BAB II
KAJIAN LITERATUR

2.1 Hoaks Digital

Hoaks digital merupakan informasi yang tidak sesuai dengan fakta atau kebenarannya tidak dapat dipastikan yang disebarkan melalui media digital dengan tujuan tertentu, seperti menyesatkan, memengaruhi opini publik, atau memperoleh keuntungan tertentu [1]. Dalam konteks perkembangan teknologi informasi, hoaks tidak hanya berbentuk berita palsu, tetapi juga dapat berupa potongan informasi, narasi manipulatif, maupun konten yang disajikan secara parsial sehingga menimbulkan kesalahpahaman di masyarakat [4].

Perkembangan internet dan media sosial telah mempercepat proses produksi dan distribusi informasi, termasuk hoaks digital, Karakteristik media digital yang bersifat cepat, masif, dan interaktif menyebabkan informasi dapat menyebar luas tanpa melalui proses verifikasi yang memadai [6]. Kondisi ini diperparah dengan rendahnya tingkat literasi digital sebagian masyarakat, sehingga hoaks mudah diterima dan disebarluaskan kembali tanpa dilakukan pengecekan kebenaran [6].

Hoaks digital sering kali dirancang dengan memanfaatkan aspek emosional, seperti ketakutan, kemarahan, atau kepanikan, agar lebih mudah menarik perhatian pengguna media digital [1]. Dalam beberapa kasus, hoaks disajikan menyerupai berita faktual dengan penggunaan bahasa persuasif, judul provokatif, dan struktur narasi yang tampak meyakinkan, sehingga sulit dibedakan secara manual dari informasi yang valid [2]. Hal ini menjadikan hoaks sebagai permasalahan serius dalam ekosistem informasi digital.

Dalam konteks masyarakat Indonesia, penyebaran hoaks digital memiliki dampak sosial yang signifikan, antara lain menimbulkan keresahan publik, konflik sosial, serta menurunkan kepercayaan terhadap institusi dan media resmi [4]. Hoaks yang beredar di ruang digital juga berpotensi memengaruhi pengambilan keputusan masyarakat, terutama ketika berkaitan dengan isu kesehatan, politik, dan keamanan [6].

Dari sudut pandang teknologi informasi, hoaks digital dapat dipandang sebagai permasalahan klasifikasi informasi, dimana suatu teks perlu diidentifikasi apakah termasuk kategori hoaks atau non-hoaks berdasarkan karakteristik tertentu [2]. Pendekatan berbasis analisis teks menjadi relevan karena sebagian besar hoaks digital disajikan dalam bentuk teks, baik sebagai artikel berita, unggahan media sosial, maupun pesan singkat [5].

Maka itu, pendekatan berbasis *Natural Language Processing* (NLP) dan *Machine Learning* banyak digunakan untuk menganalisis pola linguistik pada teks hoaks dan non-hoaks secara sistematis [5]. Dengan memanfaatkan teknik *text preprocessing*, ekstraksi fitur, dan algoritma klasifikasi, hoaks digital dapat disajikan objek penelitian untuk mengukur kinerja model klasifikasi, tanpa bertujuan menggantikan peran verifikasi manual secara langsung. Pendekatan ini memungkinkan analisis yang lebih objektif dan terukur terhadap karakteristik hoaks digital dalam konteks penelitian komputasional [2]. Dalam penelitian ini, hoaks tidak diposisikan sebagai permasalahan yang harus diselesaikan secara langsung, melainkan sebagai objek kajian dalam eksperimen klasifikasi teks untuk mengevaluasi performa metode *Machine Learning*.

2.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) merupakan bidang yang mencakup berbagai teknik dan metode untuk mengolah bahasa manusia dalam bentuk teks menggunakan sistem komputer. Dalam penerapannya, kualitas pemrosesan teks memegang peranan penting, karena data yang telah melalui tahap analisis dan pembersihan akan berpengaruh langsung terhadap kinerja algoritma yang digunakan [7].

Secara konseptual, NLP bertujuan menjembatani perbedaan antara bahasa alami yang bersifat tidak terstruktur dengan sistem komputasi yang memerlukan data terstruktur [8]. Bahasa manusia memiliki ambiguitas, variasi makna, serta perbedaan konteks yang tinggi, sehingga diperlukan tahapan pemrosesan khusus agar teks dapat direpresentasikan dalam bentuk numerik yang dapat dipahami oleh mesin [9]. Oleh karena itu, NLP tidak hanya berfokus pada pemrosesan kata secara individual, tetapi juga pada pola linguistik dan hubungan antar kata dalam suatu dokumen teks [10].

Dalam konteks analisis teks digital, NLP umumnya diterapkan melalui serangkaian tahapan yang saling berkaitan, mulai dari pembersihan data hingga proses klasifikasi [9]. Setiap tahapan dalam NLP memiliki peran tersendiri dalam mengurangi *noise* dan meningkatkan kualitas representasi teks, sehingga informasi penting dalam dokumen dapat dipertahankan [10]. Pendekatan bertahap ini memungkinkan sistem untuk mengekstraksi karakteristik linguistik yang relevan dari teks, khususnya pada data teks berbahasa Indonesia yang memiliki kompleksitas morfologi yang tinggi [8].

Tabel 2.1 Tahapan Text Preprocessing dalam NLP

Proses	Hasil
Tokenisasi	Teks dipecahkan menjadi kumpulan token berupa kata-kata individual.
Case Folding	Seluruh kata dalam teks diubah menjadi huruf kecil (*lowercase*).
Ekstraksi Fitur	Teks direpresentasikan dalam bentuk vektor numerik menggunakan TF-IDF *dan Word Embedding.*
Klasifikasi	Vektor yang dihasilkan dari TF-IDF dan *Word Embedding* akan digunakan sebagai masukan bagi algoritma SVM dan *Random Forest.*
Evaluasi	Dihasilkan nilai metrik evaluasi berupa *Accuracy, Precision, Recall,* dan *F1-Score.*

Tabel 2.1 menyajikan ringkasan tahapan utama dalam *text preprocessing* berbasis NLP yang digunakan pada penelitian ini. Tabel ini tidak dimaksudkan untuk menjelaskan detail teknis implementasi, melainkan memberikan gambaran konseptual mengenai hubungan antar tahapan dalam alur pemrosesan teks.

Tahap tokenisasi dan *case folding* berfungsi sebagai bagian awal dalam normalisasi teks, sehingga teks memiliki format yang konsisten sebelum dianalisis lebih lanjut. Selanjutnya, ekstraksi fitur dilakukan untuk merepresentasikan teks dalam bentuk vektor numerik menggunakan pendekatan TF-IDF dan *Word Embedding*, yang menjadi dasar masukan bagi algoritma klasifikasi.

Tahap klasifikasi dan evaluasi ditampilkan dalam tabel sebagai bagian dari alur konseptual NLP secara keseluruhan. Pada penelitian ini, hoaks diperlakukan sebagai objek klasifikasi, sehingga tahapan NLP digunakan untuk memastikan bahwa proses pengukuran dan perbandingan kinerja algoritma *Machine Learning* dilakukan secara konsisten dan objektif.

Dengan demikian, NLP berperan sebagai kerangka konseptual dalam mempersiapkan dan merepresentasikan teks hoaks berbahasa Indonesia sebagai data yang dapat dianalisis secara komputasional [10]. Penjelasan tahapan NLP pada sub-bab ini menjadi dasar konseptual untuk pembahasan lebih lanjut mengenai proses *text preprocessing* yang akan dijelaskan secara lebih rinci pada sub-bab 2.3.

2.3 Text Preprocessing

Text Preprocessing merupakan tahapan awal dalam pemrosesan teks yang bertujuan untuk menyiapkan data teks mentah agar dapat dianalisis secara efektif oleh algoritma *Machine Learning* [1]. Data teks yang diperoleh dari media digital umumnya bersifat tidak terstruktur dan mengandung berbagai elemen yang tidak relevan, seperti variasi penulisan, simbol, dan kata umum, sehingga perlu dilakukan pemrosesan awal untuk meningkatkan kualitas data [8].

Secara konseptual, *text preprocessing* berfungsi untuk mengurangi *noise* dan kompleksitas data teks tanpa mengubah makna informasi yang terkandung di dalamnya [7]. Dengan melakukan normalisasi dan pembersihan teks, representasi data yang dihasilkan menjadi konsisten, sehingga mendukung proses ekstraksi fitur dan klasifikasi teks secara lebih optimal [8].

Dalam konteks klasifikasi teks dan deteksi hoaks, *preprocessing* berperan penting karena perbedaan kecil dalam penulisan kata dapat menghasilkan fitur yang berbeda meskipun memiliki makna yang sama [2]. Tanpa *preprocessing* yang tepat, variasi tersebut dapat meningkatkan dimensi fitur secara tidak perlu dan menurunkan kinerja algoritma klasifikasi [9].

Pada penelitian ini, *text preprocessing* diposisikan sebagai bagian dari kerangka konseptual NLP untuk memastikan bahwa teks hoaks dan non-hoaks diperlakukan secara konsisten sebelum dilakukan ekstraksi fitur ada proses klasifikasi [8]. Penjelasan pada sub-bab ini berfokus pada konsep dan tujuan dari setiap tahapan *preprocessing*, sedangkan detail implementasi teknis dibahas lebih lanjut pada Bab III.

2.3.1 Case Folding

Case Folding merupakan proses normalisasi teks dengan mengubah seluruh karakter huruf menjadi huruf kecil (*lowercase*) untuk mengurangi variasi kata akibat perbedaan penggunaan huruf kapital [7]. Tahap penting karena perbedaan kapitalisasi tidak memberikan kontribusi semantik yang signifikan dalam tugas klasifikasi teks [2].

Dalam pemrosesan teks, kata dengan bentuk huruf besar dan kecil yang berbeda dapat dianggap sebagai token yang berbeda oleh sistem, sehingga meningkatkan jumlah fitur yang tidak diperlukan [8]. Dengan menerapkan *case folding*, teks menjadi lebih seragam dan kompleksitas data dapat dikurangi [9].

Pada teks hoaks digital, penggunaan huruf kapital sering kali tidak konsisten, baik pada judul maupun isis teks [2]. Oleh karena itu, *case folding* membantu memastikan bahwa

pola linguistik dianalisis berdasarkan makna kata, bukan berdasarkan variasi format penulisan [7].

Dalam penelitian ini, *case folding* digunakan sebagai tahap awal *preprocesssing* untuk memastikan konsistensi teks sebelum dilakukan tahapan tokenisasi dengan penghapusan kata-kata yang tidak relevan. Tahapan ini mendukung proses analisis teks yang lebih stabil dan objektif pada tahap selanjutnya [8].

2.3.2 Tokenisasi

Tokenisasi merupakan proses pemecahan teks menjadi unit-unit yang lebih kecil yang disebut token, umumnya berupa kata [7]. Tahapan ini memungkinkan sistem untuk menganalisis teks pada level kata sehingga pola linguistik dalam dokumen dapat diidentifikasi [1].

Tokenisasi berperan sebagai tahap dasar dalam analisis teks karena hampir seluruh metode ekstraksi fitur dan klasifikasi bekerja pada unit token [8]. Tanpa tokenisasi, teks hanya diperlakukan sebagai rangkaian karakter yang sulit dianalisis secara komputasional [9].

Dalam deteksi hoaks, tokenisasi membantu memisahkan kata-kata yang membentuk narasi berita sehingga karakteristik tertentu pada teks hoaks dan non-hoaks dapat dianalisis secara lebih terstruktur [2]. Token yang dihasilkan kemudian digunakan sebagai dasar dalam proses ekstraksi fitur [8].

Pada penelitian ini, tokenisasi digunakan sebagai bagian dari *preprocessing* untuk mengubah teks berita menjadi kumpulan token yang siap dianalisis lebih lebih lanjut. Tahapan ini memastikan bahwa representasi teks yang digunakan pada proses berikutnya berasal dari unit linguistik yang konsisten [7].

2.4 Ekstraksi Fitur

Ekstraksi fitur merupakan tahapan dalam pemrosesan teks yang bertujuan mengubah data teks hasil *preprocessing* menjadi representasi numerik yang dapat diproses oleh algoritma *Machine Learning* [8]. Karena algoritma klasifikasi tidak dapat bekerja langsung data teks mentah, diperlukan metode representasi yang mampu menangkap informasi penting dari teks dalam bentuk vektor numerik [1].

Ekstraksi fitur berperan sebagai penghubung antara data teks dan proses klasifikasi. Kualitas fitur yang dihasilkan sangat memengaruhi kemampuan algoritma dalam membedakan kelas data, termasuk pada klasifikasi teks hoaks dan non-hoaks [9]. Oleh

karena itu, pemilihan metode ekstraksi fitur menjadi faktor penting dalam penelitian klasifikasi teks [8].

Dalam konteks analisis teks digital, metode ekstraksi fitur dirancang untuk merepresentasikan karakteristik linguistik tertentu, seperti frekuensi kata atau hubungan semantik antar kata [1]. Setiap metode memiliki kelebihan dan keterbatasan dalam menangkap informasi dari teks, sehingga perlu dilakukan pemilihan metode yang sesuai dengan tujuan penelitian [7]. Pemilihan metode ekstraksi fitur dalam penelitian ini tidak bertujuan untuk menentukan metode terbaik secara mutlak, melainkan untuk menganalisis perbedaan karakteristik representasi fitur dalam konteks eksperimen klasifikasi teks.

Pada penelitian ini, ekstraksi fitur digunakan untuk memastikan bahwa teks hoaks dan non-hoaks direpresentasikan secara konsisten sebelum dilakukan proses klasifikasi. Dua metode ekstraksi fitur yang digunakan, yaitu TF-IDF dan *word embedding*, dipilih untuk memungkinkan analisis komparatif terhadap pengaruh representasi fitur terhadap kinerja algoritma *Machine Learning* [8].

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

Pada metode TF-IDF, bobot suatu kata merepresentasikan tingkat relevansinya dalam sebuah dokumen Secara konseptual, bobot *Term Frequency-Inverse Document Frequency* (TF-IDF) dibentuk dari dua komponen yang saling melengkapi. *Term Frequency* (TF) mengukur seberapa sering suatu kata muncul di dalam sebuah dokumen tertentu, sehingga kata yang lebih sering muncul pada dokumen tersebut dianggap lebih merepresentasikan isi dokumen itu. Sementara itu, *Inverse Document Frequency* (IDF) mengukur seberapa jarang suatu kata muncul di seluruh dokumen dalam korpus; kata yang muncul hampir di semua dokumen, seperti kata umum "dan" atau "yang", akan diberi bobot rendah, sedangkan kata yang hanya muncul pada sebagian kecil dokumen akan diberi bobot tinggi karena dianggap lebih spesifik dan informatif. Bobot akhir TF-IDF diperoleh dari hasil perkalian kedua komponen tersebut, sehingga suatu kata hanya akan memperoleh bobot tinggi apabila kata itu sering muncul pada suatu dokumen tertentu namun jarang muncul pada dokumen-dokumen lain dalam korpus.

Prinsip kerja *Term Frequency-Inverse Document Frequency* (TF-IDF) didasarkan pada asumsi bahwa kata yang sering muncul dalam suatu dokumen tetapi jarang muncul di dokumen lain memiliki nilai diskriminatif yang lebih tinggi [8]. Dengan demikian, *Term Frequency-Inverse Document Frequency* (TF-IDF) menghasilkan representasi teks dalam

bentuk vektor berdimensi tinggi yang mencerminkan distribusi kata dalam dokumen [1].

Dalam deteksi hoaks, *Term Frequency-Inverse Document Frequency* (TF-IDF) digunakan untuk merepresentasikan pola kemunculan kata yang khas pada teks hoaks maupun non-hoaks [2]. Kata atau istilah tertentu yang sering muncul dalam berita hoaks dapat memiliki bobot yang berbeda dibandingkan dengan berita valid, sehingga membantu klasifikasi berbasis teks [9].

Pada penelitian ini, *Term Frequency-Inverse Document Frequency* (TF-IDF) digunakan sebagai salah satu metode ekstraksi fitur untuk mengevaluasi kinerja algoritma klasifikasi. Penggunaan *Term Frequency-Inverse Document Frequency* (TF-IDF) memungkinkan analisis komparatif terhadap efektivitas representasi berbasis frekuensi kata ketika dikombinasikan dengan algoritma *Machine Learning* yang berbeda [8].

2.4.2 Word Embedding FastText

Word Embedding merupakan metode ekstraksi fitur yang merepresentasikan kata dalam bentuk vektor numerik berdimensi tetap dengan mempertimbangkan konteks kemunculan kata dalam teks, sehingga berbeda dari *Term Frequency-Inverse Document Frequency* (TF-IDF) yang bekerja berbasis frekuensi kemunculan kata [1], [7]. Melalui pendekatan ini, kata-kata yang memiliki makna serupa dipetakan ke dalam ruang vektor yang saling berdekatan, sehingga sistem dapat menangkap kesamaan makna antar kata meskipun memiliki bentuk kata yang berbeda [7]. Dalam konteks teks hoaks, karakteristik ini membantu merepresentasikan makna kata dan konteks narasi secara lebih komprehensif, sehingga teks hoaks dan non-hoaks dapat dianalisis berdasarkan kemiripan makna antar kata [2].

Pada penelitian ini, *Word Embedding FastText* digunakan sebagai metode ekstraksi fitur alternatif dari *Term Frequency-Inverse Document Frequency* (TF-IDF). Penggunaan *Word Embedding FastText* memungkinkan analisis pengaruh representasi semantik terhadap kinerja algoritma klasifikasi, khususnya dalam perbandingan antara *Support Vector Machine* (SVM) dan *Random Forest* [9].

Berbeda dengan metode *Word Embedding* lain yang merepresentasikan kata secara utuh, FastText tidak hanya merepresentasikan kata secara utuh, tetapi juga memecah setiap kata menjadi unit-unit yang lebih kecil berupa *n-gram* karakter atau sub-kata [18]. Sebagai contoh, kata "menghibur" akan dipecah menjadi beberapa *n-gram* seperti "me", "meng", "hibur", "ibur", dan "r", kemudian representasi vektor akhir dari kata tersebut diperoleh dari

gabungan seluruh vektor *n-gram* penyusunnya [18]. Pendekatan berbasis sub-kata ini memberikan keunggulan bagi *FastText* dalam menangani kata yang tidak terdapat dalam kosakata pelatihan (*out-of-vocabulary*), karena kata baru tetap dapat direpresentasikan melalui kombinasi *n-gram* penyusunnya meskipun kata tersebut belum pernah muncul secara utuh dalam data pelatihan [18]. Karakteristik ini menjadikan *FastText* lebih sesuai digunakan pada teks berbahasa Indonesia yang memiliki variasi morfologi tinggi, seperti pengaruh imbuhan dan penggabungan kata, dibandingkan metode *Word Embedding* yang hanya merepresentasikan kata secara utuh.

Namun, *Word Embedding FastText* pada dasarnya merepresentasikan kata dalam bentuk vektor, sedangkan algoritma klasifikasi memerlukan representasi pada tingkat dokumen. Oleh karena itu, diperlukan metode untuk menggabungkan vektor-vektor kata menjadi satu representasi dokumen.

Salah satu metode yang umum digunakan adalah *mean pooling*, yaitu dengan menghitung rata-rata seluruh vektor kata dalam suatu dokumen. Secara matematis, proses ini dinyatakan sebagai:

$$v_{doc} = \frac{1}{n} \sum_{i=1}^{n} v_i \quad \text{.....(2.1)}$$

Dimana:

v_i merupakan vektor dari setiap kata dan n adalah jumlah kata dalam dokumen. Proses ini menghasilkan representasi dokumen dalam bentuk vektor berdimensi tetap, sehingga dapat digunakan sebagai input bagi algoritma klasifikasi.

2.5 Klasifikasi

Klasifikasi teks merupakan proses penentuan kelas suatu dokumen teks ke dalam kelas tertetntu ebrdasarkan pola yang terdapat pada representasi fitur teks [1]. Dalam penelitian berbasis teks, dokumen tidak diproses dalam bentuk kalimat mentah, melainkan direpresentasikan sebagai vektor numerik hasil ekstraksi fitur sehingga dapat dianalisis oleh algoritma *Machine Learning* [8].

Pendekatan klasifikasi teks umumnya menggunakan pembelajaran terawasi, dimana model dibangun berdasarkan data latih yang telah diberi label kelas [7]. Melalui proses pelatihan tersebut, algoritma mempelajari karakteristik yang membedakan masing-masing kelas sehingga mampu memberikan prediksi terhadap data yang belum pernah dilihat sebelumnya [1].

Pada penelitian deteksi hoaks, klasifikasi teks digunakan untuk membedakan berita

hoaks dan non-hoaks berdasarkan karakteristik linguistik yang terkandung dalam teks [2]. Dengan pendekatan ini, hoaks diperlakukan sebagai kategori data yang dapat diukur kinerja klasifikasinya, sehingga memungkinkan dilakukan evaluasi dan perbandingan performa algoritma secara objektif [9].

Berdasarkan kebutuhan tersebut, penelitian ini menggunakan dua algoritma klasifikasi yang memiliki karakteristik berbeda, yaitu *Support Vector Machine* (SVM) dan *Random Forest*. Pemilihan kedua algoritma ini bertujuan untuk membandingkan kinerja model berbasis margin dan. model berbasi *ensemble* dalam mengklasifikasi teks hoaks berbahasa Indonesia.

2.5.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan metode klasifikasi yang bekerja dengan menentukan batas keputusan optimal untuk memisahkan data ke dalam beberapa kelas dalam bentuk *hyperplane* [3]. *Hyperplane* tersebut ditentukan dengan memaksimalkan jarak antara data dari kelas yang berbeda sehingga model memiliki kemampuan generalisasi yang lebih baik [9].

Dalam proses pembentukan model, SVM hanya memanfaatkan sebagian data yang berada paling dekat dengan batas pemisah, yang dikenal sebagai *support vector* [7]. Data-data inilah yang berperan utama dalam menentukan keputusan klasifikasi, sedangkan data lain yang berada jauh dari batas memiliki pengaruh yang lebih kecil [8].

SVM banyak digunakan dalam klasifikasi teks karena kemampuannya dalam menangani data berdimensi tinggi seperti representasi fitur berbasis TF-IDF [9]. Beberapa penelitian menunjukkan bahwa SVM mampu menghasilkan performa yang stabil dalam tugas klasifikasi teks, termasuk pada kasus deteksi berita hoaks [2].

Dalam penelitian ini, SVM digunakan untuk mengukur kemampuan algoritma berbasis margin dalam membedakan teks hoaks dan non-hoaks berdasarkan fitur yang dihasilkan. Hasil klasifikasi SVM kemudian digunakan sebagai salah satu acuan dalam analisis komparatif performa algoritma [9].

Secara matematis, SVM menggunakan fungsi keputusan untuk menentukan kelas suatu data, yang dinyatakan sebagai:

$$f(x) = w \cdot x + b \quad (2.2)$$

Dimana: w merupakan vektor bobot, x merupakan vektor fitur hasil representasi teks, dan b merupakan bias. Nilai dari fungsi keputusan ini digunakan untuk menentukan posisi suatu

data terhadap *hyperplane* pemisah.

Dalam proses pelatihan, SVM menggunakan fungsi *hinge loss* untuk mengukur kesalahan klasifikasi, yang dirumuskan sebagai:

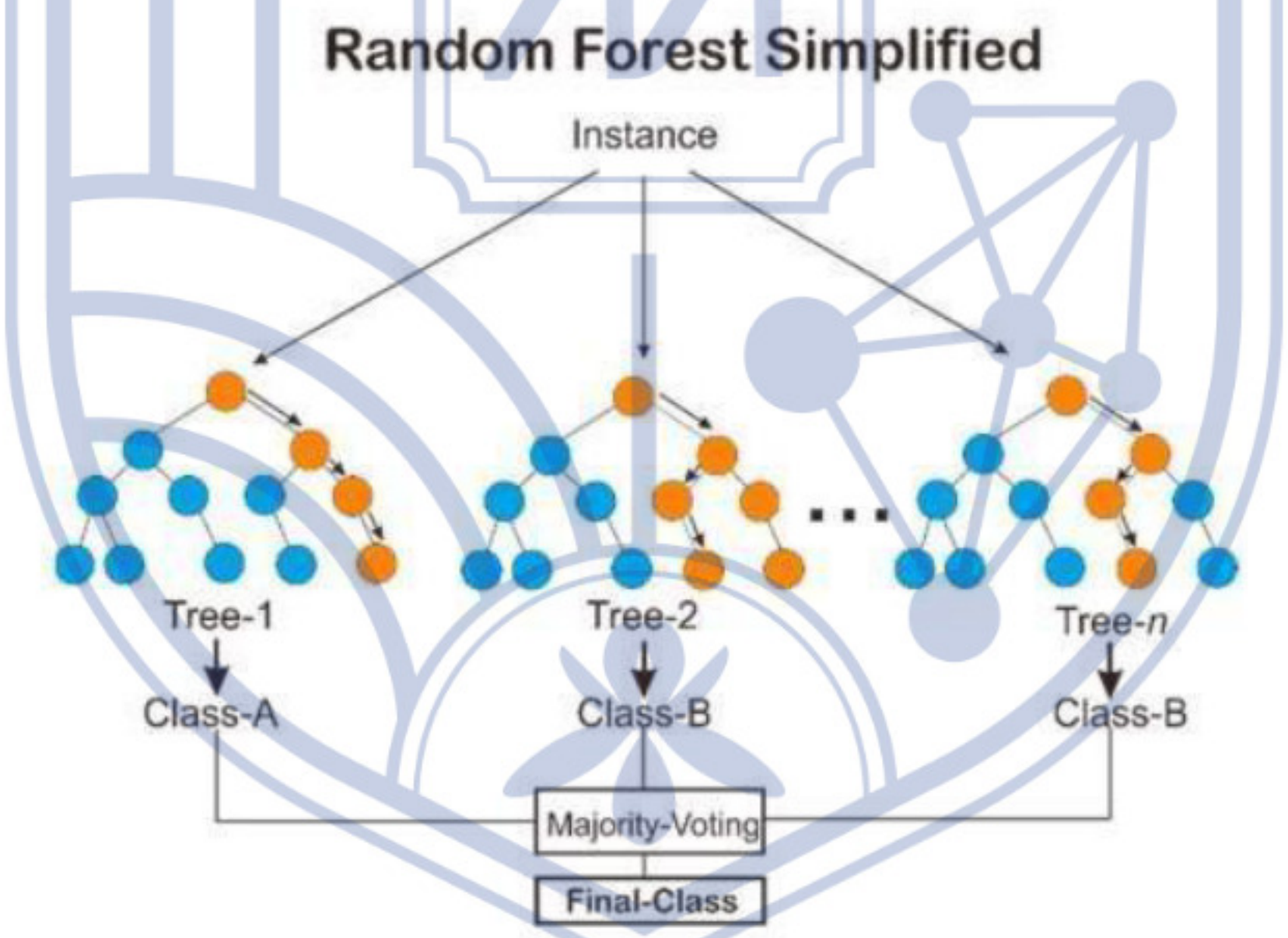
$$L = max(0,1 - y(w \cdot x + b)) \quad (2.3)$$

di mana y merupakan label kelas dengan nilai 1 untuk kelas hoaks dan 0 untuk kelas non-hoaks.

Fungsi hinge *loss* akan bernilai nol apabila data telah diklasifikasikan dengan benar dan berada di luar margin, sedangkan akan bernilai positif apabila terjadi kesalahan klasifikasi atau margin yang terbentuk belum optimal. Oleh karena itu, tujuan pelatihan SVM adalah meminimalkan nilai *loss* untuk memperoleh *hyperplane* yang mampu memisahkan data dengan margin yang maksimal.

Tujuan pelatihan SVM adalah meminimalkan nilai *loss,* sehingga model dapat memperbesar jarak (*margin*) antar kelas dan mengurangi kesalahan klasifikasi.

2.5.2 Random Forest



Gambar 2. 1 Mekanisme Klasifikasi Algoritma Random Forest [11]

Random Forest merupakan algoritma klasifikasi berbasis *ensemble learning* yang terdiri dari kumpulan pohon keputusan yang dibangun secara independen [7]. Setiap pohon keputusan dilatih menggunakan subset data dan subset fitur yang berbeda, sehingga menghasilkan variasi model dalam satu sistem klasifikasi [9]

$$RFfi_i = \frac{\Sigma j \epsilon alltrees normfi_{ij}}{T} \quad (2.4)$$

Dimana:

RFfi(i) merupakan tingkat kepentingan fitur ke-i yang diperoleh dari seluruh *decision tree* dalam model *Random Forest.*

Normfi(subij) merupakan nilai kepentingan fitur yang telah dinormalisasi untuk fitur ke-i pada pohon keputusan ke-j.

T merupakan jumlah total *decision tree* yang digunakan dalam model *Random Forest.*

Berdasarkan persamaan tersebut, tingkat kepentingan suatu fitur diperoleh dari akumulasi nilai kepentingan fitur pada seluruh pohon keputusan yang kemudian digunakan sebagai dasar dalam menentukan kontribusi masing-masing fitur terhadap proses klasifikasi [12].

Pada tahap prediksi, setiap pohon keputusan dalam *Random Forest* akan menghasilkan prediksi kelas terhadap data masukan. Hasil akhir klasifikasi ditentukan melalui mekanisme *majority voting*, yaitu kelas dengan jumlah suara terbanyak dari seluruh pohon keputusan akan dipilih sebagai keputusan akhir [9].

Pendekatan *ensemble* pada *Random Forest* bertujuan untuk mengurangi kesalahan klasifikasi yang dapat muncul pada satu pohon keputusan [7]. Dengan menggabungkan hasil dari banyak pohon, *Random Forest* mampu meningkatkan stabilitas dan ketahanan model terhadap variasi data [9].

Dalam klasifikasi teks, *Random Forest* dapat digunakan untuk menangani berbagai jenis representasi fitur, termasuk fitur berbasis frekuensi maupun fitur berbasis *embedding* [2]. Pada penelitian ini, *Random Forest* digunakan sebagai algoritma pembanding terhadap SVM untuk mengevaluasi perbedaan performa klasifikasi teks hoaks dan non-hoaks [9].

2.6 Pendekatan Kuantitatif Eksperimental

Pendekatan kuantitatif merupakan pendekatan penelitian ilmiah yang disusun secara sistematis dengan menekankan proses pengukuran dan penggunaan data berbentuk angka, statistik, maupun persentase dalam menjawab permasalahan penelitian [13]. Sementara itu, penelitian eksperimental merupakan penelitian yang dilakukan untuk menguji hasil dari suatu perlakuan tertentu dalam kondisi yang terkendali [14]. Berdasarkan kedua konsep tersebut, pendekatan kuantitatif eksperimental dapat dipahami sebagai pendekatan penelitian yang menggunakan proses eksperimen dan pengukuran numerik untuk menguji atau membandingkan suatu metode secara objektif.

Dalam penelitian berbasis *Machine Learning*, pendekatan kuantitatif eksperimental digunakan untuk melatih, menguji, dan mengevaluasi model menggunakan data serta metrik

evaluasi tertentu. Pendekatan ini memungkinkan performa model dianalisis secara terukur berdasarkan hasil pengujian yang diperoleh [7], [9], [15]. Pada penelitian klasifikasi teks, pendekatan eksperimental dilakukan dengan memberikan perlakuan yang sama terhadap dataset dan skema pengujian, kemudian membandingkan hasil performa dari algoritma yang digunakan [3], [9], [15]. Dengan demikian, perbandingan antar model dapat dilakukan secara lebih konsisten karena seluruh model diuji menggunakan data dan prosedur pengujian yang sama.

Dalam penelitian ini, pendekatan kuantitatif eksperimental digunakan karena penelitian berfokus pada perbandingan kinerja dua model klasifikasi, yaitu *Support Vector Machine* berbasis *Term Frequency-Inverse Document Frequency* dan *Random Forest* berbasis Word Embedding FastText. Kedua model diuji menggunakan dataset yang sama, skema *Stratified 5-Fold Cross Validation* yang konsisten, serta metrik evaluasi berupa *Accuracy, Precision, Recall,* dan *F1-score* [16], [17]. Hasil evaluasi tersebut digunakan sebagai dasar untuk menentukan model yang memiliki kinerja paling optimal dalam mendeteksi hoaks berbahasa Indonesia.

2.7 Evaluasi Model

Evaluasi model merupakan tahapan yang digunakan untuk menilai kemampuan algoritma klasifikasi dalam memprediksi kelas data secara benar berdasarkan hasil klasifikasi yang dihasilkan [1]. Dalam penelitian klasifikasi teks, evaluasi diperlukan untuk memastikan bahwa model tidak hanya mampu menghasilkan prediksi, tetapi juga memiliki tingkat ketepatan yang dapat dipertanggungjawabkan secara ilmiah [9].

Pada kasus deteksi hoaks, evaluasi model menjadi penting karena kesalahan klasifikasi dapat berdampak pada interpretasi performa algoritma yang digunakan [2]. Oleh karena itu, evaluasi tidak cukup dilakukan hanya dengan satu metrik, melainkan memerlukan beberapa ukuran performa untuk menggambarkan kemampuan model secara menyeluruh [9].

Evaluasi model klasifikasi teks umumnya dilakukan dengan membandingkan hasil prediksi model terhadap label data sebenarnya [7]. Perbandingan tersebut disajikan dalam bentuk *confusion matrix* yang menunjukkan distribusi prediksi benar dan salah pada setiap kelas [8].

Dalam penelitian ini, evaluasi model digunakan untuk membandingkan performa algoritma *Support Vector Machine* dan *Random Forest* dalam mengklasifikasikan teks hoaks dan non-hoaks. Evaluasi dilakukan untuk menilai perbedaan kemampuan algoritma dalam

memanfaatkan representasi fitur yang digunakan tanpa bertujuan membangun sistem deteksi hoaks secara aplikatif [9].

Untuk melakukan evaluasi tersebut, hasil prediksi model dibandingkan dengan label data sebenarnya sehingga dapat diketahui jumlah prediksi yang benar dan salah pada setiap kelas. Perbandingan ini secara umum direpresentasikan dalam bentuk *confusion matrix*, yang menjadi dasar dalam perhitungan berbagai metrik evaluasi kinerja model [7].

2.7.1 Confusion Matrix

Confusion matrix merupakan tabel evaluasi yang digunakan untuk menggambarkan hasil klasifikasi dengan membandingkan kelas prediksi dan kelas aktual. Tabel ini menunjukkan jumlah prediksi yang benar dan salah pada masing-masing kelas, sehingga memudahkan analisis kesalahan klasifikasi yang dilakukan oleh model [2].

Dalam klasifikasi teks hoaks, *confusion matrix* membantu mengidentifikasi jenis kesalahan yang dominan, seperti hoaks yang tidak terdeteksi (*false negative*) atau berita non-hoaks yang salah diklasifikasikan sebagai hoak (*false positive*) [2]. Analisis terhadap jenis kesalahan ini penting untuk memahami keterbatasan model klasifikasi yang digunakan [7].

Selain menampilkan distribusi hasil klarifikasi, *confusion matrix* memiliki peran penting sebagai dasar dalam perhitungan metrik valuasi kinerja model. Seluruh metrik evaluasi, seperti *accuracy, precision, recall dan F1-score*, diturunkan dari kombinasi nilai *True Negative, False Positive*, dan *False Negative* yang terdapat pada *confusion matrix* [7]. Oleh karena itu, pemahaman terhadap struktur dan makna setiap komponen dalam *confusion matrix* menjadi langkah awal yang penting sebelum melakukan evaluasi performa model secara kuantitatif [7]

Dalam konteks deteksi hoaks, setiap jenis kesalahan klasifikasi memiliki implikasi yang berbeda. Kesalahan *False Positivie* menunjukkan kondisi ketika informasi yang sebenarnya valid diklasifikasikan sebagai hoaks, yang berpotensi menurunkan kepercayaan terhadap sistem klasifikasi [2]. Sebaliknya, kesalahan *False Negative* menunjukkan kondisi ketika berita hoaks gagal terdeteksi dan diklasifikasikan sebagai non-hoaks, sehingga berpotensi memperbesar risiko penyebaran informasi palsu [2]. Oleh karena itu, analisis distribusi kesalahan melalui *confusion matrix* menjadi penting untuk menilai karakteristik dan risiko dari model klasifikasi yang digunakan [1]. Tabel 2.2 menunjukkan bentuk umum *confusion matrix* yang digunakan pada klasifikasi teks hoaks dan non-hoaks.

Tabel 2.2 Confusion Matrix Klasifikasi Teks Hoaks dan Non-Hoaks

Prediksi / Kelas Aktual	Hoaks	Non-Hoaks
Hoaks	TP	FN
Non-Hoaks	FP	TN

Keterangan:

1. True Positive (TP) adalah jumlah data hoaks yang diprediksi sebagai hoaks [18].
2. True Negative (TN) adalah data non-hoaks yang diprediksi sebagai non-hoaks [18].
3. False Positive (FP) adalah jumlah data non-hoaks yang salah diprediksi sebagai hoaks [8].
4. False Negative (FN) adalah jumlah data hoaks yang salah diprediksi sebagai non-hoaks [8].

Confusion matrix menjadi landasan utama dalam perhitungan metrik evaluasi karena seluruh nilai evaluasi diperoleh dari kombinasi TP, TN, FP, dan FN [18]. Nilai-nilai yang diperoleh dari *confusion matrix* tersebut selanjutnya digunakan sebagai dasar dalam perhitungan metrik evaluasi kinerja model, yang akan dibahas pada sub-bab berikutnya [7].

2.7.2 Metrik Evaluasi Klasifikasi

Metrik evaluasi klasifikasi digunakan untuk menilai kinerja model *Machine Learning* berdasarkan perbandingan antara hasil prediksi model dan label data yang sebenarnya [1]. Dalam penelitian klasifikasi teks, metrik evaluasi tidak hanya berfungsi untuk menunjukkan tingkat keberhasilan model, tetapi juga untuk menggambarkan jenis dan karakteristik kesalahan klasifikasi yang terjadi. [8].

Pada klasifikasi teks hoaks, proses evaluasi model menjadi lebih kompleks karena karakteristik data teks yang beragam serta potensi ketidakseimbangan distribusi kelas antara hoaks dan non-hoaks [2]. Kondisi tersebut menyebabkan penggunaan satu metrik evaluasi saja tidak cukup untuk merepresentasikan performa model secara menyeluruh [1]. Oleh karena itu, diperlukan beberapa metrik evaluasi yang saling melengkapi untuk menangkap berbagai aspek kinerja model klasifikasi [5].

Seluruh metrik evaluasi klasifikasi diturunkan dari *confusion matrix*, yang merepresentasikan distribusi prediksi benar dan salah pada setiap kelas [7]. Nilai *True*

Positive, True Negative, False Positive, dan *False Negative* yang terdapat pada *confusion matrix* menjadi dasar dalam perhitungan bebagai metrik evaluasi kinerja model [7]. Dengan memahami hubungan antara *confusion matrix* dan metrik evaluasi, analisis performa model dapat dilakukan secara sistematis dan terstruktur [8].

Berdasarkan pendekatan tersebut, beberapa metrik evaluasi yang umum digunakan dalam penelitian klasifikasi teks adalah *accuracy, precision, recall,* dan *F1-score* [1]. Keempat metrik ini dipilih karena masing-masing merepresentasikan sudut pandang yang berbeda dalam menilai performa model, khususnya pada data dengan distribusi kelas yang tidak seimbang seperti pada deteksi hoaks [2].

1. Accuracy

Accuracy digunakan untuk mengukur proporsi keseluruhan prediksi yang dilakukan dengan benar oleh model klasifikasi [5]. Metrik ini sering digunakan sebagai indikator awal performa model karena memberikan gambaran umum mengenai tingkat ketepatan klasifikasi secara keseluruhan [1].

Meskipun *accuracy* sering digunakan sebagai indikator awal performa model, metrik ini memiliki keterbatasan ketika diterapkan pada data dengan distribusi kelas yang tidak seimbang [5]. Dalam konteks deteksi hoaks, nilai *accuracy* yang tinggi belum tentu menunjukkan bahwa model mampu mendeteksi berita hoaks dengan baik, karena model dapat memperoleh nilai *accuracy* tinggi hanya dengan memprediksi kelas mayoritas [2].

Oleh karena itu, *accuracy* diposisikan ebagai metrik pendukung yang memberikan gambaran global performa model, namun perlu dianalisis bersama metrik evaluasi lainnya agar interpretasi kinerja model tidak bersifat menyesatkan [5].

$$Accuracy = \frac{correct\ classification}{total\ classifications} = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.5)$$

Dimana:

TP (*True Positive*), jumlah data hoaks yang benar diklasifikasikan sebagai hoaks.

TN (*True Negative*), jumlah data non-hoaks yang benar diklasifikasikan sebagai non-hoaks.

FP (*False Positive*), jumlah data non-hoaks yang salah diklasifikasikan sebagai hoaks.

FN (*False Negative*), jumlah data hoaks yang salah diklasifikasikan sebagai non-hoaks.

Untuk memudahkan interpretasi nilai *accuracy*, digunakan pengelompokan rentang nilai secara numerik sebagai acuan konseptual.

Tabel 2.3 Interpretasi Nilai Accuracy

Nilai *Accuracy*	Interpretasi
≥ 0,80 (≥ 80%)	Model memiliki kemampuan klasifikasi yang baik.
0,60-0,79 (60%-79%)	Model memiliki kemampuan klasifikasi sedang.
< 0,60 (< 60%)	Model memiliki kemampuan klasifikasi rendah.

Rentang nilai pada tabel ini digunakan sebagai acuan konseptual untuk membantu interpretasi nilai *accuracy* dan tidak merepresentasikan hasil evaluasi pada penelitian ini.

2. Precision

Precision merupakan metrik evaluasi yang digunakan untuk mengukur tingkat ketepatan model dalam memberikan prediksi terhadap suatu kelas tertentu [5]. Metrik ini menunjukkan seberapa besar proporsi data yang diprediksi sebagai suatu kelas ada dalam kelas tersebut [8].

Dalam konteks deteksi hoaks, *precision* berkaitan langsung dengan tingkat kesalahan *false positive*, yaitu ketika berita non-hoaks salah diklasifikasikan sebagai hoaks [6]. Nilai *precision* yang rendah menunjukkan tingginya risiko kesalahan tersebut, yang dapat berdampak pada penurunan kepercayaan terhadap hasil klasifikasi [3].

Dengan demikian, *precision* digunakan untuk menilai tingkat kehati-hatian model dalam memberikan label hoaks dan menjadi penting ketika kesalahan pelabelan informasi valid sebagai hoaks perlu diminimalkan [2].

Rumus *precision*:

$$Precision = \frac{correct\ classified\ actual\ positives}{everything\ classified\ as\ positive} = \frac{TP}{TP+FP} \quad (2.6)$$

Dimana:

TP (*True Positive*), jumlah data hoaks yang benar diprediksi sebagai hoaks.

FP (*False Positive*), jumlah data non-hoaks yang salah diprediksi sebagai hoaks.

Precision menekankan ketepatan keputusan model saat memprediksi kelas positif, maka itu:

a. *Precision* tinggi menandakan kesalahan dalam menandai non-hoaks sebagai hoaks.

b. *Precision* rendah menandakan banyak konten non-hoaks salah ditandai sebagai

hoaks.

Persamaan tersebut menunjukkan bahwa *precision* dihitung berdasarkan perbandingan antara jumlah prediksi hoaks yang benar dengan seluruh prediksi hoaks yang dihasilkan oleh model [5].

3. Recall

Recall merupakan metrik evaluasi yang digunakan untuk mengukur kemampuan model dalam mengidentifikasi seluruh data positif yang sebenarnya, yaitu dengan membandingkan jumlah *true positive* terhadap total data positif [5]. Metrik ini menunjukkan seberapa besar proporsi data aktual dari suatu kelas yang berhasil oleh model klasifikasi [2].

Dalam konteks deteksi hoaks, *recall* memiliki peran yang sangat penting karena berkaitan langsung dengan kesalahan *false negative*, yaitu kondisi ketika berita hoaks gagal terdeteksi oleh model [6]. Nilai *recall* yang rendah menunjukkan bahwa masih terdapat banyak berita hoaks yang tidak berhasil diklasifikasikan, sehingga dapat berpotensi menimbulkan dampak negatif bagi pengguna informasi [3].

Oleh karena itu, *recall* digunakan untuk mengevaluasi sensitivitas model terhadap kelas hoaks dan kemampuannya dalam menemukan seluruh data hoaks yang relevan [6]. Berikut merupakan rumus *recall*:

$$Recall\ (or\ TPR) = \frac{correctly\ classified\ actual\ positives}{all\ actual\ positives} = \frac{TP}{TP+FN} \quad (2.7)$$

Dimana:

TP (*True Positive*), jumlah data hoaks yang berhasil diklasifikasikan dengan benar sebagai hoaks.

FN (*False Negative*), jumlah data hoaks yang salah diklasifikasikan sebagai non-hoaks.

Recall berfokus pada kelengkapan deteksi kelas positif, berikut makna rumus:

a. *Recall* tinggi merupakan sebagian besar hoaks berhasil terdeteksi.

b. *Recall* rendah artinya banyak hoaks yang lolos sebagai non-hoaks.

$$FPR = \frac{incorrectly\ classified\ actual\ negatives}{all\ actual\ negatives} = \frac{FP}{FP+TN} \quad (2.8)$$

Dimana:

FP (*False Positive*), jumlah data non-hoaks yang salah diklasifikasikan sebagai hoaks.

TN (*True Negative*), jumlah data non-hoaks yang berhasil diklasifikasikan dengan benar sebagi non-hoaks.

Berdasarkan persamaan tersebut, *recall* dihitung dengan membandingkan jumlah data hoaks yang berhasil diprediksi dengan benar terhadap seluruh data hoaks yang ada [5].

4. F1-Score

Precision dan *recall* sering kali menunjukkan hubungan yang saling bertolak belakang, sehingga diperlukan metrik evaluasi yang mampu merepresentasikan keseimbangan antara keduanya [5]. *F-1 Score* digunakan untuk menggabungkan nilai *precision* dan *recall* dalam satu ukuran performa yang lebih komprehensif [2].

Pada klasifikasi teks hoaks, *F1-score* sering dijadikan metrik utama karena mampu memberikan gambaran performa model yang lebih adil ketika distribusi kelas tidak seimbang [6]. Penggunaan *F1-score* membantu menghindari bias evaluasi yang dapat muncul apabila penilaian performa hanya didasarkan pada *accuracy* [3].

$$F1 = 2 \times \frac{Precision \times Recall}{Precision \times Recall} \quad (2.9)$$

Dimana:

F1-Score tinggi artinya model memiliki *precision* dan *recall* yang sama-sama baik.

F1-Score rendah artinya salah satu (atau keduanya) bernilai rendah.

Persamaan tersebut menunjukkan bahwa *F1-score* akan bernilai tinggi apabila *precision* dan *recall* sama-sama tinggi, serta menurun apabila salah satu metrik memiliki nilai yang rendah [5]. Oleh karena itu, *F1-score* digunakan untuk mengevaluasi performa model klasifikasi secara lebih seimbang dan representatif [4].

Penggunaan *accuracy, precision, recall,* dan *F1-score* pada penelitian ini bertujuan untuk memberikan gambaran menyeluruh terhadap performa algoritma *Support Vector Machine* dan *Random Forest* dalam mengklasifikasikan teks hoaks berbahasa Indonesia [3]. Dengan menggunakan beberapa metrik evaluasi, perbandingan kinerja algoritma dapat dilakukan secara lebih objektif dan tidak bergantung pada salah satu indikator performa saja [1].

2.7.3 Interpretasi Nilai Evaluasi

Interpretasi nilai evaluasi diperlukan untuk memberikan makna terhadap hasil pengukuran performa model klasifikasi [5]. Nilai metrik evaluasi tidak hanya menunjukkan besar kecilnya performa model, tetapi juga mencerminkan karakteristik kesalahan yang dihasilkan oleh algoritma klasifikasi [7].

Tabel 2.4 Interpretasi Nilai Metrik Evaluasi Klasifikasi

Metrik	Rendah	Sedang	Tinggi	Interpretasi Umum
Accuracy	< 60%	60-80%	> 80%	Konsistensi prediksi model secara keseluruhan
Precision	< 60%	60-80%	> 80%	Ketepatan prediksi terhadap kelas hoaks.
Recall	< 60%	60-80%	> 80%	Kemampuan model mendeteksi seluruh data hoaks.
F1-Score	< 60%	60-80%	> 80%	Keseimbangan antara *precision* dan *recall.*

Sebagai panduan konseptual, nilai metrik evaluasi dapat dikelompokkan ke dalam kategori rendah, sedang, dan tinggi untuk membantu analisis performa model [5]. Pengelompokan ini bersifat indikatif dan tidak merepresentasikan hasil eksperimen penelitian secara langsung.

Interpretasi nilai evaluasi pada tabel tersebut digunakan sebagai acuan teoritis dalam memahami performa model klasifikasi pada penelitian ini [5]. Pembahasan hasil evaluasi secara kuantitatif berdasarkan data eksperimen akan disajikan pada BAB IV.