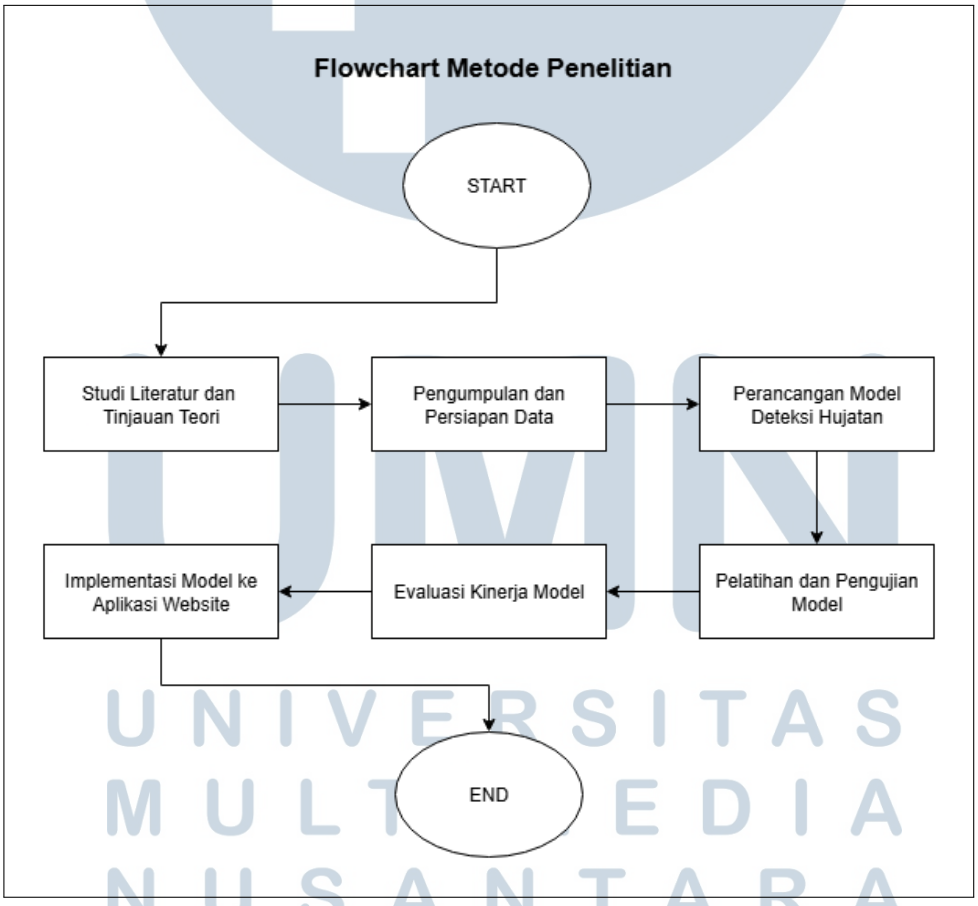
BAB 3
METODOLOGI PENELITIAN

Bab ini menjelaskan tahapan-tahapan penelitian yang dilakukan untuk membangun sistem deteksi hujatan berbasis algoritma Support Vector Machine. Setiap tahapan disusun secara sistematis mulai dari studi literatur, pengumpulan dan pengolahan data, perancangan model, pelatihan dan pengujian, hingga implementasi dan penyusunan laporan akhir.

Adapun tahapan metodologi penelitian ini divisualisasikan dalam bentuk diagram alur proses penelitian seperti ditunjukkan pada Gambar 3.1. Diagram tersebut merepresentasikan alur logis dari proses yang dilakukan secara berurutan untuk mencapai tujuan penelitian.



Gambar 3.1. Flowchart metode penelitian

3.1 Studi Literatur dan Tinjauan Teori

Studi literatur dilakukan untuk membangun fondasi konseptual dalam pengembangan sistem deteksi hujatan berbasis teks menggunakan pendekatan pembelajaran mesin. Penelaahan dilakukan terhadap berbagai sumber ilmiah seperti jurnal penelitian, artikel konferensi, dan dokumentasi teknis, yang mencakup lima aspek utama dalam penelitian ini: *dataset* dan validitasnya, algoritma klasifikasi yang digunakan, teknik ekstraksi fitur, metode evaluasi performa model, serta pendekatan *tuning hyperparameter*.

Pada aspek *dataset*, penelitian ini mengacu pada pendekatan validasi data berbasis *crowdsourcing*, yaitu pelabelan data oleh non-ahli dengan dukungan *annotation guidelines* yang jelas dan *gold standard*. Studi oleh Snow et al. [16] dan Sabou et al. [17] menunjukkan bahwa anotasi non-ahli dapat mencapai kualitas setara dengan anotator ahli jika proses dilakukan secara terstruktur. Dalam konteks ini, data pelatihan dalam penelitian ini sebagian disusun secara sintetik, namun dievaluasi secara tidak langsung melalui performa pada data uji yang telah divalidasi oleh pihak ketiga. Strategi ini diadopsi untuk meningkatkan efisiensi tanpa mengorbankan kualitas model yang dihasilkan.

Pada algoritma klasifikasi, studi sebelumnya menunjukkan bahwa SVM merupakan salah satu algoritma yang efektif untuk klasifikasi teks, terutama dalam menangani data berdimensi tinggi dan bersifat *sparse* seperti hasil dari ekstraksi TF-IDF. SVM dengan *kernel* linear menjadi pilihan utama karena efisien secara komputasi dan menunjukkan performa stabil pada representasi teks. Berdasarkan studi oleh Rennie et al. [20] dan Fortuna & Nunes [7], data teks yang direpresentasikan dengan TF-IDF umumnya menghasilkan distribusi vektor yang secara alami mendekati *linear separable*, sehingga tidak memerlukan transformasi *kernel* non-linear yang lebih kompleks.

Terkait fitur yang digunakan, penelitian ini menggabungkan beberapa pendekatan ekstraksi fitur untuk menangkap variasi bahasa dan pola hujatan secara lebih komprehensif. TF-IDF *word-level* digunakan untuk menangkap konteks kata secara eksplisit, sementara TF-IDF *char-level* efektif untuk mengenali variasi penulisan kasar atau modifikasi simbolik seperti "@nj!ng". Selain itu, pendekatan leksikal digunakan dengan menyisipkan fitur tambahan berupa jumlah kata hujatan dari kamus khusus. Pendekatan gabungan ini didukung oleh studi seperti Zhang et al. [29] dan Waseem & Hovy [6] yang menekankan pentingnya variasi representasi dalam deteksi ujaran kebencian.

Proses evaluasi performa model dilakukan menggunakan metrik klasifikasi seperti akurasi, *precision*, *recall*, dan *F1-score*. Metrik ini dipilih karena memberikan gambaran menyeluruh tentang keseimbangan antara kemampuan model mendeteksi hujatan dan menghindari kesalahan prediksi. Evaluasi didukung dengan penggunaan *confusion matrix* untuk menganalisis distribusi kesalahan klasifikasi antar kelas.

Setelah dilakukan evaluasi, untuk meningkatkan performa model, dilakukan proses *hyperparameter tuning* menggunakan pendekatan otomatis berbasis *Optuna*. Studi oleh Akiba et al. [28] membuktikan bahwa teknik optimasi berbasis *Tree-structured Parzen Estimator (TPE)* yang digunakan `Optuna` mampu mencari kombinasi parameter terbaik dengan efisiensi tinggi, terutama pada model berbasis SVM.

Dengan dasar teori yang diperoleh melalui studi literatur ini, seluruh tahapan penelitian mulai dari penyusunan *dataset*, pemilihan metode klasifikasi, desain fitur, evaluasi model, hingga *tuning* dilakukan dengan mengacu pada praktik terbaik yang telah terbukti dalam penelitian sebelumnya.

3.2 Pengumpulan dan Persiapan Data

Data yang digunakan dalam penelitian ini terdiri dari dua *subset* utama, yaitu data latih dan data uji. Seluruh data berupa teks berbahasa Indonesia yang diklasifikasikan ke dalam dua kelas, yaitu hujatan dan bukan hujatan. Data latih terdiri dari 9.600 data yang dibagi secara seimbang antara kelas hujatan dan bukan hujatan. Sebanyak 4.800 data hujatan dikembangkan secara sintetik menggunakan model bahasa generatif (ChatGPT) melalui *prompt* yang dirancang menyerupai ujaran kasar dalam berbagai variasi. Sementara itu, 4.800 data bukan hujatan diambil dari *dataset* yang digunakan dalam penelitian sebelumnya oleh [3], yang berisi teks umum dalam bahasa Indonesia dan telah dinyatakan bersih dari unsur hujatan. Kedua kelas ini digabungkan dan digunakan sebagai data pelatihan model klasifikasi.

Data uji terdiri dari 2.400 data (1.200 hujatan dan 1.200 bukan hujatan) yang diambil dari penelitian Ibrohim dan Budi [30]. Dataset ini telah melalui proses anotasi dan validasi yang ketat menggunakan pendekatan *crowdsourcing* terstruktur, di mana anotator non-ahli diberikan pelatihan, panduan anotasi, dan *gold standard* untuk memastikan kualitas label. Validasi ini didukung pula oleh pengawasan dari ahli bahasa, sehingga data uji dapat dianggap sebagai referensi

terpercaya untuk mengevaluasi performa model secara objektif. Secara komposisi dapat dilihat pada Tabel 3.1

Tabel 3.1. Komposisi Data Berdasarkan Kelas

Kategori	**Jumlah Data**	**Jenis Data**	**Sumber**
Hujatan	4.800	Data Latih	Sintetik dari ChatGPT
Bukan Hujatan	4.800	Data Latih	Dataset dari penelitian [3]
Hujatan	1.200	Data Uji	Dataset dari penelitian [30]
Bukan Hujatan	1.200	Data Uji	Dataset dari penelitian [30]

Berdasarkan sumber dan cara perolehannya, seluruh data yang digunakan dalam penelitian ini dapat dikategorikan sebagai data primer. Hal ini dikarenakan data latih disusun dan dikembangkan secara langsung oleh peneliti melalui proses sintesis dan kurasi manual, sedangkan data uji diperoleh dari studi terdahulu yang merupakan hasil anotasi manual berbasis *crowdsourcing* dan telah divalidasi oleh ahli. Tidak ada data sekunder atau hasil olahan turunan yang digunakan dalam eksperimen ini.

Untuk menjamin kualitas data latih, dilakukan proses *quality check* terhadap 370 data sampel yang diambil secara acak dari total data latih menggunakan Rumus 2.1 dengan tingkat kepercayaan 95% dan *margin of error* 5%. Sampel ini diperiksa secara manual untuk mendeteksi kemungkinan kesalahan sintesis dan ketidaksesuaian label.

Setelah penyusunan dan pemeriksaan data selesai, dilakukan pembagian *dataset* menjadi dua bagian: data latih (80%) dan data uji (20%) menggunakan pendekatan *stratified sampling*. Pembagian ini dilakukan untuk menjaga proporsi kelas tetap seimbang pada masing-masing *subset*.

Sebelum digunakan dalam proses pelatihan model, seluruh data melalui tahap *preprocessing* yang bertujuan untuk menormalisasi dan membersihkan teks agar dapat direpresentasikan secara numerik. Tahapan *preprocessing* yang dilakukan mencakup:

- Konversi teks menjadi huruf kecil (*lowercasing*)
- Penghapusan tanda baca, simbol, dan karakter non-alfabet
- Normalisasi kata tidak baku, kata alay, dan singkatan
- Penghapusan *stopwords* dalam bahasa Indonesia

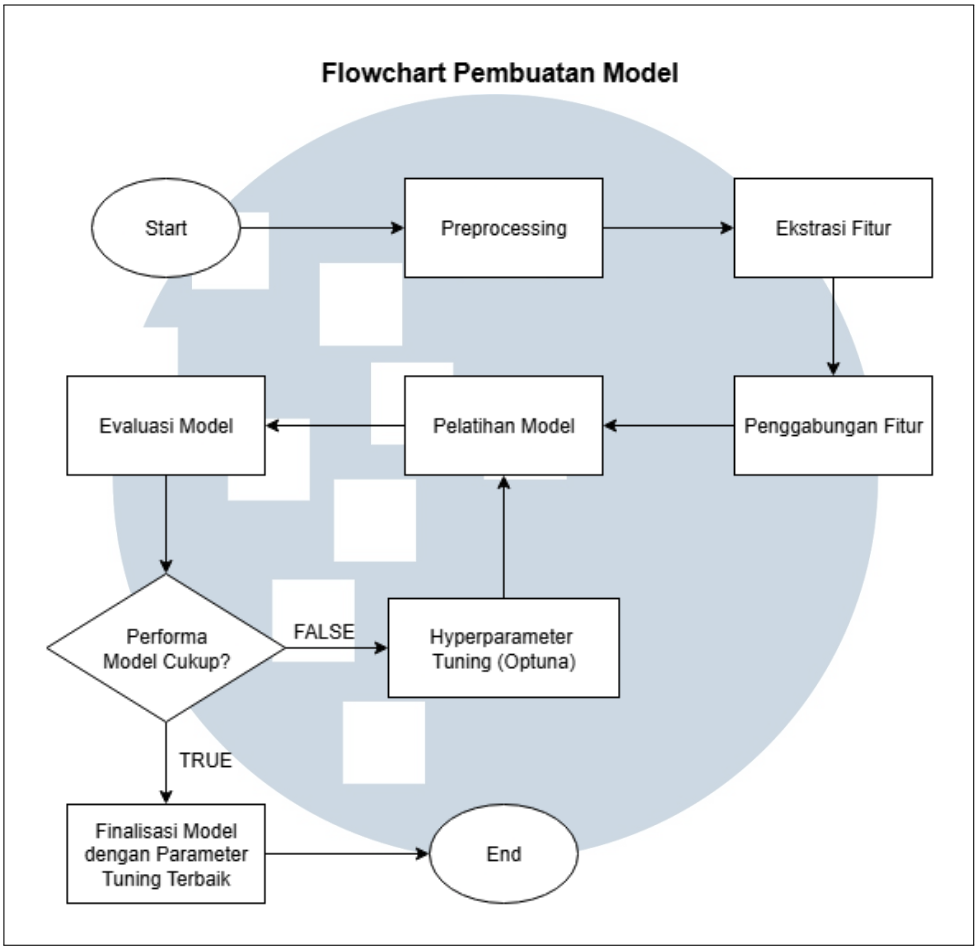
Seluruh proses *preprocessing* dilakukan secara terintegrasi di dalam *pipeline* pemodelan. Pendekatan ini diterapkan untuk menjaga konsistensi alur data dan mencegah risiko *data leakage* yang dapat memengaruhi hasil pelatihan dan evaluasi model.

3.3 Perancangan Model Deteksi Hujatan

Perancangan model deteksi hujatan dalam penelitian ini dilakukan melalui pendekatan berbasis *pipeline* terintegrasi yang mencakup tahap *preprocessing* teks, ekstraksi fitur, pelatihan model klasifikasi, hingga evaluasi performa model. Pendekatan ini dipilih untuk memastikan bahwa setiap proses pengolahan data dilakukan secara sistematis, konsisten, dan dapat direproduksi tanpa terjadi *data leakage*.

Pipeline model dirancang menggunakan bahasa pemrograman Python dengan memanfaatkan pustaka *scikit-learn*, yang menyediakan kerangka modular untuk membangun dan menggabungkan berbagai komponen pemrosesan teks dan model pembelajaran mesin. Setiap tahapan dalam *pipeline* dirancang untuk menangani karakteristik data teks bahasa Indonesia, serta untuk mengoptimalkan performa klasifikasi dalam mendeteksi ujaran hujatan.

Penjelasan lebih lanjut mengenai tahapan dalam perancangan model dijabarkan dalam subbab-subbab berdasarkan Gambar 3.2.



Gambar 3.2. Flowchart perancangan model

3.3.1 Preprocessing

Preprocessing teks merupakan tahap awal dalam perancangan model deteksi hujatan yang bertujuan untuk membersihkan dan menormalkan bentuk data teks agar lebih siap untuk dikonversi menjadi fitur numerik. Teks yang bersumber dari media sosial atau interaksi daring umumnya memiliki banyak variasi penulisan seperti kata tidak baku, singkatan, penggunaan karakter simbolik, serta elemen non-linguistik yang dapat mengganggu proses ekstraksi informasi.

Proses *preprocessing* dalam penelitian ini dilakukan secara otomatis sebagai bagian dari *pipeline*, sehingga seluruh data—baik data latih maupun data uji—diproses secara konsisten dan terintegrasi. Hal ini penting untuk menghindari terjadinya *data leakage* dan memastikan bahwa semua transformasi dilakukan dalam satu alur sistematis yang dapat direplikasi.

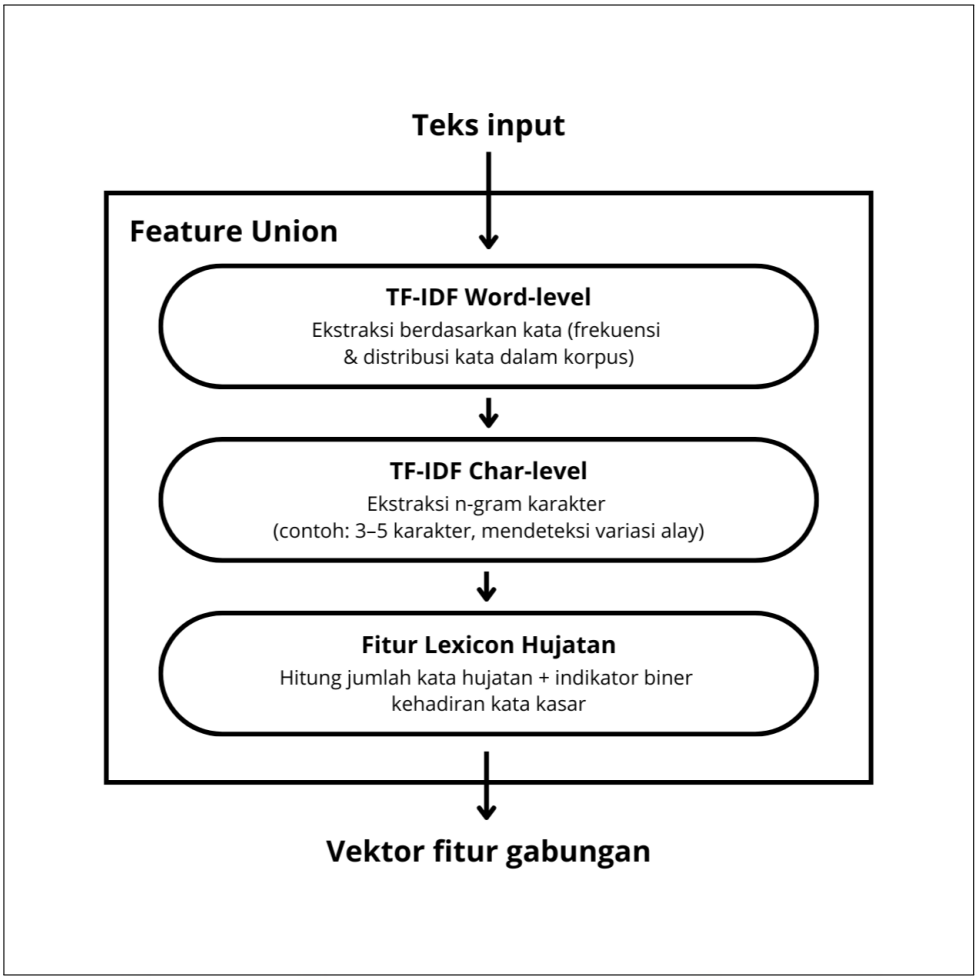
Tahapan *preprocessing* yang diterapkan antara lain:

- **Konversi huruf kecil (*lowercasing*)**: Seluruh teks diubah menjadi huruf kecil untuk mengurangi variasi yang disebabkan oleh perbedaan kapitalisasi.
- **Pembersihan simbol dan tanda baca**: Menghapus karakter non-alfabet seperti angka, emoji, simbol khusus, dan tanda baca yang tidak memiliki makna semantik dalam konteks deteksi hujatan.
- **Normalisasi kata tidak baku dan alay**: Melakukan substitusi terhadap kata-kata yang umum ditulis dalam bentuk tidak baku atau gaya alay menjadi bentuk standar bahasa Indonesia.
- **Penghapusan *stopwords***: Menghilangkan kata-kata umum (seperti "yang", "dan", "dengan") yang tidak memiliki kontribusi signifikan terhadap klasifikasi.

Seluruh proses *preprocessing* diimplementasikan menggunakan pustaka Python seperti *re* untuk pembersihan teks, serta pemetaan normalisasi berbasis kamus yang disusun dari sumber-sumber umum kata alay. Hasil dari tahap ini akan menjadi input bagi proses ekstraksi fitur yang dijelaskan pada subbab berikutnya.

3.3.2 Ekstraksi Fitur

Setelah melalui tahap *preprocessing*, teks perlu diubah menjadi representasi numerik agar dapat digunakan sebagai input bagi algoritma pembelajaran mesin. Proses ini disebut ekstraksi fitur, dan dalam penelitian ini dilakukan dengan pendekatan kombinatif yang menggabungkan representasi berbasis frekuensi dan berbasis leksikal untuk meningkatkan kemampuan model dalam mengenali berbagai bentuk hujatan seperti yang digambarkan pada Gambar 3.3.



Gambar 3.3. Diagram ekstraksi fitur

Terdapat tiga jenis fitur utama yang digunakan:

- **TF-IDF Word-level**: Menghitung bobot setiap kata berdasarkan frekuensinya dalam dokumen dan distribusinya dalam seluruh korpus. Pendekatan ini membantu model mengenali kata-kata penting dalam konteks hujatan.
- **TF-IDF Char-level**: Menghitung bobot n-gram berbasis karakter (misalnya 3–5 karakter). Fitur ini dirancang untuk menangkap pola penulisan yang dimodifikasi atau alay, seperti "@nj!ng" atau "b4n9s4t".
- **Fitur Leksikon Hujatan**: Menghitung jumlah kata hujatan yang terdapat dalam teks berdasarkan kamus kata kasar yang telah dikompilasi. Selain itu, ditambahkan fitur biner yang menunjukkan apakah terdapat minimal satu kata hujatan dalam teks.

Ketiga jenis fitur ini kemudian digabungkan menggunakan pendekatan *FeatureUnion* dari pustaka *scikit-learn*, sehingga membentuk satu vektor fitur terpadu untuk setiap sampel data. Penggabungan ini memungkinkan model untuk memanfaatkan keunggulan masing-masing tipe fitur secara simultan, yaitu sensitivitas terhadap konteks (word), fleksibilitas terhadap variasi penulisan (char), serta sinyal eksplisit keberadaan hujatan (leksikal).

Seluruh proses ekstraksi dan penggabungan fitur ini dilakukan secara otomatis di dalam *pipeline*, memastikan bahwa tidak ada perbedaan perlakuan antara data latih dan data uji serta mencegah risiko *data leakage*.

3.3.3 Pelatihan dan Evaluasi Model

Setelah teks dikonversi menjadi vektor fitur, proses selanjutnya adalah melatih model klasifikasi untuk mendeteksi apakah suatu teks mengandung hujatan atau tidak. Dalam penelitian ini, digunakan algoritma *Support Vector Machine (SVM)* dengan *kernel* linear yang diimplementasikan melalui kelas `LinearSVC` dari pustaka *scikit-learn*.

Pemilihan LinearSVC didasarkan pada efisiensinya dalam menangani data berdimensi tinggi dan *sparse*, seperti yang dihasilkan oleh TF-IDF. Selain itu, kernel linear dianggap cocok karena representasi TF-IDF umumnya menghasilkan data yang mendekati *linear separable*, sehingga penggunaan *non-linear kernel* tidak diperlukan. LinearSVC juga unggul dari segi kecepatan pelatihan dan prediksi, yang sangat penting untuk aplikasi berbasis web.

Proses pelatihan dilakukan menggunakan data latih sebanyak 80% dari total dataset, sedangkan 20% sisanya digunakan untuk evaluasi model. Pipeline dibangun secara terintegrasi, mulai dari preprocessing, ekstraksi fitur, penggabungan fitur, hingga pelatihan model, untuk memastikan konsistensi data dan menghindari *data leakage*.

Evaluasi awal dilakukan terhadap data uji menggunakan beberapa metrik klasifikasi, yaitu:

- **Akurasi**: mengukur proporsi prediksi yang benar terhadap seluruh data uji.
- **Precision**: mengukur seberapa tepat model dalam memprediksi teks hujatan.
- **Recall**: mengukur sejauh mana model berhasil menemukan semua teks hujatan yang benar.

- **F1-score**: gabungan harmonis antara precision dan recall.

Selain metrik tersebut, digunakan pula *confusion matrix* untuk menganalisis distribusi prediksi benar dan salah antar kelas. Hasil evaluasi awal ini menjadi dasar untuk menentukan apakah diperlukan proses *hyperparameter tuning*, yang akan dibahas lebih lanjut pada subbab berikutnya.

3.3.4 Optimasi Model dengan Hyperparameter Tuning

Untuk meningkatkan performa model, dilakukan proses optimasi melalui *hyperparameter tuning*. Tidak seperti parameter yang dipelajari oleh model secara langsung selama pelatihan, *hyperparameter* ditentukan sebelum pelatihan dan memiliki pengaruh signifikan terhadap kemampuan generalisasi model.

Dalam konteks algoritma *Support Vector Machine (SVM)* dengan *kernel* linear, beberapa *hyperparameter* penting yang dapat dioptimalkan antara lain:

- **C** (regularization parameter): mengontrol tingkat toleransi terhadap kesalahan klasifikasi. Nilai C yang lebih kecil menghasilkan margin yang lebih besar namun memungkinkan lebih banyak kesalahan.

- **max_iter**: jumlah maksimum iterasi dalam proses konvergensi model.

Untuk melakukan optimasi ini, digunakan pustaka *Optuna*, yaitu *framework hyperparameter optimization* berbasis *define-by-run* yang fleksibel dan efisien. Optuna menggunakan pendekatan *Tree-structured Parzen Estimator (TPE)* untuk mengeksplorasi ruang parameter secara adaptif, sehingga proses pencarian kombinasi parameter terbaik dapat dilakukan lebih efisien dibandingkan metode konvensional seperti *grid search* atau *random search*.

Proses *tuning* dilakukan dengan cara mendefinisikan fungsi objektif yang mengevaluasi performa model (menggunakan metrik *F1-score*) untuk setiap kombinasi *hyperparameter*. Optuna kemudian secara iteratif mencoba kombinasi yang berbeda dan memilih hasil terbaik berdasarkan nilai tertinggi dari fungsi objektif. *Hyperparameter* terbaik yang diperoleh dari proses *tuning* ini kemudian digunakan ulang untuk melatih model akhir. Proses pelatihan dan evaluasi ulang menggunakan parameter hasil *tuning* dijelaskan pada subbab berikutnya.

Hyperparameter tuning dilakukan setelah memperoleh *base model* untuk memastikan bahwa proses optimasi dilakukan atas model yang sudah memiliki struktur *pipeline* dan performa dasar yang stabil. Dengan membangun *base*

model terlebih dahulu, peneliti dapat mengetahui performa awal sistem dan menggunakan metrik tersebut sebagai tolok ukur peningkatan performa setelah *tuning*. Pendekatan ini juga umum digunakan dalam praktik dan penelitian pembelajaran mesin, seperti yang dijelaskan oleh Bergstra dan Bengio [31], yang menyatakan bahwa *tuning hyperparameter* hanya akan efektif bila dilakukan atas model yang sudah memiliki konfigurasi awal yang layak. Tanpa *baseline*, peningkatan performa tidak dapat diverifikasi secara objektif dan berisiko menghasilkan *tuning* yang tidak bermakna.

Selama proses *tuning hyperparameter* menggunakan Optuna, digunakan metode validasi silang (*cross-validation*) sebanyak 5-*fold* untuk memastikan bahwa evaluasi performa model dilakukan secara menyeluruh terhadap berbagai *subset* data latih. Metode ini membagi data latih menjadi lima bagian, kemudian melakukan pelatihan dan evaluasi secara bergantian pada setiap *fold*. Nilai rata-rata dari metrik yang diperoleh pada tiap *fold* dijadikan acuan dalam pemilihan kombinasi *hyperparameter* terbaik. Dengan pendekatan ini, risiko *overfitting* terhadap data latih dapat diminimalkan, serta performa model yang diperoleh lebih representatif terhadap data baru.

3.3.5 Evaluasi Lanjutan Model Setelah Tuning

Setelah diperoleh kombinasi *hyperparameter* terbaik melalui proses *tuning*, model dilatih ulang menggunakan parameter tersebut untuk memperoleh performa yang lebih optimal. Proses pelatihan dan evaluasi dilakukan kembali dengan *pipeline* yang sama, namun dengan pengaturan nilai-nilai *hyperparameter* hasil eksplorasi dari Optuna.

Evaluasi model dilakukan terhadap data uji yang sama seperti pada evaluasi awal, menggunakan metrik-metrik klasifikasi yang telah dijelaskan sebelumnya, yaitu akurasi, *precision*, *recall*, dan *F1-score*. Selain itu, digunakan kembali *confusion matrix* untuk memvisualisasikan sebaran prediksi benar dan salah setelah *tuning* dilakukan.

Dari hasil evaluasi lanjutan ini, diperoleh performa yang umumnya lebih baik dibandingkan dengan model awal sebelum *tuning*. Peningkatan ini dapat terlihat terutama pada metrik *recall* dan *F1-score*, yang menunjukkan bahwa model menjadi lebih sensitif dalam mendeteksi teks hujatan dan lebih seimbang antara ketepatan dan kelengkapan deteksi.

Perbandingan kuantitatif antara performa model awal dan model hasil *tuning*

akan dijelaskan lebih rinci pada Bab 4, bersamaan dengan pembahasan grafik dan tabel hasil pengujian.

3.4 Implementasi Model Pembelajaran Mesin Ke Aplikasi Website

Setelah model selesai diuji dan menunjukkan performa yang optimal, langkah selanjutnya adalah mengimplementasikan model ke dalam aplikasi berbasis *website* agar dapat digunakan secara *real-time* oleh pengguna. Dalam penelitian ini, sistem akan menggunakan *Flask* sebagai *backend* untuk menyediakan API yang akan berkomunikasi dengan *frontend* yang dibangun menggunakan HTML, CSS dan JavaScript.

Backend Flask akan memuat model SVM yang telah dilatih dan disimpan dalam format *file* (`.pkl`). Ketika pengguna memasukkan teks melalui antarmuka *website*, *frontend* akan mengirimkan teks tersebut ke *backend* melalui *request HTTP POST*. Flask kemudian akan melakukan *preprocessing* teks menggunakan *TF-IDF Vectorizer*, menjalankan prediksi menggunakan model SVM, dan mengembalikan hasil klasifikasi ke *frontend*. *Frontend* akan menerima respons dari API dan menampilkan hasil deteksi hujatan kepada pengguna dalam antarmuka yang responsif. Dengan pendekatan ini, sistem dapat berjalan secara fleksibel dan dapat diakses dari berbagai perangkat tanpa perlu menjalankan model langsung di sisi klien.