BAB 2
LANDASAN TEORI

2.1 Hate Speech

Hate speech merujuk pada ekspresi yang digunakan untuk menyerang, merendahkan, mendiskriminasi, atau menghasut individu maupun kelompok berdasarkan identitas sosial tertentu, misalnya suku, agama, ras, gender, atau karakteristik sosial lain. Dalam ekosistem digital, keberadaan konten seperti ini menjadi persoalan serius karena dampaknya tidak berhenti pada level bahasa, melainkan dapat merambat menjadi konflik sosial dan memperburuk kualitas interaksi publik [21, 22].

Penyebaran konten semacam itu makin sulit dikendalikan seiring meningkatnya aktivitas pada media sosial seperti *X (Twitter)*, *Facebook*, dan *Instagram*. Jumlah unggahan yang sangat besar membuat moderasi manual tidak lagi memadai, sehingga pendekatan otomatis berbasis *machine learning* diperlukan untuk membantu proses identifikasi secara lebih cepat dan konsisten [4, 23].

Pada penelitian *Natural Language Processing* (NLP), *hate speech* umumnya dikelompokkan ke dalam beberapa kategori berikut:

1. *Hate Speech* (ujaran kebencian langsung)
2. *Offensive Language* (bahasa kasar)
3. *Abusive Language* (bahasa menyerang individu)
4. *Non Hate Speech* (teks normal)

Klasifikasi *multiclass* semacam ini tidak hanya bertujuan membedakan ada atau tidaknya ujaran kebencian, tetapi juga membantu sistem mengenali bentuk ekspresi yang muncul dalam sebuah teks [24].

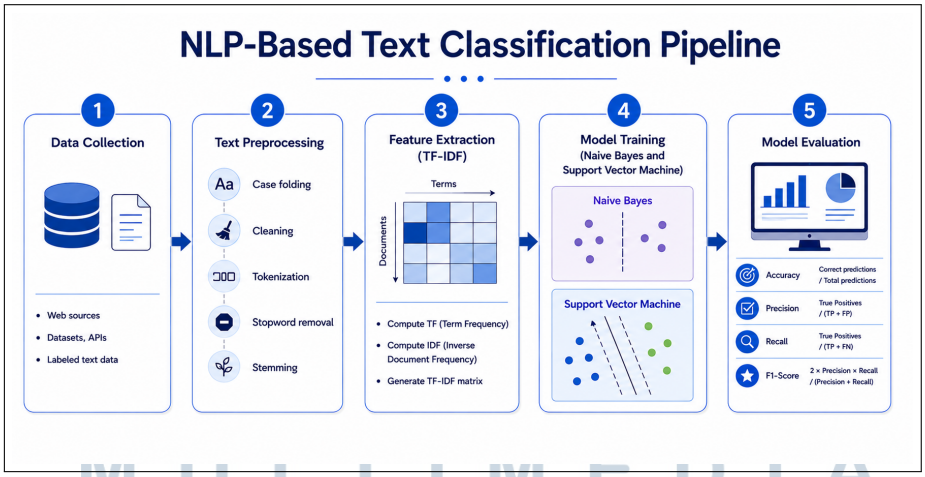
Sejumlah penelitian juga menunjukkan bahwa metode klasifikasi teks klasik masih cukup andal untuk mendeteksi konten bermuatan kebencian pada media sosial, terutama ketika data direpresentasikan dalam ruang fitur berdimensi tinggi [13, 23].

2.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) adalah cabang kecerdasan buatan yang berfokus pada pengembangan teknik agar komputer dapat memproses, menafsirkan, dan memahami bahasa manusia, baik dalam bentuk teks maupun ujaran [25]. Melalui pendekatan ini, data bahasa yang semula tidak terstruktur dapat diubah menjadi bentuk yang lebih siap dianalisis secara komputasional [26].

Pada tugas klasifikasi *hate speech*, NLP berfungsi sebagai kerangka kerja untuk mengubah teks mentah menjadi masukan yang dapat diolah model. Alur yang umum digunakan mencakup pengumpulan data (*data collection*), pembersihan dan normalisasi teks (*text preprocessing*), pembentukan representasi fitur (*feature extraction*), pelatihan model (*model training*), serta pengukuran kinerja (*model evaluation*) [23].

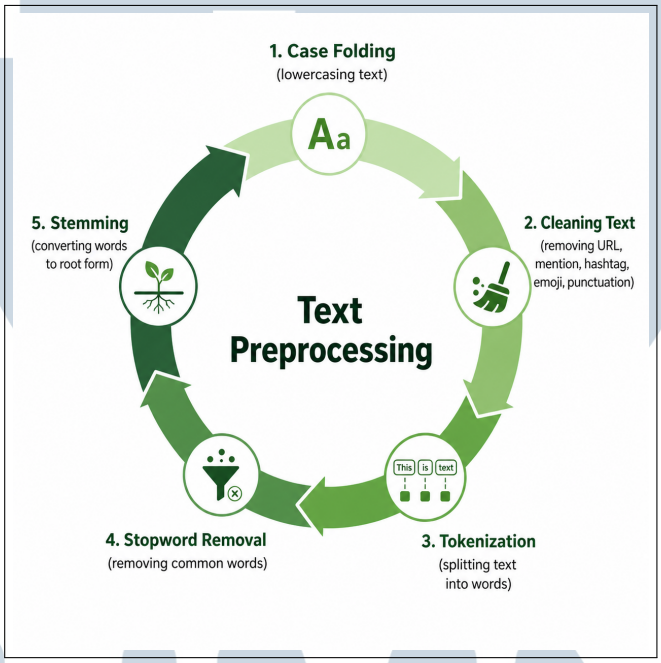
Tahap *preprocessing* menjadi bagian penting karena kualitas data sangat memengaruhi hasil klasifikasi. Pada media sosial, teks sering mengandung *noise* berupa *slang*, singkatan, penulisan tidak baku, atau campuran bahasa. Karena itu, data perlu dibersihkan terlebih dahulu sebelum dianalisis [26]. Gambar 2.1 menunjukkan alur sistem yang digunakan pada penelitian ini, dimulai dari pengumpulan data, *preprocessing*, ekstraksi fitur TF-IDF, pelatihan model, hingga evaluasi.



Gambar 2.1. *Pipeline* sistem deteksi *hate speech* berbasis *Natural Language Processing (NLP)*

2.3 Text Preprocessing

Text preprocessing adalah tahap penyiapan data yang bertujuan merapikan teks sebelum masuk ke proses analisis. Pada tahap ini, teks mentah yang masih mengandung simbol, karakter khusus, atau unsur yang tidak relevan dibersihkan agar representasi akhirnya menjadi lebih konsisten [27]. Rangkaian proses tersebut digambarkan pada Gambar 2.2.



Gambar 2.2. Tahapan text preprocessing dalam Natural Language Processing (NLP)

Berikut adalah tahapan *text preprocessing* yang dilakukan dalam penelitian ini:

1. *Case Folding*
   Tahap ini bertujuan untuk menyeragamkan seluruh karakter teks menjadi huruf kecil (*lowercase*). Hal ini dilakukan agar mesin tidak membaca kata yang sama sebagai dua entitas berbeda hanya karena perbedaan huruf kapital. Contoh: `GOBLOK` → `goblok`

2. *Data Cleaning* (Pembersihan Teks)
   Proses untuk menyingkirkan elemen-elemen yang tidak memiliki nilai semantik atau tidak diperlukan dalam proses klasifikasi, seperti *mention*, *hashtag*, *emoticon/emoji*, URL/tautan, dan tanda baca.

3. *Tokenization*

   Tahap pemecahan kalimat atau dokumen teks yang utuh menjadi satuan-satuan kata tunggal yang disebut token.

   Contoh: `"dasar goblok banget"` → `["dasar", "goblok", "banget"]`

4. *Stopword Removal*

   Tahap penyaringan dengan menyingkirkan kata-kata umum yang memiliki frekuensi kemunculan tinggi namun kurang bermakna bagi proses klasifikasi, seperti *di, ke, yang, dari,* dan *dan*.

5. *Stemming*

   Proses pemotongan imbuhan (awalan, sisipan, akhiran) pada sebuah kata untuk mengembalikannya ke dalam bentuk kata dasar.

   Contoh: `membenci` → `benci`

Untuk bahasa Indonesia, proses *stemming* umumnya memanfaatkan pustaka *Sastrawi* [27]. Secara keseluruhan, tahapan-tahapan tersebut membantu mengurangi *noise* dan membuat teks lebih seragam, sehingga data yang masuk ke model menjadi lebih representatif [28].

2.4 Term Frequency – Inverse Document Frequency (TF-IDF)

TF-IDF merupakan metode *feature extraction* yang mengubah teks menjadi representasi numerik [29]. Komponen penyusun TF-IDF terdiri dari:

1. *Term Frequency (TF)*: frekuensi kemunculan kata dalam dokumen
2. *Inverse Document Frequency (IDF)*: tingkat kepentingan kata dalam korpus

Pendekatan ini efektif untuk data berdimensi tinggi dan membantu menurunkan pengaruh kata yang terlalu umum muncul dalam korpus. Oleh sebab itu, TF-IDF sering digunakan bersama model klasifikasi teks yang bekerja pada ruang fitur besar [29, 30].

$$TFIDF(t,d) = TF(t,d) \times IDF(t) \tag{2.1}$$

dengan komponen IDF yang dihitung menggunakan Persamaan 2.2.

$$IDF(t) = \log\left(\frac{N}{df(t)}\right) \tag{2.2}$$

Keterangan:

1. $TF(t,d)$ menyatakan frekuensi kemunculan *term* t pada dokumen d.
2. N merupakan jumlah seluruh dokumen dalam *dataset*.
3. $df(t)$ menunjukkan jumlah dokumen yang memuat *term* t.

2.5 Feature Selection

Feature Selection atau seleksi fitur merupakan proses memilih fitur-fitur yang paling relevan terhadap kelas target serta menghilangkan fitur yang tidak memberikan kontribusi signifikan terhadap proses klasifikasi. Pada klasifikasi teks, jumlah fitur yang dihasilkan dari representasi TF-IDF umumnya sangat besar karena setiap kata unik dalam *corpus* direpresentasikan sebagai sebuah fitur. Kondisi tersebut menyebabkan permasalahan dimensi tinggi (*high dimensionality*) yang dapat meningkatkan kompleksitas komputasi, kebutuhan memori, serta risiko *overfitting*. Oleh karena itu, seleksi fitur diterapkan untuk mengurangi dimensi data tanpa menghilangkan informasi penting yang dibutuhkan model dalam proses klasifikasi [31].

Pada penelitian ini digunakan dua metode seleksi fitur, yaitu *Chi-Square* dan *Mutual Information*. Kedua metode tersebut termasuk ke dalam kategori metode *filter*, yaitu metode yang mengevaluasi relevansi setiap fitur terhadap kelas target secara independen sebelum proses pelatihan model dilakukan. Dibandingkan metode *wrapper* maupun *embedded*, metode *filter* memiliki kompleksitas komputasi yang lebih rendah sehingga lebih efisien diterapkan pada data teks berdimensi tinggi [32].

2.5.1 Chi-Square

Chi-Square (χ^2) merupakan metode seleksi fitur dalam kategori *filter* yang mengukur tingkat ketergantungan antara kemunculan suatu *term* dengan kelas target [31]. Semakin tinggi nilai χ^2 suatu *term* terhadap kelas tertentu, semakin besar ketergantungan antara *term* tersebut dengan kelas tersebut, sehingga *term* tersebut dianggap semakin relevan untuk dipertahankan sebagai fitur. Sebaliknya, nilai χ^2 yang rendah menandakan kemunculan *term* tidak berhubungan secara signifikan dengan kelas manapun, sehingga fitur tersebut dapat dibuang dari model [33].

Nilai χ^2 untuk setiap *term* t terhadap kelas c dapat dihitung menggunakan tabel kontingensi 2×2 dengan persamaan berikut:

$$\chi^2(t,c) = \frac{N(AD-BC)^2}{(A+B)(C+D)(A+C)(B+D)} \tag{2.3}$$

Keterangan:

1. A menunjukkan jumlah dokumen pada kelas c yang mengandung *term* t.
2. B menunjukkan jumlah dokumen di luar kelas c yang mengandung *term* t.
3. C menunjukkan jumlah dokumen pada kelas c yang tidak mengandung *term* t.
4. D menunjukkan jumlah dokumen di luar kelas c yang tidak mengandung *term* t.
5. N menunjukkan jumlah keseluruhan dokumen yang digunakan, yaitu $N = A+B+C+D$.

Fitur yang memiliki nilai χ^2 di atas ambang batas tertentu, atau berada pada peringkat teratas berdasarkan skor χ^2-nya, akan dipertahankan sebagai fitur masukan bagi model klasifikasi, sementara fitur lainnya dibuang.

2.5.2 Mutual Information

Mutual Information (MI) adalah metode seleksi fitur yang mengukur seberapa besar informasi yang diperoleh mengenai keberadaan suatu kelas dari keberadaan suatu *term*, dan sebaliknya. Berbeda dengan *Chi-Square* yang bersifat normalisasi statistik, *Mutual Information* berangkat dari teori informasi sehingga nilainya menggambarkan besarnya ketergantungan (*dependency*) antara distribusi kemunculan *term* dan distribusi kelas [34].

Nilai *Mutual Information* antara *term* t dan kelas c dapat dihitung dengan persamaan berikut:

$$I(t,c) = \log_2 \frac{A \times N}{(A+C)(A+B)} \tag{2.4}$$

Keterangan:

1. A menunjukkan jumlah dokumen pada kelas c yang mengandung *term* t.
2. B menunjukkan jumlah dokumen di luar kelas c yang mengandung *term* t.
3. C menunjukkan jumlah dokumen pada kelas c yang tidak mengandung *term* t.
4. N menunjukkan jumlah keseluruhan dokumen yang digunakan.

Semakin tinggi nilai $I(t,c)$, semakin besar informasi yang diberikan oleh *term* t terhadap kelas c, sehingga *term* tersebut semakin layak dipertahankan sebagai fitur. Nilai $I(t,c)$ dihitung untuk setiap pasangan *term* dan kelas, kemudian fitur-fitur dengan skor tertinggi dipilih sebagai representasi akhir dari *dataset* sebelum masuk ke tahap klasifikasi [34].

2.6 Cross Validation

Cross Validation adalah teknik evaluasi untuk menilai kemampuan model saat berhadapan dengan data yang tidak dipakai selama pelatihan [35]. Caranya adalah membagi *dataset* ke beberapa bagian, lalu menggunakan bagian-bagian tersebut secara bergantian sebagai data latih dan data validasi. Dengan skema ini, kinerja model dapat diamati secara lebih menyeluruh dibandingkan hanya memakai satu kali pembagian data.

Penggunaan teknik ini membantu menghasilkan evaluasi yang lebih stabil, menekan risiko *overfitting*, dan memberikan estimasi performa yang lebih mendekati kondisi saat model menghadapi data baru [36]. Karena alasan tersebut, *cross validation* sering dipakai pada tahap pemilihan model maupun optimasi parameter.

Dalam penelitian ini digunakan *Shuffle Split Cross Validation*. Pada setiap iterasi, *dataset* diacak terlebih dahulu sebelum dipisahkan menjadi data pelatihan dan data validasi. Pengacakan berulang memungkinkan model diuji pada berbagai kombinasi data, sehingga hasil evaluasi tidak terlalu bergantung pada satu skenario pembagian tertentu, sebagaimana diimplementasikan pada pustaka *scikit-learn* melalui fungsi *ShuffleSplit*.

2.7 Hyperparameter Tuning

Hyperparameter Tuning adalah tahap optimasi untuk mencari konfigurasi *hyperparameter* yang paling sesuai agar performa model meningkat [37]. Berbeda

dari parameter model yang dipelajari otomatis selama pelatihan, *hyperparameter* ditentukan sebelum proses belajar dimulai dan berfungsi mengatur cara model mempelajari data.

Beberapa teknik yang umum dipakai pada tahap ini antara lain *Grid Search*, *Random Search*, dan *Bayesian Optimization*. Masing-masing memiliki pola pencarian yang berbeda, baik dari sisi efisiensi maupun peluang menemukan kombinasi parameter yang baik [38].

Pada penelitian ini, optimasi dilakukan menggunakan *Grid Search* melalui *GridSearchCV*. Metode ini menguji seluruh kombinasi nilai *hyperparameter* yang telah ditetapkan sebelumnya, kemudian memilih kombinasi dengan performa terbaik berdasarkan metrik evaluasi yang digunakan [39]. Pendekatan ini dipilih karena sederhana untuk diterapkan dan masih efisien ketika ruang pencarian parameter belum terlalu besar.

2.8 Naive Bayes

Naive Bayes merupakan metode klasifikasi probabilistik yang didasarkan pada Teorema Bayes. Prinsip utamanya adalah menghitung peluang suatu data termasuk ke dalam kelas tertentu berdasarkan fitur yang dimilikinya. Istilah *naive* dipakai karena metode ini mengasumsikan bahwa setiap fitur saling bebas satu sama lain [30]. Dalam klasifikasi teks, pendekatan ini banyak digunakan karena komputasinya ringan dan tetap dapat bekerja pada jumlah fitur yang besar.

Secara umum, Teorema Bayes dapat dinyatakan sebagai berikut:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \tag{2.5}$$

Keterangan:

1. $P(C|X)$ menunjukkan probabilitas data X termasuk ke dalam kelas C.
2. $P(X|C)$ menunjukkan probabilitas kemunculan data X dengan kondisi kelasnya adalah C.
3. $P(C)$ menunjukkan probabilitas awal dari kelas C.
4. $P(X)$ menunjukkan probabilitas kemunculan data X pada keseluruhan data.

Pada klasifikasi teks, data X terdiri atas sekumpulan fitur berupa kata atau *term* yang telah direpresentasikan sebagai vektor numerik. Dengan asumsi independensi antarf fitur, penentuan kelas dapat dituliskan sebagai berikut:

$$\hat{C} = \arg\max_{C} P(C) \prod_{i=1}^{n} P(x_i|C) \tag{2.6}$$

Pada persamaan tersebut, $\hat{C}$ menyatakan kelas hasil prediksi, C adalah kandidat kelas, x_i merupakan fitur ke-i, dan n menunjukkan banyaknya fitur dalam dokumen. Kelas dengan probabilitas tertinggi dipilih sebagai hasil klasifikasi.

Varian yang paling umum dipakai pada klasifikasi teks adalah *Multinomial Naive Bayes*. Varian ini sesuai untuk data yang dinyatakan sebagai frekuensi atau bobot fitur, termasuk hasil ekstraksi TF-IDF. Probabilitas suatu fitur terhadap kelas tertentu dapat dihitung dengan persamaan berikut:

$$P(x_i|C) = \frac{N_{x_i,C} + \alpha}{N_C + \alpha|V|} \tag{2.7}$$

Keterangan:

1. $N_{x_i,C}$ menunjukkan jumlah kemunculan atau bobot fitur x_i pada kelas C.
2. N_C menunjukkan total kemunculan atau bobot seluruh fitur pada kelas C.
3. α merupakan parameter *smoothing* yang digunakan untuk mencegah nilai probabilitas menjadi nol.
4. $|V|$ menunjukkan jumlah seluruh fitur atau kosakata yang terdapat pada data.

Beberapa kelebihan metode ini adalah sebagai berikut:

1. Memiliki proses *training* yang relatif cepat.
2. Mudah diterapkan pada berbagai kasus klasifikasi.
3. Cukup efektif digunakan pada klasifikasi teks.
4. Mampu bekerja pada data dengan dimensi fitur yang tinggi.

Di sisi lain, pendekatan ini juga memiliki beberapa keterbatasan:

1. Asumsi bahwa setiap fitur bersifat independen tidak selalu sesuai dengan kondisi data sebenarnya.

2. Hasil klasifikasi dapat dipengaruhi oleh distribusi data.

3. Performa model dapat menurun apabila terdapat hubungan yang kuat antarfitur.

Pada penelitian ini, varian *Multinomial Naive Bayes* digunakan untuk membedakan teks berbahasa Indonesia ke dalam dua kategori, yaitu *hate speech* dan *non-hate speech*. Sebelum masuk ke model, data terlebih dahulu melalui tahap *preprocessing* dan ekstraksi fitur menggunakan TF-IDF.

2.9 Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan algoritma klasifikasi yang bekerja dengan mencari *hyperplane* optimal untuk memisahkan kelas data dengan margin maksimum. Karakteristik tersebut membuat SVM memiliki kemampuan generalisasi yang baik pada data berdimensi tinggi, sehingga banyak diterapkan dalam berbagai tugas klasifikasi teks, termasuk pada representasi fitur berbasis TF-IDF [40].

Secara umum, *hyperplane* pada SVM dapat dituliskan sebagai berikut:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2.8}$$

Keterangan:

1. **w** menunjukkan vektor bobot yang menentukan arah *hyperplane*.
2. **x** menunjukkan vektor fitur dari data masukan.
3. b menunjukkan nilai bias.

Pada proses klasifikasi, SVM menentukan label data berdasarkan posisi titik data terhadap *hyperplane*. Fungsi keputusannya dapat dituliskan sebagai berikut:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.9}$$

Jika nilai fungsi keputusan positif, data dimasukkan ke salah satu kelas. Sebaliknya, jika nilainya negatif, data ditempatkan pada kelas lainnya. Pada

klasifikasi biner, dua kelas ini umumnya direpresentasikan dengan label $+1$ dan -1.

Tujuan utama SVM adalah menemukan *hyperplane* dengan margin maksimum. Margin merupakan jarak antara *hyperplane* dan data terdekat dari masing-masing kelas yang disebut *support vector*. Besarnya margin dapat dirumuskan sebagai berikut:

$$Margin = \frac{2}{\|\mathbf{w}\|} \tag{2.10}$$

Semakin besar margin yang diperoleh, semakin baik kemampuan model dalam memisahkan kelas. Untuk itu, SVM berupaya meminimalkan nilai $\|\mathbf{w}\|$. Jika data tidak sepenuhnya dapat dipisahkan secara linear, digunakan konsep *soft margin* dengan menambahkan penalti terhadap kesalahan klasifikasi. Fungsi optimasinya dapat dituliskan sebagai berikut:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \tag{2.11}$$

dengan batasan:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \tag{2.12}$$

Keterangan:

1. y_i menunjukkan label kelas dari data ke-i.
2. $\mathbf{x}_i$ menunjukkan vektor fitur dari data ke-i.
3. ξ_i menunjukkan variabel kesalahan atau *slack variable*.
4. C menunjukkan parameter penalti yang mengatur keseimbangan antara margin dan kesalahan klasifikasi.
5. n menunjukkan jumlah data yang digunakan dalam proses pelatihan.

Parameter C berperan penting dalam menentukan perilaku model. Nilai C yang besar memberikan penalti lebih tinggi terhadap kesalahan klasifikasi, sehingga model cenderung memisahkan data latih secara lebih ketat. Sebaliknya, nilai yang

lebih kecil memberi toleransi lebih besar sehingga margin yang terbentuk bisa lebih lebar.

Beberapa kelebihan algoritma ini adalah sebagai berikut:

1. Memiliki kemampuan klasifikasi yang baik.
2. Efektif digunakan pada data berdimensi tinggi.
3. Mampu menghasilkan pemisah kelas dengan margin yang optimal.
4. Cukup efektif digunakan pada klasifikasi teks.

Adapun beberapa keterbatasannya adalah sebagai berikut:

1. Membutuhkan proses *parameter tuning* untuk memperoleh performa yang optimal.
2. Waktu *training* dapat menjadi lebih lama pada data dengan ukuran besar.
3. Pemilihan parameter yang kurang tepat dapat memengaruhi performa model.

Pada sejumlah studi klasifikasi teks, SVM kerap menunjukkan performa yang kuat karena mampu membentuk batas keputusan yang baik pada ruang fitur berdimensi tinggi. Dalam penelitian ini, algoritma tersebut digunakan untuk membedakan teks ke dalam kategori *hate speech* dan *non-hate speech* setelah data melalui tahap *preprocessing* dan ekstraksi fitur TF-IDF [40].

2.10 Comparative Study

Comparative study adalah pendekatan penelitian yang digunakan untuk membandingkan dua atau lebih metode, algoritma, atau model berdasarkan kriteria evaluasi tertentu. Dalam konteks *machine learning*, pendekatan ini dipakai untuk melihat metode mana yang memberikan hasil terbaik pada suatu permasalahan. Agar perbandingannya adil, *dataset*, tahapan pemrosesan, dan metrik evaluasi perlu dibuat seragam.

Dalam riset klasifikasi teks, pendekatan ini sering digunakan untuk menilai beberapa algoritma sekaligus, misalnya *Naive Bayes*, SVM, *Decision Tree*, hingga *K-Nearest Neighbor*. Hasilnya menunjukkan bahwa performa sebuah algoritma sangat dipengaruhi oleh karakteristik *dataset*, fitur yang dipakai, dan metode evaluasi yang digunakan [41].

Pada penelitian ini, *comparative study* digunakan untuk membandingkan performa dua model klasifikasi pada tugas deteksi *hate speech* berbahasa Indonesia. Perbandingan dilakukan menggunakan metrik *accuracy*, *precision*, *recall*, dan *F1-score*. Pendekatan ini relevan karena penelitian terdahulu juga banyak memakai skema serupa untuk mengevaluasi klasifikasi teks berbahasa Indonesia [42].

2.11 Experimental Research

Experimental research adalah pendekatan penelitian yang dijalankan melalui serangkaian percobaan untuk melihat pengaruh suatu perlakuan terhadap hasil yang diamati. Dalam *machine learning*, pendekatan ini lazim digunakan untuk menguji performa model melalui tahapan yang sistematis, mulai dari pengumpulan data, *preprocessing*, ekstraksi fitur, pelatihan model, pengujian, hingga evaluasi hasil.

Riset eksperimental pada bidang ini biasanya menetapkan kondisi pengujian yang terkontrol agar hasilnya bisa dibandingkan secara adil. Kondisi tersebut antara lain mencakup penggunaan *dataset* yang sama, strategi pembagian data yang seragam, teknik ekstraksi fitur yang konsisten, serta metrik evaluasi yang tetap. Dengan cara itu, perbedaan hasil dapat lebih masuk akal untuk dikaitkan dengan metode yang diuji, bukan dengan faktor luar [43].

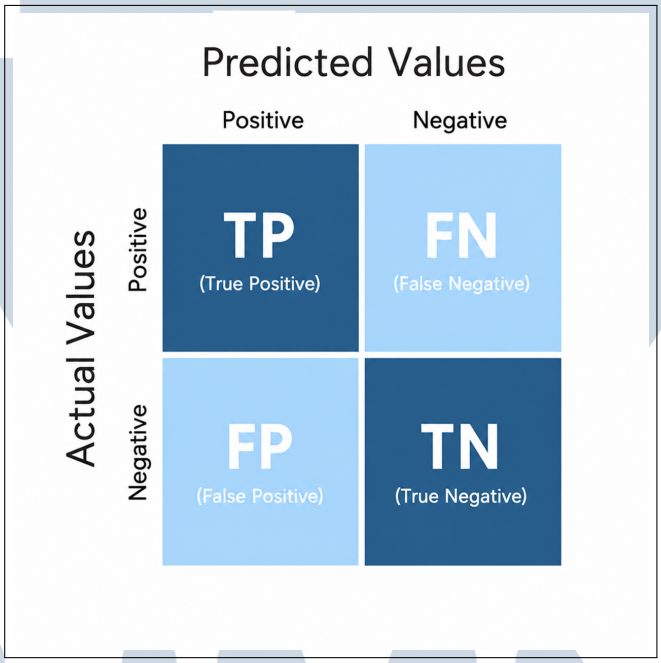
Dalam penelitian ini, pendekatan eksperimen diwujudkan melalui pengujian pada dua algoritma klasifikasi. Seluruh rangkaian dilakukan melalui *text preprocessing*, ekstraksi fitur TF-IDF, pembagian data latih dan data uji, pelatihan model, hingga evaluasi performa. Hasilnya kemudian dipakai untuk melihat metode mana yang lebih baik dalam membedakan teks *hate speech* dan *non-hate speech*.

2.12 Performance Evaluation

Performance evaluation adalah tahap untuk menilai seberapa baik model melakukan klasifikasi pada data uji. Melalui proses ini, kemampuan model dalam mengenali pola dan menghasilkan prediksi yang sesuai dapat dianalisis secara lebih objektif. Pada penelitian klasifikasi, evaluasi biasanya diawali dengan *confusion matrix*, lalu dilanjutkan dengan perhitungan metrik seperti *accuracy*, *precision*, *recall*, dan *F1-score* [44].

2.12.1 Confusion Matrix

Confusion matrix adalah representasi tabular yang memperlihatkan hubungan antara hasil prediksi model dan label sebenarnya [44]. Matriks ini terdiri atas empat komponen utama, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN), yang menjadi dasar untuk menghitung berbagai metrik evaluasi. Struktur *confusion matrix* ditunjukkan pada Gambar 2.3.



Gambar 2.3. Struktur confusion matrix

Keterangan:

1. *TP* (*True Positive*) menunjukkan jumlah data kelas positif yang diprediksi dengan benar sebagai kelas positif.
2. *TN* (*True Negative*) menunjukkan jumlah data kelas negatif yang diprediksi dengan benar sebagai kelas negatif.
3. *FP* (*False Positive*) menunjukkan jumlah data kelas negatif yang keliru diprediksi sebagai kelas positif.
4. *FN* (*False Negative*) menunjukkan jumlah data kelas positif yang keliru diprediksi sebagai kelas negatif.

2.12.2 Accuracy

Accuracy menunjukkan proporsi prediksi yang benar terhadap seluruh data uji. Walaupun sering dipakai sebagai gambaran umum performa model, metrik ini bisa kurang representatif ketika distribusi kelas pada *dataset* tidak seimbang [44].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.13}$$

2.12.3 Precision

Precision menggambarkan ketepatan model saat menyatakan suatu data sebagai kelas positif. Nilai yang tinggi menunjukkan bahwa sebagian besar prediksi positif memang sesuai dengan kondisi sebenarnya [44].

$$Precision = \frac{TP}{TP + FP} \tag{2.14}$$

2.12.4 Recall

Recall berfokus pada kemampuan model untuk menemukan sebanyak mungkin data yang benar-benar termasuk ke dalam kelas positif. Semakin tinggi nilainya, semakin banyak data positif yang berhasil dikenali [44].

$$Recall = \frac{TP}{TP + FN} \tag{2.15}$$

2.12.5 F1-Score

F1-score digunakan ketika dibutuhkan keseimbangan antara *precision* dan *recall*, karena metrik ini dihitung dari rata-rata harmonik keduanya. Oleh sebab itu, *F1-score* sering dijadikan indikator penting pada klasifikasi teks, terutama saat distribusi kelas tidak sepenuhnya seimbang [45].

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.16}$$

Dalam penelitian ini, proses evaluasi digunakan untuk membandingkan performa kedua algoritma melalui metrik *accuracy*, *precision*, *recall*, dan *F1-score*, sehingga perbedaan kekuatan maupun keterbatasannya dapat dianalisis dengan lebih jelas. Selain itu, efisiensi waktu komputasi juga dibandingkan untuk memberikan gambaran mengenai kecepatan proses pelatihan dan pengujian masing-masing algoritma.