BAB 2
LANDASAN TEORI

2.1 Hate Speech dan Harassment

Harassment dalam dunia digital mengacu pada berbagai tindakan yang bertujuan untuk mengintimidasi, merendahkan, atau menyakiti seseorang melalui platform daring. Bentuk-bentuk *harassment* dapat mencakup ancaman, pelecehan verbal, penguntitan digital, hingga penyebaran ujaran kebencian melalui media sosial, forum daring, atau layanan pesan instan [10]. Penelitian menunjukkan bahwa *harassment* secara daring dapat memberikan dampak signifikan terhadap kesehatan mental korban, termasuk peningkatan kecemasan, depresi, hingga risiko gangguan psikologis yang lebih serius [11]. Studi lain juga menemukan bahwa korban *harassment* yang berusia muda cenderung mengalami efek jangka panjang terhadap kesejahteraan emosional mereka, terutama ketika pelecehan terjadi di ruang digital pribadi [12].

Salah satu bentuk *harassment* yang paling umum dan mudah terdeteksi secara tekstual adalah *hate speech* atau ujaran kebencian. *Hate speech* merujuk pada pernyataan atau ekspresi yang bertujuan untuk mendiskreditkan, merendahkan, atau menyerang individu maupun kelompok berdasarkan atribut tertentu seperti ras, agama, gender, atau orientasi seksual [13]. Dengan meningkatnya penggunaan media sosial dan platform komunikasi daring lainnya, penyebaran ujaran kebencian semakin meluas dan berpotensi menimbulkan konsekuensi sosial yang serius, seperti diskriminasi, kekerasan, bahkan genosida dalam beberapa kasus ekstrem [14].

Dalam konteks ini, hujatan atau makian menjadi salah satu bentuk *hate speech* yang paling eksplisit dan mudah diamati. Oleh karena itu, penelitian ini memfokuskan diri pada pengembangan sistem yang mampu mendeteksi hujatan secara otomatis sebagai langkah awal dalam menangani kasus *harassment* digital. Sistem semacam ini diharapkan dapat membantu dalam proses moderasi konten dan mencegah eskalasi kekerasan verbal yang terjadi di ruang daring.

2.2 Text Classification

Text classification atau klasifikasi teks merupakan salah satu tugas fundamental dalam bidang *Natural Language Processing* (NLP), yang telah banyak

digunakan untuk berbagai keperluan seperti analisis sentimen, deteksi spam, dan moderasi konten daring [15]. Dalam konteks ujaran kebencian, klasifikasi teks digunakan untuk membedakan antara konten yang bersifat ofensif dan non-ofensif [7].

Dalam konteks deteksi hujatan, klasifikasi teks digunakan untuk membedakan antara teks yang mengandung hujatan dengan teks yang tidak mengandung hujatan. Sistem klasifikasi akan dilatih menggunakan *dataset* yang telah diberi label (misalnya "hujatan" atau "bukan hujatan") agar mampu mempelajari pola-pola tertentu dalam data, seperti struktur kalimat, kata kunci, dan konteks kata yang sering muncul pada teks bernuansa kasar atau ofensif.

Klasifikasi teks umumnya dilakukan melalui beberapa tahapan utama, mulai dari *preprocessing* (pembersihan teks), *feature extraction* (ekstraksi fitur dari teks), hingga pelatihan model klasifikasi dengan algoritma pembelajaran mesin *Support Vector Machine* (SVM). Dengan pendekatan ini, sistem dapat digunakan untuk mengotomatisasi proses moderasi konten secara efisien dan konsisten dalam skala besar.

2.3 Dataset dan Validitas Data

Kualitas *dataset* merupakan salah satu faktor kunci dalam keberhasilan sistem berbasis *machine learning*, khususnya dalam tugas klasifikasi teks seperti deteksi hujatan. Dalam penelitian ini, data pelatihan terdiri dari kombinasi data yang diambil dari *dataset* publik dan data sintetik yang dihasilkan menggunakan model bahasa generatif seperti ChatGPT. Data sintetik disusun berdasarkan struktur dan konteks ujaran kebencian yang umum ditemukan dalam bahasa Indonesia, namun tidak melalui proses validasi langsung oleh ahli bahasa. Meski demikian, desain data mengikuti prinsip-prinsip anotasi yang umum digunakan dalam penelitian NLP, termasuk penggunaan label yang konsisten dan berbasis konteks.

Untuk mengukur performa model secara objektif, proses pengujian dilakukan menggunakan data uji yang telah melalui proses validasi oleh pihak ketiga, seperti institusi riset atau komunitas bahasa, yang secara tidak langsung menjamin kualitas anotasinya. Hal ini memungkinkan evaluasi model dilakukan terhadap data yang memiliki tingkat kepercayaan tinggi, sehingga jika model menunjukkan performa yang baik terhadap data uji tersebut, maka dapat disimpulkan bahwa data pelatihan (termasuk data sintetik) memiliki kualitas yang cukup representatif untuk tugas klasifikasi hujatan.

Dalam pengembangan *dataset* berbasis anotasi, validitas data tidak selalu harus berasal dari anotator ahli (*linguist*). Beberapa studi menunjukkan bahwa pendekatan berbasis *crowdsourcing* atau partisipasi non-ahli dapat menghasilkan data yang valid asalkan anotator diberikan panduan yang jelas (*annotation guideline*) dan contoh anotasi yang sesuai (*gold standard*). Penelitian oleh [16] dan [17] menunjukkan bahwa anotasi dari non-ahli dengan bimbingan yang tepat dapat mendekati kualitas anotasi dari para pakar. Pendekatan ini juga sering dilengkapi dengan penggunaan platform anotasi berbasis web untuk memastikan proses anotasi berlangsung secara sistematis dan dapat diaudit. Dengan kombinasi antara data sintetik yang dirancang secara terstruktur, penggunaan data uji yang telah divalidasi, serta pendekatan validasi melalui prinsip anotasi modern, *dataset* yang digunakan dalam penelitian ini dinilai layak untuk mendukung pembangunan sistem deteksi hujatan berbasis *Support Vector Machine*.

2.4 Data Sampling

Dalam penelitian berbasis data, proses pengambilan sampel merupakan tahapan penting untuk memastikan bahwa data yang digunakan merepresentasikan populasi secara proporsional dan dapat mendukung validitas hasil penelitian. Sampling diperlukan tidak hanya untuk pembagian data latih dan data uji, tetapi juga untuk keperluan evaluasi manual terhadap kualitas data.

Untuk keperluan pengecekan kualitas data latih secara manual, dilakukan pengambilan sampel dari data latih yang tersedia. Proses ini bertujuan untuk memverifikasi representasi data secara acak, serta mendeteksi kemungkinan kesalahan anotasi atau inkonsistensi dalam penyusunan *dataset*.

Jumlah sampel ditentukan menggunakan Rumus 2.1 yang diadopsi dari [18], yang umum digunakan dalam penentuan ukuran sampel dari populasi terbatas. Rumus tersebut adalah sebagai berikut:

$$n = \frac{N \cdot Z^2 \cdot p(1-p)}{(e^2 \cdot (N-1)) + Z^2 \cdot p(1-p)} \tag{2.1}$$

Dengan:

- n = jumlah sampel
- N = jumlah populasi

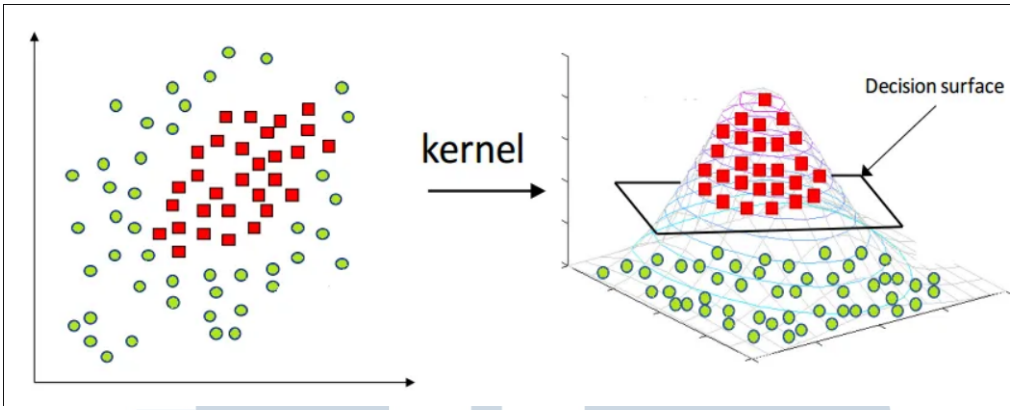
- Z = skor Z untuk *confidence level*
- p = proporsi populasi
- e = *margin of error*

2.5 Support Vector Machine

Support Vector Machine (SVM) merupakan algoritma pembelajaran mesin yang banyak digunakan dalam tugas klasifikasi, termasuk dalam klasifikasi teks. SVM bekerja dengan mencari sebuah *hyperplane* optimal yang dapat memisahkan dua kelas dalam ruang berdimensi tinggi. Model ini dikenal efektif dalam menangani data berdimensi besar yang dihasilkan dari teknik representasi seperti TF-IDF, dan memiliki kemampuan generalisasi yang baik meskipun jumlah data pelatihan relatif terbatas [8].

Salah satu keunggulan utama dari *Support Vector Machine (SVM)* adalah kemampuannya untuk menangani data yang tidak dapat dipisahkan secara linear dengan memanfaatkan teknik yang disebut *kernel trick*. *Kernel trick* memungkinkan pemetaan data dari ruang berdimensi rendah ke ruang berdimensi yang lebih tinggi tanpa harus menghitung koordinat titik-titik secara eksplisit, melainkan cukup melalui fungsi kernel $K(x_i, x_j)$ yang merepresentasikan hasil perkalian dalam ruang fitur.

Dengan kata lain, fungsi kernel menghitung kemiripan antara dua vektor dalam ruang fitur berdimensi tinggi tanpa perlu benar-benar memetakan data ke ruang tersebut. Teknik ini memungkinkan SVM untuk menemukan *hyperplane* pemisah yang optimal bahkan ketika data tidak *linearly separable* pada ruang asli seperti pada ilustrasi Gambar 2.1.



Gambar 2.1. Ilustrasi Kernel Trick: pemetaan data dari ruang dua dimensi ke tiga dimensi untuk mencapai separabilitas linear

Sumber: [19]

Berikut ini adalah beberapa jenis *kernel function* yang umum digunakan dalam praktik:

Tabel 2.1. Jenis-jenis Kernel yang Umum Digunakan dalam SVM

Jenis Kernel	**Fungsi dan Karakteristik**
Linear	$K(x,x') = x^\top x'$ Digunakan saat data diperkirakan dapat dipisahkan secara linear. Cepat dan sederhana.
Polynomial	$K(x,x') = (x^\top x' + c)^d$ Cocok untuk memodelkan hubungan non-linear dengan derajat tertentu.
RBF (Radial Basis Function)	$K(x,x') = \exp(-\gamma\|x - x'\|^2)$ Umum digunakan karena fleksibel dan mampu menangani kompleksitas tinggi.
Sigmoid	$K(x,x') = \tanh(\alpha x^\top x' + c)$ Serupa dengan fungsi aktivasi pada jaringan saraf, namun jarang digunakan.

Dalam penelitian ini, digunakan `LinearSVC`, yaitu varian dari SVM dengan kernel linear, karena representasi fitur menggunakan TF-IDF menghasilkan vektor berdimensi tinggi dan bersifat *sparse*, yang secara empiris terbukti mendukung pemisahan linear [5, 20].

Secara formal, tujuan dari SVM adalah memaksimalkan margin antara dua kelas, yang dirumuskan seperti Rumus 2.5 yang diadopsi dari [8].

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \forall i \tag{2.2}$$

Dimana:

- $\mathbf{w}$ adalah vektor bobot,
- b adalah bias atau intercept,
- $\mathbf{x}_i$ adalah vektor fitur dari data ke-i,
- $y_i \in \{-1, 1\}$ adalah label kelas dari data ke-i.

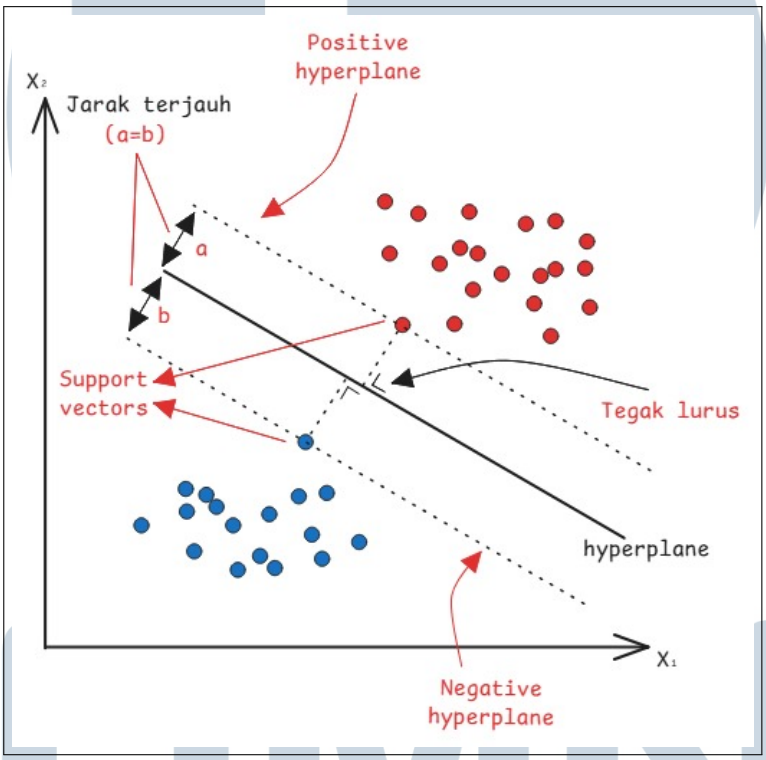
Ketika data tidak dapat dipisahkan secara sempurna (non-*separable*), digunakan versi *soft margin* dengan menambahkan variabel relaksasi ξ_i dan parameter regulasi C dan dirumuskan seperti Rumus 2.5 yang diadopsi dari [8].

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0 \tag{2.3}$$

Dalam penelitian ini, digunakan varian linear dari SVM, yaitu `LinearSVC`, yang merupakan implementasi SVM dengan *kernel* linear yang dioptimalkan untuk efisiensi pada *dataset* berdimensi sangat tinggi. Pemilihan `LinearSVC` didasarkan pada karakteristik representasi fitur yang digunakan, yaitu TF-IDF, yang menghasilkan vektor data dengan dimensi besar dan bersifat *sparse*. Representasi semacam ini cenderung membuat data menjadi *linear-separable*, sehingga pendekatan *kernel* linear dinilai cukup untuk memodelkan pola antar kelas secara efektif [5]. Selain itu, `LinearSVC` memiliki keunggulan dalam hal kecepatan pelatihan dan efisiensi memori, yang sangat relevan untuk diterapkan dalam sistem berbasis web.

Pada penelitian ini tidak dilakukan uji eksplisit terhadap linearitas data, karena secara umum, representasi TF-IDF menghasilkan ruang fitur berdimensi tinggi yang secara empiris telah terbukti mendukung *linear separability* [20]. Oleh karena itu, pendekatan SVM dengan *kernel* linear dipandang sebagai solusi praktis dan efektif untuk tugas klasifikasi teks berbasis hujatan, tanpa memerlukan pemetaan non-linear tambahan.

Studi sebelumnya juga mendukung penggunaan model SVM dengan *kernel* linear dalam konteks deteksi ujaran kebencian. Penelitian oleh [6] serta [7] menunjukkan bahwa kombinasi TF-IDF dan SVM mampu memberikan hasil klasifikasi yang akurat pada berbagai jenis bahasa, termasuk bahasa yang memiliki struktur kompleks. Oleh karena itu, pendekatan `LinearSVC` dalam penelitian ini dipilih sebagai metode utama untuk membangun sistem deteksi hujatan yang ringan, cepat, dan akurat.



Gambar 2.2. Ilustrasi cara kerja support vector machine

2.6 Feature Extraction dengan TF-IDF Vectorizer

Dalam penelitian ini, *feature extraction* dilakukan dengan menggunakan metode *Term Frequency-Inverse Document Frequency* (TF-IDF). TF-IDF digunakan untuk mengubah teks menjadi representasi numerik dengan mempertimbangkan bobot kata berdasarkan kepentingan kata tersebut dalam suatu kalimat. Penelitian ini menggunakan TF-IDF karena *support vector machine* (SVM) merupakan algoritma yang bekerja dengan vektor fitur atau representasi numerik dari suatu data. Data pada penelitian ini adalah kalimat sehingga akan sangat cocok jika dikombinasikan.

Dapat dilihat pada Tabel 2.2, TF-IDF merupakan pilihan yang tepat karena penelitian ini menggunakan *dataset* yang kecil dan mencari metode yang tidak memerlukan komputasi yang besar tapi dapat menghasilkan performa yang baik.

Tabel 2.2. Perbandingan TF-IDF dengan metode main dalam klasifikasi teks

Kriteria	**Deskripsi Metode**
TF-IDF	Memberi bobot pada kata berdasarkan frekuensi di dokumen dan seberapa jarang kata tersebut muncul di keseluruhan korpus. Cocok untuk model sederhana dan efektif pada dataset kecil hingga menengah [21].
Word2Vec	Membangun vektor kata dengan mempertimbangkan konteks kata di sekitarnya, sehingga memahami hubungan semantik. Namun, membutuhkan dataset besar dan pelatihan lebih lama agar akurat [22].
Bag of Words (BoW)	Mewakili teks sebagai sekumpulan kata tanpa mempertimbangkan urutan atau makna. Sangat sederhana dan cepat, tetapi kurang akurat karena semua kata diperlakukan sama [23].
FastText	Mirip Word2Vec, tetapi menambahkan pemahaman terhadap bagian dari kata (sub-word), sehingga lebih baik menangani kata baru atau salah eja. Lebih kompleks dan memerlukan komputasi lebih besar [24].

Secara matematis, TF-IDF terdiri dari dua komponen utama [21]:

1. *Term Frequency* (TF): Mengukur seberapa sering sebuah kata muncul dalam sebuah dokumen dengan Rumus 2.4.

$$TF(t) = \frac{\text{Jumlah kemunculan kata } t \text{ dalam dokumen}}{\text{Total jumlah kata dalam dokumen}} \tag{2.4}$$

Kata yang sering muncul dalam dokumen akan memiliki nilai TF yang lebih tinggi.

2. *Inverse Document Frequency* (IDF): Mengukur seberapa unik atau penting suatu kata dalam seluruh kumpulan dokumen dengan Rumus 2.5

$$IDF(t) = \log\left(\frac{\text{Total jumlah dokumen}}{\text{Jumlah dokumen yang mengandung kata } t}\right) \tag{2.5}$$

Kata yang muncul di banyak dokumen akan memiliki IDF yang lebih rendah, sementara kata yang jarang muncul akan memiliki IDF yang lebih tinggi.

Kombinasi dari kedua komponen ini dihitung dengan rumus:

$$TF-IDF(t) = TF(t) \times IDF(t) \tag{2.6}$$

Nilai ini digunakan sebagai bobot dalam representasi vektor teks, sehingga kata-kata yang lebih bermakna memiliki bobot yang lebih tinggi dibandingkan kata-kata umum yang sering muncul di semua dokumen [21].

Untuk implementasinya, penelitian ini menggunakan `TfidfVectorizer` dari pustaka *Scikit-learn*, yang memungkinkan pengaturan parameter seperti *max_features* (untuk membatasi jumlah fitur yang digunakan), *n-gram range* (untuk mempertimbangkan kombinasi kata), serta penghapusan *stopwords*. Dengan pendekatan ini, model pembelajaran mesin dapat memahami hubungan antar kata dengan lebih efektif, meningkatkan akurasi klasifikasi teks hujatan, dan mengurangi ketergantungan pada kata-kata yang tidak informatif.

2.7 Evaluasi Kinerja Model

Evaluasi model merupakan langkah penting dalam proses pembangunan sistem pembelajaran mesin untuk mengetahui sejauh mana model mampu memprediksi kelas dengan benar. Dalam penelitian ini, digunakan beberapa metrik evaluasi yang umum digunakan dalam tugas klasifikasi biner, yaitu akurasi, *precision*, *recall*, dan *F1-score*.

- **Akurasi** mengukur proporsi jumlah prediksi yang benar terhadap total jumlah data dengan Rumus 2.7 yang diadopsi dari [25]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

- **Precision** mengukur ketepatan prediksi positif yang dilakukan oleh model dengan Rumus 2.7 yang diadopsi dari [26]:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.8}$$

- **Recall** atau *sensitivity* mengukur kemampuan model dalam mendeteksi kelas positif secara benar dengan Rumus 2.7 yang diadopsi dari [26]:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.9}$$

- **F1-score** adalah rata-rata harmonis dari *precision* dan *recall* dengan Rumus 2.7 yang diadopsi dari [27]:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.10}$$

Evaluasi model juga dilengkapi dengan *confusion matrix* seperti pada Tabel 2.3 untuk memberikan gambaran distribusi prediksi benar dan salah pada masing-masing kelas. Penggunaan metrik yang beragam ini penting untuk mendapatkan pemahaman menyeluruh terhadap performa model, terutama dalam konteks data yang tidak seimbang. Dalam evaluasi model klasifikasi biner, hasil prediksi model dapat dianalisis menggunakan *confusion matrix* yang terdiri dari empat komponen utama, yaitu *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN). *True Positive* (TP) adalah jumlah kasus positif yang berhasil diklasifikasikan dengan benar sebagai positif oleh model. Sebaliknya, *False Positive* (FP) merupakan jumlah kasus negatif yang secara keliru diprediksi sebagai positif. *False Negative* (FN) adalah jumlah kasus positif yang salah diklasifikasikan sebagai negatif, sedangkan *True Negative* (TN) mencerminkan jumlah kasus negatif yang diprediksi benar sebagai negatif. Keempat komponen ini menjadi dasar dalam perhitungan berbagai metrik evaluasi seperti akurasi, *presisi*, *recall*, dan *F1-score* untuk menilai kinerja keseluruhan dari model klasifikasi.

Tabel 2.3. Confusion Matrix

Prediksi / Aktual	Positif	Negatif
Positif	True Positive (TP)	False Negative (FN)
Negatif	False Positive (FP)	True Negative (TN)

Pemilihan metrik-metrik ini didasarkan pada karakteristik permasalahan klasifikasi biner yang umum terjadi dalam deteksi hujatan, di mana kemungkinan terdapat ketidakseimbangan antara jumlah data hujatan dan bukan hujatan. Dalam konteks seperti ini, akurasi saja tidak cukup untuk merepresentasikan performa model secara menyeluruh.

Sebagai contoh, model yang hanya memprediksi seluruh data sebagai kelas mayoritas bisa saja menghasilkan akurasi tinggi, namun gagal mendeteksi kelas minoritas yang justru penting — dalam hal ini, hujatan. Oleh karena itu, *precision* digunakan untuk mengukur seberapa akurat prediksi terhadap kelas positif (hujatan), sedangkan *recall* mengukur kemampuan model dalam menemukan semua teks hujatan yang benar.

F1-score digunakan sebagai metrik gabungan yang seimbang, terutama berguna saat *trade-off* antara *precision* dan *recall* menjadi penting. Penggunaan metrik-metrik ini memungkinkan evaluasi performa model secara lebih holistik dan sesuai dengan tujuan penelitian, yaitu mendeteksi hujatan secara efektif tanpa mengorbankan presisi maupun sensitivitas model.

2.8 Hyperparameter Tuning

Hyperparameter tuning adalah proses optimasi nilai-nilai parameter luar (yang tidak dipelajari langsung oleh model) guna meningkatkan performa model pembelajaran mesin. Beberapa *hyperparameter* penting dalam algoritma SVM, seperti nilai regularisasi *C*, jumlah iterasi maksimal, dan toleransi kesalahan, dapat sangat mempengaruhi hasil prediksi model.

Dalam penelitian ini, digunakan `Optuna`, yaitu pustaka Python yang dirancang untuk melakukan optimasi *hyperparameter* secara otomatis dan efisien. Optuna menggunakan pendekatan *Tree-structured Parzen Estimator (TPE)* untuk mengeksplorasi ruang parameter dengan cara yang adaptif dan terarah.

Keunggulan utama `Optuna` terletak pada kemampuannya melakukan optimasi berbasis definisi tujuan (*define-by-run*), memungkinkan fleksibilitas dalam merancang ruang pencarian parameter, serta efisiensi dalam penggunaan sumber daya komputasi. Studi oleh Akiba et al. [28] menunjukkan bahwa `Optuna` mampu menemukan kombinasi parameter terbaik dengan jumlah iterasi yang lebih sedikit dibandingkan pendekatan konvensional seperti *grid search* atau *random search*. Dengan memanfaatkan `Optuna`, model yang dibangun dalam penelitian ini dapat dioptimalkan secara otomatis untuk mencapai performa klasifikasi terbaik.