakukan generalisasi dengan baik terhadap data yang belum pernah dilihat sebelumnya [14]. Pembagian *dataset* yang umum digunakan adalah data latih (*training data*) dan data uji (*testing data*), dengan proporsi yang bervariasi seperti 80:20 atau 70:30, tergantung pada ukuran *dataset* dan kebutuhan analisis [32].

2.1.11 Random Forest

Random Forest merupakan salah satu algoritma *machine learning* berbasis *ensemble learning* yang menggabungkan banyak *Decision Tree* untuk menghasilkan prediksi yang lebih akurat dan stabil [26]. Algoritma ini dikembangkan sebagai solusi atas keterbatasan yang dimiliki oleh *Decision Tree* tunggal. Meskipun *Decision Tree* dikenal intuitif dan mudah diinterpretasikan [33], algoritma ini memiliki kelemahan utama, yaitu sangat rentan mengalami *overfitting* [26].

Overfitting terjadi ketika model belajar terlalu spesifik pada data latih hingga ikut mempelajari pola yang tidak relevan, sehingga tidak mampu membuat prediksi yang akurat pada data baru [33]. Hal ini terjadi karena *Decision Tree*, jika tidak dibatasi, akan terus membuat cabang-cabang baru yang sangat spesifik untuk ”menghafal” setiap detail pada data latih. Proses ini membuatnya ikut mempelajari pola-pola yang tidak relevan, sehingga model kehilangan kemampuan generalisasi dan pada akhirnya berkinerja buruk saat diuji dengan data baru [34] [35].

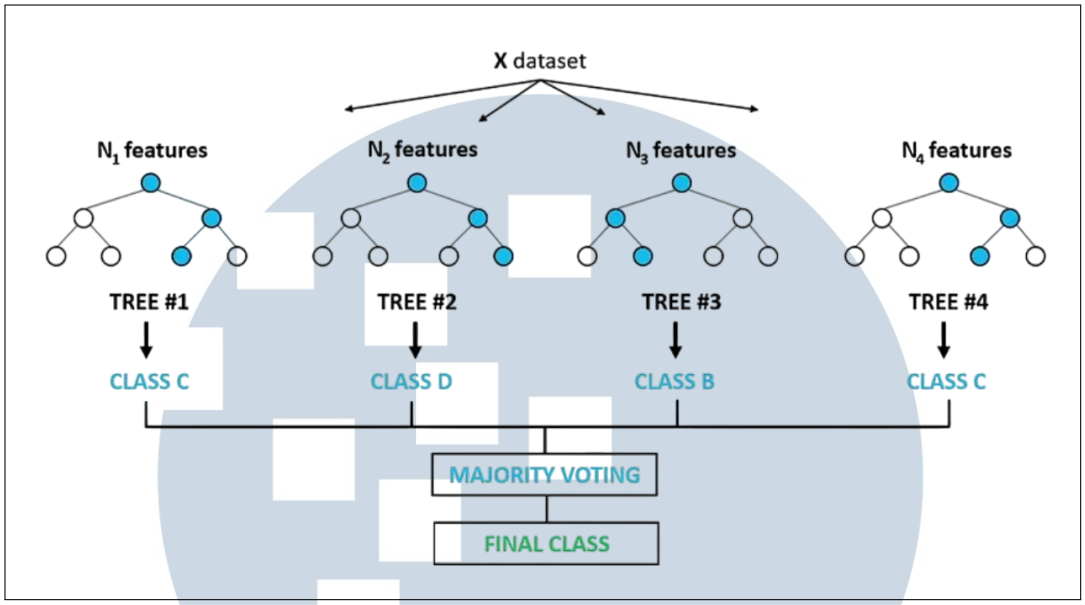
Untuk mengatasi masalah *overfitting* tersebut, *Random Forest* menerapkan dua strategi utama: *bootstrap aggregating (bagging)* dan pemilihan fitur acak. Pertama, *bagging* adalah proses pengambilan sampel acak dengan pengembalian

dari *dataset* pelatihan untuk membangun setiap pohon keputusan. Dengan cara ini, setiap pohon dilatih pada *subset* data yang sedikit berbeda, yang mendorong keragaman antar pohon [14]. Kedua adalah pemilihan fitur secara acak. Seperti yang diilustrasikan pada Gambar 2.1, setiap pohon (`TREE #1`, `TREE #2`, dst.) tidak menggunakan keseluruhan fitur, melainkan hanya dibangun menggunakan *subset* fitur acak yang berbeda (`N1 features`, `N2 features`, dst.). Dalam penelitian ini, fitur-fitur tersebut berasal dari representasi numerik TF-IDF. Teknik ganda ini secara efektif mengurangi korelasi antar pohon, meningkatkan kemampuan generalisasi model, dan membuatnya lebih tahan terhadap *overfitting* [15].

Proses kerja *Random Forest* secara visual dapat dipahami melalui alur pada Gambar 2.1. Proses dimulai dengan sebuah `X dataset` sebagai input utama. Dari dataset ini, dibangun sejumlah pohon keputusan independen, contohnya `TREE #1` hingga `TREE #4`. Setiap pohon kemudian membuat prediksinya sendiri terhadap data, menghasilkan kelas yang berbeda-beda seperti `CLASS C`, `CLASS D`, dan `CLASS B`. Semua hasil prediksi individual ini kemudian dikumpulkan dan diagregasi melalui tahap `MAJORITY VOTING` [16]. Pada contoh di gambar, terdapat dua suara untuk `CLASS C`, satu untuk `CLASS D`, dan satu untuk `CLASS B`, sehingga `FINAL CLASS` yang terpilih sebagai hasil akhir model adalah `CLASS C`. Pendekatan kolektif ini membuat kesalahan prediksi dari satu pohon dapat dikompensasi oleh suara mayoritas dari pohon-pohon lainnya, sehingga menghasilkan model yang jauh lebih stabil dan akurat dibandingkan jika hanya menggunakan satu *Decision Tree* [33].

Untuk membangun setiap pohon keputusan pada Gambar 2.1, prosesnya dimulai dari simpul akar (*root node*) yang berisi seluruh subset data pelatihan untuk pohon tersebut. Pada setiap simpul, algoritma secara berulang akan mencari percabangan (*split*) terbaik, yang didefinisikan sebagai pemisahan data yang mampu menghasilkan kelompok-kelompok turunan paling murni atau homogen [16]. Untuk menemukan *split* terbaik ini, algoritma melakukan pencarian terhadap kombinasi fitur dan nilai ambang batas (*threshold*) yang paling efektif [16].

Khusus untuk fitur numerik seperti nilai TF-IDF, proses pencarian *threshold* ini dilakukan secara sistematis. Pertama, algoritma akan mengurutkan semua nilai unik dari fitur tersebut [29]. Kemudian, setiap titik tengah di antara dua nilai berurutan diuji sebagai kandidat *threshold* [16] [36]. Untuk setiap kandidat *threshold*, data pada simpul akan dipisah menjadi dua kelas baru, dan kualitas pemisahan tersebut diukur secara kuantitatif menggunakan sebuah kriteria yang disebut Gini Index [15]. Kombinasi fitur dan *threshold* yang menghasilkan nilai Gini Index gabungan terendah akan dipilih sebagai pemisah terbaik untuk simpul



Gambar 2.1. Cara Kerja *Random Forest*
Sumber: [16]

tersebut [15][16]. Mekanisme inilah yang bekerja di setiap simpul (digambarkan sebagai lingkaran) hingga pohon selesai dibangun, dan dasar perhitungannya ditunjukkan pada Persamaan 2.2.

$$Gini(T) = 1 - \sum_{j=1}^{n} (p_j)^2 \tag{2.2}$$

Keterangan dari persamaan di atas adalah sebagai berikut [37]:

- $Gini(T)$: Nilai *Gini Index* untuk sebuah simpul (*node*) T.
- n : Jumlah total kelas dalam *dataset*.
- p_j : Proporsi atau probabilitas sampel yang termasuk dalam kelas j pada simpul T.

Nilai *Gini* berkisar antara 0 dan 1 [37]. Ketika sebuah simpul mencapai nilai Gini 0, hal ini menandakan bahwa simpul tersebut sudah murni sempurna (semua sampel di dalamnya milik satu kelas) [37]. Oleh karena itu, simpul ini tidak akan membuat percabangan lagi dan menjadi simpul daun (*leaf node*) yang menentukan hasil prediksi untuk jalur tersebut [37]. Sebaliknya, nilai Gini yang lebih tinggi menunjukkan tingkat ketidakmurnian yang lebih besar, sehingga simpul tersebut masih perlu dipecah [37].

Setelah semua pohon selesai dibangun dan setiap cabangnya mencapai *leaf node*, setiap pohon akan memberikan prediksinya. Proses agregasi dari seluruh prediksi ini, seperti yang dijelaskan pada Gambar 2.1, dapat direpresentasikan secara matematis. Proses pengambilan keputusan akhir berdasarkan suara terbanyak dari seluruh pohon ini disebut *majority voting*. Jika $\hat{C}_b(x)$ adalah hasil prediksi dari pohon ke-b untuk data input x, maka hasil akhir dari model *Random Forest*, $\hat{C}^B_{rf}(x)$, dapat dinyatakan sebagai berikut [15]:

$$\hat{C}^B_{rf}(x) = majority\ vote\{\hat{C}_b(x)\}^B_1 \tag{2.3}$$

Keterangan dari persamaan di atas adalah sebagai berikut:

- $\hat{C}^B_{rf}(x)$: Hasil prediksi akhir dari model *Random Forest* untuk data input x.
- *majority vote* : Fungsi untuk menentukan kelas yang mendapatkan suara terbanyak.
- $\hat{C}_b(x)$: Hasil prediksi dari satu pohon keputusan ke-b.
- B : Jumlah total pohon keputusan dalam *Random Forest*.
- x : Sampel data input yang akan diklasifikasikan.

Persamaan ini adalah representasi matematis dari tahap `MAJORITY VOTING` pada Gambar 2.1, di mana keputusan akhir diambil berdasarkan suara terbanyak dari seluruh B pohon yang telah dibangun. Berkat kombinasi strategi ini, *Random Forest* mampu menghasilkan prediksi yang tidak hanya akurat, tetapi juga sangat stabil dan kuat [33].

Kinerja dari model *Random Forest* ini dapat dikonfigurasi dan disesuaikan melalui beberapa parameter penting (*hyperparameters*) pada saat pelatihan model [16]:

- `n_estimators`: Jumlah pohon keputusan yang akan dibangun dalam hutan. Nilai default-nya adalah `100`, artinya jika tidak diatur, model akan secara otomatis membangun 100 pohon keputusan.
- `max_features`: Jumlah maksimum fitur yang dipertimbangkan saat mencari *split* terbaik di setiap simpul. Default-nya adalah `"sqrt"`, artinya tanpa pengaturan manual, jumlah fitur yang dipilih secara acak pada setiap *split* adalah akar kuadrat dari total fitur yang ada.

- `criterion`: Fungsi untuk mengukur kualitas *split*. Nilai default-nya adalah `"gini"`, artinya secara otomatis model akan menggunakan Gini Index sebagai metrik untuk menentukan pemisahan data terbaik.
- `max_depth`: Kedalaman maksimum yang diizinkan untuk setiap pohon. Default-nya adalah `None`, artinya jika tidak dibatasi, setiap pohon akan terus tumbuh sedalam mungkin hingga tiap daunnya berisi sampel dari satu kelas saja.
- `min_samples_split`: Jumlah minimum sampel yang dibutuhkan dalam sebuah simpul agar dapat dipecah lagi. Nilai default-nya adalah `2`, artinya sebuah simpul akan terus dipecah selama suatu simpul memiliki minimal 2 sampel data.
- `min_samples_leaf`: Jumlah minimum sampel yang harus ada pada sebuah simpul daun (*leaf node*) setelah pemisahan terjadi. Nilai default-nya adalah `1`, artinya sebuah pemisahan dianggap valid jika setiap simpul anak yang dihasilkannya memiliki setidaknya 1 sampel.
- `bootstrap`: Menentukan apakah teknik *bootstrap sampling* digunakan saat membangun pohon. Nilai default-nya adalah `True`, artinya secara otomatis setiap pohon akan dilatih pada sampel acak yang diambil dari dataset asli dengan metode pengembalian.
- `random_state`: Digunakan untuk mengontrol keacakan agar hasil eksperimen dapat direproduksi. Default-nya adalah `None`, artinya tanpa pengaturan spesifik, proses pengacakan akan berbeda setiap kali model dijalankan, sehingga hasilnya tidak akan sama persis.

2.1.12 Ensemble Learning

Ensemble learning adalah teknik dalam pembelajaran mesin yang menggabungkan beberapa model untuk meningkatkan akurasi dan mengurangi risiko *overfitting* [38]. Terdapat tiga metode utama dalam *ensemble learning*, yaitu:

1. *Bagging* (*Bootstrap Aggregating*): Metode ini membangun beberapa model dari algoritma yang sama pada berbagai *subset* data yang diambil secara acak dengan pengembalian (*random sampling with replacement*). Prediksi akhir

diperoleh dengan merata-ratakan hasil prediksi atau menggunakan *majority voting* [38].

2. *Boosting*: Berbeda dengan *bagging*, *boosting* membangun model secara berurutan, di mana setiap model berusaha memperbaiki kesalahan dari model sebelumnya [38].

3. *Stacking*: Metode ini menggabungkan prediksi dari berbagai algoritma untuk menghasilkan prediksi akhir yang lebih akurat menggunakan model *meta-learner* [38].

Metode *bagging* dipilih melalui penggunaan algoritma *Random Forest* karena kemampuannya yang telah terbukti efektif dalam menangkap pola bahasa negatif tanpa mudah mengalami *overfitting* [15][38].

2.1.13 Hyperparameter Tuning

Kinerja dari sebuah model *machine learning*, termasuk *Random Forest*, tidak hanya ditentukan oleh kualitas data atau pemilihan algoritma, tetapi juga sangat dipengaruhi oleh konfigurasi dari parameter-parameter yang telah dijelaskan sebelumnya [39]. Nilai parameter bawaan (*default*) seringkali tidak cukup untuk menghasilkan performa terbaik pada setiap kasus data yang unik [39]. Oleh karena itu, diperlukan sebuah proses sistematis untuk menemukan kombinasi nilai parameter yang paling efektif, yang dikenal sebagai *hyperparameter tuning* [40].

Hyperparameter tuning adalah proses pencarian kombinasi nilai hyperparameter yang optimal untuk sebuah model [40]. Tujuannya adalah untuk meningkatkan akurasi, kemampuan generalisasi, atau metrik performa lainnya dengan menyesuaikan parameter-parameter yang dapat diubah pada algoritma [40] [39]. Proses ini melibatkan pelatihan dan evaluasi model menggunakan berbagai kombinasi *hyperparameter* yang berbeda untuk memilih set kombinasi yang menghasilkan nilai kesalahan terendah [39]. Salah satu metode yang paling umum dan sistematis untuk melakukan proses ini adalah *GridSearchCV* [40].

2.1.14 GridSearchCV

GridSearchCV adalah metode *hyperparameter tuning* yang bekerja dengan cara yang sangat teliti dan menyeluruh [40]. Cara kerjanya adalah dengan melakukan pemindaian pada sejumlah *hyperparameter* yang telah dipilih [41].

Proses diawali dengan mendefinisikan sebuah *grid* atau ruang pencarian yang berisi daftar nilai-nilai spesifik untuk setiap hyperparameter yang ingin diuji [39].

Metode ini kemudian secara sistematis akan mencoba satu per satu semua kemungkinan kombinasi parameter dari *grid* tersebut [40]. Untuk memastikan validitas dan keandalan evaluasi, performa dari setiap kombinasi *hyperparameter* diukur menggunakan validasi silang (*cross-validation*) [**?**] [41]. Proses validasi silang ini sangat penting untuk menjaga agar performa model tetap andal saat dihadapkan pada data yang belum pernah dilihat sebelumnya selama tahap pelatihan [42]. Setelah semua kombinasi selesai dievaluasi melalui proses ini, *GridSearchCV* akan menentukan satu kombinasi parameter yang menghasilkan performa terbaik sebagai konfigurasi model yang optimal [39].

2.1.15 Evaluasi Model

Evaluasi model dilakukan untuk mengukur akurasi prediksi terhadap data berlabel. Salah satu metode umum adalah *confusion matrix*, yang menunjukkan jumlah prediksi benar dan salah [14]. Struktur ditampilkan pada Gambar 3.1.

		Predicted Class	
		(1) Positive	**(0) Negative**
Actual Class	**(1) Positive**	**TP** (True Positive)	**FN** (False Negative)
	(0) Negative	**FP** (False Positive)	**TN** (True Negative)

Gambar 2.2. Tabel Confusion Matrix

Sumber: [14]

Tabel ini terdiri dari empat komponen utama yang merepresentasikan hasil klasifikasi [16]:

- *True Positive* (TP): Jumlah data yang diklasifikasikan dengan benar sebagai kelas positif.
- *False Positive* (FP): Jumlah data yang secara keliru diklasifikasikan sebagai kelas positif, padahal seharusnya termasuk dalam kelas negatif.

- *True Negative* (TN): Jumlah data yang diklasifikasikan dengan benar sebagai kelas negatif.
- *False Negative* (FN): Jumlah data yang secara keliru diklasifikasikan sebagai kelas negatif, padahal seharusnya termasuk dalam kelas positif.

Berdasarkan nilai-nilai tersebut, beberapa metrik evaluasi dapat dihitung, seperti akurasi, *precision*, *recall*, dan *F1-score*. Masing-masing metrik memiliki fungsi yang berbeda dalam menilai performa model dan dapat dirumuskan sebagai berikut [14]:

- Akurasi digunakan untuk mengukur seberapa sering model membuat prediksi yang benar terhadap keseluruhan data yang diuji [14]. Rumus akurasi adalah:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{2.4}$$

- *Precision* mengukur proporsi data yang diprediksi sebagai positif dan benar-benar merupakan data positif [14]. Rumus *precision* adalah:

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

- *Recall* mengukur kemampuan model dalam mendeteksi semua data positif yang ada [14]. Rumus *recall* adalah:

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

- *F1-score* merupakan rata-rata harmonis antara *precision* dan *recall* [14]. Rumus *F1-score* adalah:

$$F1\text{-}Score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{2.7}$$

Dengan menghitung keempat metrik ini, performa model dapat dievaluasi secara komprehensif, memastikan bahwa model tidak hanya memiliki akurasi yang

tinggi tetapi juga mampu mengenali kelas atau kategori komentar secara akurat dan seimbang [29]. Evaluasi ini sangat penting untuk memahami kekuatan dan kelemahan model sebelum diterapkan secara nyata [29].

2.2 Jabaran Penelitian Terdahulu

Tabel menyajikan ringkasan penelitian terdahulu yang relevan, yang memberikan justifikasi metodologis untuk pemilihan algoritma *Random Forest*, *feature extraction TF-IDF*, dan pengaruh *hyperparameter tuning* pada performa *baseline*.

Tabel 2.1. Tabel Daftar Penelitian Terdahulu

Nama Penulis	Judul Penelitian	Metode dan Hasil Penelitian
Hasyim Al Fattah; Endang Wahyu Pamungkas [9]	Pendeteksian Ujaran Platform X dengan Seksisme pada Algoritma Machine Learning Tradisional	1. Kasus: Ujaran seksisme terhadap Nadin Amizah. 2. Data: 1.622 *tweet* dikumpulkan dengan *social network scraper*. 3. Preprocessing: Tahap lengkap termasuk *stemming*. 4. Fitur: Pembobotan TF-IDF. 5. Metode: SVM dan *Naïve Bayes* (rasio 70:30). 6. Hasil: Akurasi SVM 80,36%, *Naïve Bayes* 68,62%.
Heri Santoso, Raissa Amanda Putri, Sahbandi [14]	Deteksi Komentar Cyberbullying pada Media Sosial Instagram Menggunakan Algoritma Random Forest	1. Data: 1000 komentar Instagram, dilabeli manual (*bullying/non-bullying*). 2. Fitur: Ekstraksi menggunakan *FastText* karena sifat data implisit. 3. Metode: Klasifikasi *Random Forest* (rasio 80:20) dengan parameter default.

Tabel 2.1 – Tabel Daftar Penelitian Terdahulu (Lanjutan)

Nama Penulis	Judul Penelitian	Metode dan Hasil Penelitian
		4. Konfigurasi RF: *n_estimators* hingga 100, *max_features*='sqrt', *criterion*='gini'. 5. Hasil: Model mencapai akurasi terbaik sebesar 84%.
Rachmad Masbadi Hatullah Nurnaryo, dkk.[28]	Penerapan Random Forest dan Information Gain pada Deteksi Cyberbullying di Twitter	1. Fokus: Deteksi *cyberbullying* berbahasa Indonesia berdasarkan kata kunci eksplisit. 2. Fitur: *TF-IDF* dipilih karena deteksi sangat bergantung pada bobot kata. 3. Metode: Implementasi algoritma *Random Forest* dengan seleksi fitur. 4. Hasil: Akurasi 72,1%, membuktikan *TF-IDF* pendekatan yang kuat untuk kasus ini.
Laila Qadrini [22]	Undersampling dan K-Fold Random Forest Untuk Klasifikasi Kelas Tidak Seimbang	1. Fokus: Mengatasi masalah kelas tidak seimbang (*imbalanced class*). 2. Metode: Menerapkan teknik *undersampling* pada *Random Forest*. 3. Alasan: *Undersampling* terbukti memberikan akurasi lebih baik daripada *oversampling*.

Tabel 2.1 – Tabel Daftar Penelitian Terdahulu (Lanjutan)

Nama Penulis	Judul Penelitian	Metode dan Hasil Penelitian
		4. Hasil: *Recall* untuk *undersampling RF* 65%, sementara RF dengan *K-Fold* 83%.
Randy Himawan, Amalia Zahra [31]	Comparison of Deep Learning and Machine Learning Model for Phising Email Classification	1. Fokus: Perbandingan model untuk klasifikasi email *phishing*. 2. Hasil: *Random Forest* + TF-IDF (fitur 'Subject') mencapai akurasi 100%. 3. Perbandingan: Mengungguli *FastText* (akurasi 99,15%) pada skenario sama. 4. Kesimpulan: Membuktikan efektivitas superior TF-IDF untuk tugas ini.
Lakshmi Kalyani, dkk. [30]	A Comparative Analysis of Text Embeddings (TFIDF, Word2Vec, Fast Text) for Machine Learning Based Fake News Detection	1. Fokus: Membandingkan teknik *embedding* (TF-IDF, Word2Vec, FastText) untuk deteksi berita palsu. 2. Temuan: TF-IDF secara konsisten menghasilkan akurasi tertinggi di semua algoritma. 3. Hasil Puncak: Kombinasi *Random Forest* + TF-IDF mencapai akurasi 99,16%. 4. Perbandingan: Unggul signifikan dari *Word2Vec* (94,01%) dan *FastText* (94,88%).

Tabel 2.1 – Tabel Daftar Penelitian Terdahulu (Lanjutan)

Nama Penulis	Judul Penelitian	Metode dan Hasil Penelitian
Hardian Oktavianto, dkk. [43]	Analisis Komparasi Kinerja Metode Decision Tree dan Random Forest dalam Klasifikasi Teks Data Kesehatan	1. Fokus: Perbandingan performa *Decision Tree* dan *Random Forest* pada klasifikasi teks. 2. Hasil *Random Forest*: Secara konsisten unggul dan stabil, akurasi mencapai 99%. 3. Hasil *Decision Tree*: Hanya mampu stabil di kisaran akurasi 75%.
Lee Jia Thun, dkk. [44]	CyberAid: Are your children safe from cyberbullying?	1. Fokus: Perbandingan algoritma *machine learning* untuk klasifikasi *cyberbullying*. 2. Hasil: *Random Forest* mencapai akurasi tertinggi sebesar 92%. 3. Perbandingan: Mengungguli *SVM* (91%), *Decision Tree* (89%), dan *Naive Bayes* (86%).
Ririt Sheila Tina Rahmayani, dkk. [45]	Analisa Optimasi Grid Search pada Algoritma Random Forest dan Decision Tree untuk Klasifikasi Stunting	1. Fokus: Membuktikan keunggulan *Random Forest* (RF) atas *Decision Tree* (DT). 2. Sebelum Tuning: Akurasi RF (77,9%) mengungguli DT (70,2%).

Tabel 2.1 – Tabel Daftar Penelitian Terdahulu (Lanjutan)

Nama Penulis	Judul Penelitian	Metode dan Hasil Penelitian
		3. Setelah Grid Search: Akurasi RF (84,1%) tetap unggul atas DT (82,8%). Parameter uji RF: `n_estimators` [100, 200, 300], `max_depth` [10, 20, 30], `min_samples_split` [2, 5, 10], dan `min_samples_leaf` [1, 2, 4].
Nisa Amalia, Asmunin [40]	Prediksi Customer Churn Bank dengan Random Forest dan GridSearchCV	1. Fokus: Memprediksi nasabah bank yang akan *churn* akibat persaingan industri. 2. Metode: *Random Forest* yang dioptimasi dengan *Hyperparameter Tuning* menggunakan *GridSearchCV*. 3. Parameter uji untuk *tuning* meliputi: `n_estimators` [100, 200, 300, 400, 500], `max_depth` [5, 10, 15, 20, 25, 30], `max_features` ['auto', 'sqrt', 'log2'], `min_samples_split` [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], `min_samples_leaf` [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], dan `criterion` ['gini', 'entropy', 'log_loss'] 4. Hasil Akurasi: Model mencapai akurasi sebesar 89,61%.

Tabel 2.1 – Tabel Daftar Penelitian Terdahulu (Lanjutan)

Nama Penulis	Judul Penelitian	Metode dan Hasil Penelitian
		5. Hasil Metrik Lain: Recall 90,15%, Presisi 88,72%, dan F1-Score 89,43%.
Anna Baita, dkk. [39]	Hyperparameter Tuning on Random Forest for Diagnose COVID-19	1. Tujuan: Optimasi *Random Forest* dengan *GridSearchCV* untuk diagnosis *Covid-19*. 2. Parameter Uji: `n_estimators` [200-500] dan `max_depth` [4-8]. 3. Hasil Tuning: Kombinasi terbaik `n_estimators=300`, `max_depth=8`, `max_features='auto'`. 4. Peningkatan: Akurasi meningkat dari 94% (sebelum tuning) menjadi 96% (setelahnya). 5. Kesimpulan: Membuktikan efektivitas *hyperparameter tuning* via *GridSearchCV*.
Stephen Kurnia, dkk. [46]	Klasifikasi Tweet Cyberbullying dengan Menggunakan Algoritma Random Forest	1. Fokus: Menekankan pentingnya pemilihan *hyperparameter* untuk optimasi RF. 2. Hasil: Pengaturan `max_depth=350` & `n_estimators=100` mencapai akurasi **93,3%**. 3. Kesimpulan: Penyesuaian *hyperparameter* dapat meningkatkan performa *baseline Random Forest*.

Pemilihan metode dalam penelitian ini didasarkan pada tinjauan komprehensif terhadap penelitian-penelitian terdahulu yang relevan, seperti yang telah dirangkum pada tabel 2.1. Urgensi penelitian ini berawal dari keterbatasan studi sebelumnya yang dilakukan oleh Hasyim Al Fattah dkk. Cakupannya masih sebatas seksisme secara umum [9]. Hal ini membuka peluang untuk mengembangkan model yang lebih spesifik dalam mendeteksi komentar pelecehan seksual verbal. Mengingat karakteristik komentar pelecehan seksual seringkali tumpang tindih dengan *cyberbullying*, maka penelitian terkait deteksi *cyberbullying* yang dilakukan oleh Heri Santoso, dkk. menjadi acuan awal yang relevan dan valid dalam menentukan pendekatan yang digunakan dalam penelitian ini [14].

Langkah pertama dalam perancangan model adalah penanganan data. Kasus seperti deteksi komentar negatif seringkali menghasilkan *dataset* yang tidak seimbang (*imbalanced class*) [22]. Penelitian oleh Laila Qadrini [22] menunjukkan bahwa teknik *resampling* seperti *random undersampling* merupakan pendekatan yang efektif, di mana metode tersebut dipilih karena terbukti memberikan akurasi yang lebih baik dibandingkan *oversampling* dalam menangani bias pada data [22] [27].

Untuk algoritma klasifikasi, *Random Forest* dipilih sebagai solusi untuk mengatasi salah satu kelemahan fundamental dari *Decision Tree*, yaitu tingginya risiko *overfitting* [33]. Kecenderungan *Decision Tree* untuk tumbuh terlalu kompleks dan ”menghafal” data latih membuatnya kurang stabil [34] [35]. Hal ini dibuktikan dalam penelitian Hardian Oktavianto, dkk. [43], di mana *Decision Tree* hanya mampu mencapai akurasi di kisaran 75%. Sebaliknya, dengan strategi *ensemble*-nya yang menggabungkan banyak pohon, *Random Forest* mampu memitigasi risiko *overfitting* tersebut, sehingga menghasilkan model yang jauh lebih stabil dan akurat dengan pencapaian 99% pada studi kasus yang sama [43]. Keunggulan fundamental ini diperkuat oleh studi Ririt Sheila Tina Rahmayani, dkk. [45] yang menunjukkan *Random Forest* (84,1%) tetap superior bahkan setelah keduanya dioptimasi dengan *Grid Search*. Lebih lanjut, penelitian oleh Lee Jia Thun, dkk. [44] juga mengkonfirmasi bahwa *Random Forest* (92%) menjadi model dengan akurasi tertinggi dibandingkan *SVM* (91%), *Decision Tree* (89%), dan *Naive Bayes* (86%).

Setelah memilih algoritma, tahap selanjutnya adalah *feature extraction* untuk mengubah data teks menjadi format numerik yang dapat diproses oleh model. Sejalan dengan batasan penelitian ini yang berfokus pada deteksi komentar

pelecehan seksual verbal secara eksplisit berdasarkan kata kunci tertentu, maka *Term Frequency-Inverse Document Frequency* (TF-IDF) dipilih sebagai pendekatan yang paling tepat karena kemampuannya mengukur dan menonjolkan pentingnya kata-kata spesifik [28] [29]. Efektivitas pendekatan ini didukung oleh penelitian Rachmad Masbadi Hatullah Nurnaryo, dkk. [28] yang berhasil mencapai akurasi 72,1% pada kasus deteksi berbasis kata kunci. Keunggulan TF-IDF dalam hal akurasi juga dibuktikan secara komparatif oleh Lakshmi Kalyani, dkk. [30], di mana kombinasi *Random Forest* dengan *TF-IDF* (99,16%) secara signifikan mengungguli metode berorientasi makna semantik seperti *FastText* (94,88%) dan *Word2Vec* (94,01%). Temuan ini diperkuat pula oleh penelitian Randy Himawan dan Amalia Zahra yang menunjukkan bahwa *TF-IDF* (100%) sedikit lebih unggul dari *FastText* (99,15%), menjadikan TF-IDF metode paling andal untuk memaksimalkan akurasi pada tugas klasifikasi yang sensitif terhadap kehadiran kata kunci spesifik [31].

Terakhir, penelitian ini tidak hanya bertujuan untuk berhenti pada performa awal, seperti yang dilakukan oleh Heri Santoso, dkk. [14], tetapi berupaya untuk memaksimalkan potensi penuh dari algoritma *Random Forest*. Untuk mencapai performa terbaik tersebut, langkah esensial yang dilakukan adalah *hyperparameter tuning* guna mencari kombinasi parameter yang paling optimal. Pentingnya langkah ini telah dibuktikan secara kuantitatif oleh berbagai penelitian. Studi oleh Anna Baita, dkk. [39] menunjukkan bahwa optimasi dengan *GridSearchCV* mampu meningkatkan akurasi model dari 94% menjadi 96%. Demikian pula, penelitian oleh Ririt Sheila Tina Rahmayani, dkk. [45] juga menerapkan *Grid Search* untuk optimasi dan berhasil mencapai akurasi 84,1%. Selain itu, Nisa Amalia dan Asmunin [40] juga membuktikan efektivitas pendekatan ini dengan mencapai akurasi sebesar 89,61% dalam kasus prediksi nasabah *churn*. Temuan-temuan ini sejalan dengan kesimpulan dari Stephen Kurnia, dkk. [46] yang menekankan bahwa *hyperparameter* dapat mencapai performa puncak *Random Forest*. Metode *GridSearchCV* sendiri dipilih karena keunggulannya dalam mengevaluasi setiap kombinasi parameter secara sistematis menggunakan validasi silang (*cross-validation*), yang memastikan bahwa performa model yang dihasilkan lebih stabil dan andal saat dihadapkan pada data baru [42].