BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Tabel 2. 1 Penelitian Terdahulu

No	:	01
Judul Artikel	:	Comparison of Naïve Bayes and Random Forest Algorithm in Webtoon Application Sentiment Analysis
Nama Jurnal, Penulis & Tahun	:	Innovation in Research of Informatics (INNOVATICS), Vol. 6, No. 1 Admojo, F. T., Risnanto, S., Windiawati, A. W., Innuddin, M., & Mualfah, D., 2024
Metode Penelitian	:	Menggunakan dataset ulasan aplikasi Webtoon dari Google Play Store dengan dua algoritma klasifikasi, yaitu Naïve Bayes dan Random Forest, serta metrik evaluasi accuracy, precision, recall, dan F1-score.
Temuan Utama	:	Random Forest menghasilkan akurasi 88%, lebih unggul dibandingkan Naïve Bayes dengan akurasi 74%. Random Forest direkomendasikan sebagai algoritma klasifikasi sentimen untuk aplikasi Webtoon.
Relevansi Terhadap Project Ini	:	Relevan karena menggunakan objek penelitian yang sama (aplikasi Webtoon) dan membuktikan efektivitas Random Forest. Penelitian ini menjadi dasar justifikasi pemilihan Random Forest sebagai salah satu algoritma dalam penelitian ini, sekaligus menunjukkan bahwa penelitian sebelumnya belum menganalisis pengaruh konfigurasi n-gram pada TF-IDF.
Referensi	:	[11]
No	:	02
Judul Artikel	:	Aspect-Based Sentiment Analysis on Webtoon Reviews Using Ensemble Learning
Nama Jurnal, Penulis & Tahun	:	JASMINE: Journal of Intelligent Systems and Machine Learning

		Amrullah, R. F., Mahardika, F., & Junaedi, D. I., 2026
Metode Penelitian	:	Menggunakan framework Aspect-Based Sentiment Analysis (ABSA) dengan anotasi manual pada tiga aspek (plot, karakter, visual), ekstraksi fitur TF-IDF, serta pelatihan model ensemble Random Forest dan XGBoost dengan tuning hyperparameter dan validasi 10-fold cross-validation.
Temuan Utama	:	Model Random Forest ABSA dengan kombinasi fitur TF-IDF dan One-Hot Encoded aspect menghasilkan akurasi terbaik 65,84% dengan weighted F1-score 0,65. Sentimen negatif merupakan kelas yang paling sulit diklasifikasikan karena ekspresi informal dan konteks yang beragam.
Relevansi Terhadap Project Ini	:	Relevan karena menggunakan objek dan algoritma yang sama (Webtoon, Random Forest, XGBoost, TF-IDF). Penelitian ini membuktikan bahwa TF-IDF efektif digunakan pada ulasan Webtoon, namun belum mengeksplorasi pengaruh variasi konfigurasi n-gram terhadap performa model, yang menjadi fokus utama penelitian ini.
Referensi	:	[12]
No	:	03
Judul Artikel	:	Perbandingan Logistic Regression dan SVM untuk Analisis Sentimen Pengguna Netflix Menggunakan TF-IDF dan Bot Telegram
Nama Jurnal, Penulis & Tahun	:	RIGGS: Journal of Artificial Intelligence and Digital Business, Vol. 4, No. 4, pp. 2559–2566 Ngara, S. M., Anggara, I. G., Saputra, R. A., Rahmatullah, B., Wahidin, A. J., & Kurniawati, I., 2026
Metode Penelitian	:	Menggunakan 5.620 ulasan pengguna Netflix berbahasa Indonesia dari Google Play Store. Preprocessing mencakup pembersihan teks, case folding, tokenisasi, stopword removal, dan stemming. Klasifikasi menggunakan Logistic Regression dan SVM dengan representasi teks TF-IDF. Implementasi sistem melalui dashboard Streamlit dan bot Telegram.
Temuan Utama	:	SVM menghasilkan performa tertinggi dengan akurasi 86,03%, F1-score 77,73%, precision 86,98%, recall 70,26%, dan ROC-AUC 92,06%. Logistic Regression mencatat akurasi 85,32% dan F1-score 76,26%.

Relevansi Terhadap Project Ini	:	Relevan karena membuktikan efektivitas SVM dengan TF-IDF pada analisis sentimen ulasan aplikasi berbahasa Indonesia dari Google Play Store, yang mendukung pemilihan SVM sebagai salah satu algoritma dalam penelitian ini. Penelitian ini juga menggunakan implementasi Streamlit yang serupa, namun tidak menguji variasi konfigurasi n-gram.
Referensi	:	[13]
No	:	04
Judul Artikel	:	Sentiment Analysis for Customer Review: Case Study of Traveloka
Nama Jurnal, Penulis & Tahun	:	Procedia Computer Science, Vol. 216, pp. 682–690 Diekson, Z. A., Prakoso, M. R. B., Putra, M. S. Q., Syaputra, M. S. A. F., Achmad, S., & Sutoyo, R., 2022
Metode Penelitian	:	Menggunakan 1.200 tweet terkait Traveloka yang diambil melalui Twitter API. Analisis dilakukan dengan tiga metode klasifikasi: Support Vector Machine (SVM), Naive Bayes, dan Logistic Regression, menggunakan library Scikit-learn.
Temuan Utama	:	SVM menunjukkan akurasi terbaik dibandingkan Naive Bayes dan Logistic Regression, dengan akurasi di atas 75% pada seluruh dataset kandidat yang diuji.
Relevansi Terhadap Project Ini	:	Penelitian ini relevan sebagai referensi penerapan SVM dalam analisis sentimen aplikasi digital (Traveloka). Namun, berbeda dengan penelitian ini, penelitian tersebut menggunakan data dari Twitter (bukan Google Play Store) dan tidak menguji pengaruh representasi TF-IDF dengan variasi konfigurasi n-gram, sehingga penelitian ini tetap relevan untuk melengkapi celah tersebut.
Referensi	:	[17]
No	:	05
Judul Artikel	:	Actual Rating Calculation of the Zoom Cloud Meetings App Using User Reviews on Google Play Store with Sentiment Annotation of BERT and Hybridization of RNN and LSTM
Nama Jurnal, Penulis & Tahun	:	Expert Systems with Applications, Vol. 223, Art. 119919 Islam, M. J., Datta, R., & Iqbal, A., 2023

Metode Penelitian	:	Menggunakan ulasan aplikasi Zoom Cloud Meetings dari Google Play Store. Anotasi sentimen dilakukan menggunakan BERT, kemudian dilakukan klasifikasi dan prediksi rating aktual aplikasi menggunakan model hybrid RNN dan LSTM.
Temuan Utama	:	Model hybrid RNN-LSTM dengan anotasi sentimen berbasis BERT mampu menghitung rating aktual aplikasi secara lebih representatif dibandingkan rating rata-rata yang ditampilkan di Play Store.
Relevansi Terhadap Project Ini	:	Penelitian ini relevan sebagai referensi dalam analisis sentimen ulasan aplikasi digital dari Google Play Store. Namun, berbeda dengan penelitian ini yang menggunakan pendekatan deep learning (BERT, RNN, LSTM), penelitian ini berfokus pada pendekatan machine learning klasik (SVM, Random Forest, XGBoost) dengan representasi TF-IDF dan variasi konfigurasi n-gram, sehingga tetap relevan sebagai pembanding pendekatan deep learning vs machine learning klasik pada domain ulasan aplikasi digital.
Referensi	:	[18]
No	:	06
Judul Artikel	:	Self Voting Classification Model for Online Meeting App Review Sentiment Analysis and Topic Modeling
Nama Jurnal, Penulis & Tahun	:	PeerJ Computer Science, Vol. 8, Art. e1141 Aslam, N., et al., 2022
Metode Penelitian	:	Menggunakan ulasan aplikasi meeting online (Zoom, Google Meet, Microsoft Teams, dll.) dari Google Play Store dan Apple App Store. Fitur diekstraksi menggunakan tiga skema representasi teks (Bag of Words, TF-IDF, dan Hashing Vectorizer), diuji dengan algoritma SVM, Decision Tree, Logistic Regression, dan KNN, kemudian diusulkan model ensemble self-voting.
Temuan Utama	:	Model self-voting yang mengombinasikan beberapa classifier mengungguli performa masing-masing model individual dalam klasifikasi sentimen ulasan aplikasi.
Relevansi Terhadap Project Ini	:	Penelitian ini relevan karena menguji representasi TF-IDF dengan beberapa algoritma klasifikasi (termasuk SVM) pada ulasan aplikasi digital dari Google Play Store. Perbedaannya, penelitian tersebut tidak menguji variasi konfigurasi n-gram (unigram, bigram, trigram) secara spesifik pada representasi

		TF-IDF, sehingga celah inilah yang menjadi pembeda dan kontribusi dari penelitian ini.
Referensi	:	[19]
No	:	07
Judul Artikel	:	Sentiment Analysis of Google Play Store Reviews using Support Vector Machines
Nama Jurnal, Penulis & Tahun	:	International Journal of Applied Information Systems, Vol. 12, No. 42, pp. 48–53 Idris, M., 2024
Metode Penelitian	:	Menggunakan ulasan aplikasi Fintech iGrow dari Google Play Store dengan algoritma Support Vector Machine (SVM) menggunakan dua jenis kernel, yaitu Linear Kernel dan RBF Kernel. Data diproses menggunakan teknik data mining untuk klasifikasi sentimen positif dan negatif.
Temuan Utama	:	SVM dengan Linear Kernel dan RBF Kernel menghasilkan akurasi yang sama sebesar 77%. SVM dengan RBF Kernel lebih baik dalam mengklasifikasikan ulasan positif, namun masih terdapat ruang perbaikan pada precision untuk klasifikasi sentimen negatif.
Relevansi Terhadap Project Ini	:	Relevan sebagai referensi penggunaan SVM pada analisis sentimen ulasan Google Play Store. Penelitian ini mendukung pemilihan SVM sebagai algoritma dalam penelitian ini dan menunjukkan bahwa SVM dengan kernel linear cukup efektif untuk data ulasan aplikasi.
Referensi	:	[20]
No	:	08
Judul Artikel	:	Ensemble Approach to Sentiment Analysis of Google Play Store App Reviews
Nama Jurnal, Penulis & Tahun	:	Mustofa, Y. A., dan Idris, I. S. K. (2024). *Jambura Journal of Electrical and Electronics Engineering*, Vol. 6, No. 2, halaman 181–188.
Metode Penelitian	:	Menggunakan ulasan aplikasi Zoom dan Shopee dari Google Play Store dengan metode ensemble (Random Forest dan Boosting) dibandingkan algoritma individual (Naïve Bayes dan

		SVM). Preprocessing mencakup cleaning, case folding, tokenisasi, stopword removal, dan normalisasi.
Temuan Utama	:	Random Forest menghasilkan performa tertinggi dengan akurasi 94,15% pada ulasan Zoom dan 80,69% pada ulasan Shopee, mengkonfirmasi keunggulan pendekatan ensemble dibandingkan algoritma individual dalam menangani kompleksitas dan variabilitas data ulasan.
Relevansi Terhadap Project Ini	:	Relevan sebagai referensi yang membuktikan keunggulan Random Forest sebagai metode ensemble pada analisis sentimen ulasan Google Play Store. Penelitian ini memperkuat justifikasi pemilihan Random Forest dan XGBoost sebagai algoritma ensemble dalam penelitian ini.
Referensi	:	[21]
No	:	09
Judul Artikel	:	Improving Performance Sentiment Movie Review Classification Using Hybrid Feature TF-IDF, N-Gram, Information Gain and Support Vector Machine
Nama Jurnal, Penulis & Tahun	:	Mathematical Modelling of Engineering Problems, Vol. 11, No. 2 Alamin, Z., Lorosae, T. A., & Ramadhan, S., 2024
Metode Penelitian	:	Menggunakan ulasan film dari platform streaming online dengan model hybrid TF-IDF+N-Gram sebagai representasi fitur, dilanjutkan seleksi fitur menggunakan Information Gain (IG), dan klasifikasi menggunakan Support Vector Machine (SVM). Konfigurasi n-gram yang diuji adalah bigram dan trigram.
Temuan Utama	:	TF-IDF+Bigram+IG mencapai akurasi 78% (meningkat 8% dari baseline 70%), dan TF-IDF+Trigram+IG mencapai akurasi 66% (meningkat 22% dari baseline 44%). Kombinasi TF-IDF dengan n-gram terbukti secara signifikan meningkatkan akurasi klasifikasi sentimen.
Relevansi Terhadap Project Ini	:	Sangat relevan karena secara langsung membuktikan bahwa kombinasi TF-IDF dengan konfigurasi n-gram yang berbeda menghasilkan performa yang berbeda pula. Temuan ini menjadi landasan empiris utama yang mendukung tujuan penelitian ini untuk menganalisis pengaruh berbagai konfigurasi n-gram pada TF-IDF terhadap performa klasifikasi sentimen ulasan Webtoon.

Referensi	:	[9]
No	:	10
Judul Artikel	:	The Optimization of n-Gram Feature Extraction Based on Term Occurrence for Cyberbullying Classification
Nama Jurnal, Penulis & Tahun	:	Data Science Journal, Vol. 23, No. 1, Art. 31 Setiawan, Y., Maulidevi, N. U., & Surendro, K., 2024
Metode Penelitian	:	Menggunakan dataset cyberbullying berbahasa Inggris (8.129 tweet dari Twitter, kategori sarkasme dan ironi). Ekstraksi fitur dilakukan dengan TF-IDF menggunakan tiga konfigurasi n-gram (unigram, bigram, trigram), divalidasi dengan enam algoritma klasifikasi: SVM, KNN, Linear Regression, Naive Bayes, dan Random Forest, dengan variasi parameter Min-DF dan Max-DF.
Temuan Utama	:	SVM memperoleh performa terbaik pada fitur TF-IDF Unigram (akurasi hingga 99,67%), sementara Random Forest unggul secara konsisten pada fitur TF-IDF Bigram di seluruh kelompok data (akurasi hingga 98,67%). Hasil ini menunjukkan bahwa performa algoritma optimal dapat berbeda tergantung konfigurasi n-gram yang digunakan.
Relevansi Terhadap Project Ini	:	relevan karena secara langsung menguji pengaruh variasi konfigurasi n-gram (unigram, bigram, trigram) pada representasi TF-IDF terhadap performa berbagai algoritma klasifikasi, termasuk SVM dan Random Forest — memperkuat dasar teoritis dan metodologis penelitian ini. Perbedaannya, penelitian tersebut diterapkan pada domain deteksi cyberbullying berbahasa Inggris, sedangkan penelitian ini berfokus pada analisis sentimen ulasan aplikasi Webtoon berbahasa Indonesia dan turut menguji algoritma XGBoost serta konfigurasi kombinasi n-gram (unigram+bigram, unigram+bigram+trigram).
Referensi	:	[10]

Berdasarkan sepuluh penelitian terdahulu yang telah dipaparkan, terdapat keterhubungan yang jelas antar artikel: penelitian [11], [12], [20], [21] sama-sama menggunakan objek penelitian berupa ulasan aplikasi digital dari *Google Play Store* (termasuk aplikasi *Webtoon* secara langsung pada [11] dan [12]) dengan algoritma klasifikasi yang menjadi fokus penelitian ini, yaitu *SVM*, *Random Forest*, dan/atau

XGBoost; penelitian [13], [17], [18], [19] memperluas konteks pembanding pada domain aplikasi digital sejenis (*Netflix*, *Traveloka*, *Zoom*, aplikasi *meeting online*) dengan representasi fitur *TF-IDF*; sementara penelitian [9] dan [10] secara spesifik membuktikan bahwa variasi konfigurasi *n-gram* (*bigram*, *trigram*) pada representasi *TF-IDF* menghasilkan performa klasifikasi yang berbeda-beda pada berbagai algoritma, termasuk *SVM* dan *Random Forest*.

Dari kesepuluh penelitian tersebut, terdapat beberapa hal yang diadopsi ke dalam penelitian ini: penggunaan algoritma *SVM*, *Random Forest*, dan *XGBoost* sebagai model klasifikasi (diadopsi dari [11], [20], [21]); penggunaan *TF-IDF* sebagai metode representasi fitur teks (diadopsi dari [13], [17], [18], [19]); serta pendekatan pengujian berbagai konfigurasi *n-gram* pada *TF-IDF* (diadopsi dan dikembangkan lebih lanjut dari [9] dan [10]).

Namun, penelitian ini memiliki sejumlah pembeda mendasar dari seluruh penelitian terdahulu tersebut. Pertama, dari segi objek dan bahasa data, penelitian ini berfokus pada ulasan aplikasi *Webtoon* berbahasa Indonesia dari *Google Play Store*, berbeda dengan [9], [10], [17], [18] yang menggunakan domain lain (film, *cyberbullying*, *Traveloka*, *Zoom*) maupun bahasa lain (Inggris). Kedua, dari segi metodologi, tidak satu pun penelitian terdahulu yang menguji secara komprehensif kelima konfigurasi *n-gram* (*unigram*, *bigram*, *trigram*, *unigram+bigram*, dan *unigram+bigram+trigram*) secara sistematis pada ketiga algoritma klasifikasi (*SVM*, *Random Forest*, dan *XGBoost*) sekaligus dalam satu kerangka penelitian yang sama—sebagian penelitian hanya menguji satu atau dua algoritma ([11], [13], [20]), sebagian hanya menguji konfigurasi *n-gram* secara terbatas tanpa variasi kombinasi ([9], [10], [12]), dan sebagian lain tidak menguji *n-gram* sama sekali ([17], [18], [19], [21]). Dengan demikian, penelitian ini mengisi celah penelitian (*research gap*) berupa analisis komprehensif pengaruh variasi konfigurasi *n-gram* pada representasi *TF-IDF* terhadap performa tiga algoritma klasifikasi sekaligus, khususnya pada domain ulasan aplikasi *Webtoon* berbahasa Indonesia.

2.2 Teori yang berkaitan

2.2.1 Sentiment Analysis

Sentiment analysis atau analisis sentimen merupakan metode yang digunakan untuk mengekstrak informasi mengenai sikap atau opini seseorang terhadap suatu topik berdasarkan data teks, baik dalam bentuk sentimen positif, netral, maupun negatif. Sentimen sendiri merupakan opini pribadi yang disampaikan pengguna internet sebagai bentuk penilaian terhadap suatu produk, layanan, aplikasi, maupun isu tertentu. Seiring meningkatnya penggunaan media sosial dan platform digital, jumlah opini yang dihasilkan pengguna menjadi sangat besar sehingga proses identifikasi sentimen secara manual menjadi sulit dilakukan. Oleh karena itu, analisis sentimen berperan penting dalam mengklasifikasikan polaritas teks berdasarkan kecenderungan sentimen yang terkandung di dalamnya. Selain menentukan polaritas positif, negatif, atau netral, analisis sentimen juga dapat digunakan untuk mengidentifikasi ekspresi emosional seperti kesedihan, kegembiraan, maupun kemarahan yang terdapat dalam suatu teks[22].

Sentiment analysis dapat dikategorikan berdasarkan metode yang digunakan ke dalam tiga pendekatan utama. Pertama, pendekatan berbasis leksikon yang mengandalkan kamus sentimen berisi daftar kata beserta polaritasnya untuk menentukan sentimen sebuah teks secara langsung tanpa proses pembelajaran. Kedua, pendekatan berbasis *machine learning* yang memanfaatkan algoritma pembelajaran mesin seperti *Support Vector Machine*, *Naive Bayes,* dan *Random Forest* yang dilatih menggunakan data berlabel untuk membangun model klasifikasi yang mampu menggeneralisasi pola sentimen. Ketiga, pendekatan *deep learning* yang menggunakan arsitektur jaringan saraf tiruan dalam seperti *LSTM*, *BERT,* dan model *transformer* untuk memahami konteks semantik yang lebih kompleks dan nuansa bahasa yang halus. Pendekatan berbasis *machine learning* dinilai paling sesuai untuk penelitian berskala menengah karena menawarkan keseimbangan yang baik antara performa klasifikasi dan interpretabilitas model[7].

Dalam hal tingkatan analisis, *sentiment analysis* dibedakan ke dalam tiga level yang berbeda sesuai dengan granularitas unit teks yang dianalisis. *Document-level sentiment analysis* mengklasifikasikan keseluruhan dokumen ke dalam satu kategori sentimen tunggal, cocok untuk analisis ulasan yang setiap ulasan mewakili satu opini keseluruhan. *Sentence-level sentiment analysis* menganalisis sentimen pada level kalimat individual dalam sebuah dokumen. *Aspect-level sentiment analysis* mengidentifikasi sentimen terhadap aspek-aspek spesifik yang disebutkan dalam teks, seperti kualitas layanan, harga, atau antarmuka pengguna. Dalam penelitian ini, analisis dilakukan pada tingkatan dokumen, di mana setiap ulasan pengguna aplikasi *Webtoon* diperlakukan sebagai satu unit dokumen yang diklasifikasikan secara menyeluruh ke dalam kategori sentimen positif atau negatif.

Penerapan *sentiment analysis* pada ulasan aplikasi digital memberikan manfaat strategis yang signifikan bagi pengembang. Melalui analisis sentimen otomatis, pengembang dapat memantau persepsi pengguna dalam skala besar secara *real-time*, mengidentifikasi fitur-fitur yang mendapat respons positif atau negatif dari pengguna, memprioritaskan perbaikan berdasarkan data nyata, serta merespons keluhan pengguna secara lebih cepat dan tepat sasaran. Hal ini pada akhirnya berkontribusi pada peningkatan kualitas aplikasi dan kepuasan pengguna secara berkelanjutan[23].

2.2.2 Text Mining

Text mining adalah proses penemuan pengetahuan yang bermakna dan berguna dari kumpulan data teks tidak terstruktur dalam jumlah besar melalui serangkaian teknik komputasi yang sistematis. Berbeda dari pencarian informasi konvensional yang hanya mengambil kembali informasi yang sudah ada secara eksplisit dalam teks, *text mining* bertujuan untuk menemukan pola tersembunyi, hubungan antar konsep, tren, serta wawasan baru yang tidak

tersurat secara langsung dalam teks. Dengan kata lain, *text mining* mentransformasi teks tidak terstruktur menjadi pengetahuan terstruktur yang dapat digunakan untuk pengambilan keputusan.

Pipeline text mining pada umumnya terdiri dari beberapa tahapan yang saling berkaitan secara berurutan. Tahap pertama adalah pengumpulan data (*data collection*), di mana teks mentah dikumpulkan dari berbagai sumber seperti media sosial, forum diskusi, situs ulasan, atau basis data teks. Tahap kedua adalah *preprocessing* teks yang mencakup pembersihan, normalisasi, dan transformasi teks agar siap diproses oleh algoritma selanjutnya. Tahap ketiga adalah representasi fitur (*feature representation*), di mana teks yang telah diproses diubah menjadi representasi numerik menggunakan teknik seperti *TF-IDF*, *Bag of Words*, atau *word embeddings*. Tahap keempat adalah penerapan algoritma *machine learning* untuk berbagai tugas analitik seperti klasifikasi, pengelompokan (*clustering*), atau ekstraksi informasi. Tahap kelima adalah interpretasi dan evaluasi hasil untuk memastikan kualitas dan relevansi pengetahuan yang berhasil ditemukan[24].

Dalam konteks penelitian ini, *text mining* diterapkan secara menyeluruh pada ulasan pengguna aplikasi *Webtoon* yang merupakan data teks tidak terstruktur hasil *scraping* dari *Google Play Store*. Proses *text mining* memungkinkan pengolahan 10.000 ulasan secara otomatis yang tidak mungkin dilakukan secara manual dengan efisien dan konsisten. Keberhasilan *text mining* sangat bergantung pada kualitas tahap *preprocessing*, karena data teks dari platform digital seperti *Google Play Store* umumnya mengandung banyak *noise* berupa singkatan tidak baku, ejaan tidak standar, simbol, *emoji*, dan karakter khusus yang harus dibersihkan sebelum analisis lebih lanjut dapat dilakukan.

2.2.3 Natural Language Processing (NLP)

Natural Language Processing (*NLP*) merupakan sub-bidang dari kecerdasan buatan (*Artificial Intelligence*) yang berfokus pada pengembangan metode dan sistem komputasional yang memungkinkan komputer untuk memahami, menginterpretasikan, memanipulasi, dan menghasilkan bahasa manusia secara alami dan otomatis. *NLP* menjembatani kesenjangan antara komunikasi manusia yang bersifat alami, ambigu, dan tidak terstruktur dengan kemampuan komputer yang membutuhkan input yang terstruktur dan formal. *NLP* menggabungkan ilmu linguistik komputasional, ilmu kognitif, statistik, dan *machine learning* untuk memproses serta menganalisis data bahasa alami dalam skala besar.

Terdapat beberapa tugas utama dalam *NLP* yang menjadi fondasi berbagai aplikasi modern. *Tokenisasi* adalah proses memecah teks menjadi unit-unit bermakna berupa kata atau subkata yang menjadi satuan dasar analisis. *Part-of-speech tagging* adalah proses mengidentifikasi peran gramatikal setiap kata dalam kalimat, seperti kata benda, kata kerja, atau kata sifat. *Named entity recognition* adalah proses mengidentifikasi dan mengklasifikasikan entitas yang disebutkan dalam teks, seperti nama orang, tempat, organisasi, atau tanggal. *Sentiment analysis* adalah tugas mengklasifikasikan polaritas opini yang terkandung dalam teks. *Machine translation* adalah proses menerjemahkan teks secara otomatis dari satu bahasa ke bahasa lain. *Text summarization* adalah proses meringkas dokumen panjang menjadi representasi yang lebih singkat namun tetap informatif[25]. Dalam penelitian ini, teknik-teknik *NLP* khususnya *tokenisasi, stopword removal,* dan *stemming* digunakan secara langsung dalam tahap *preprocessing* data ulasan *Webtoon* berbahasa Indonesia.

Pemrosesan bahasa alami yang bersifat informal seperti ulasan pengguna di platform digital menghadirkan tantangan-tantangan khusus yang perlu diatasi.

Teks ulasan pengguna umumnya mengandung bahasa gaul dan tidak baku, singkatan yang tidak terstandarisasi, pengulangan karakter untuk penekanan emosi (seperti “bagussss” atau “kereennnn”), campuran bahasa Indonesia dan Inggris dalam satu kalimat (*code-switching*), serta kesalahan ejaan yang tidak konsisten. Tantangan-tantangan ini membuat tahap *preprocessing* yang tepat dan komprehensif menjadi sangat krusial dalam *pipeline NLP* untuk data ulasan aplikasi berbahasa Indonesia[26].

2.2.4 Text Preprocessing

Text preprocessing merupakan tahapan fundamental dalam pipeline *text mining* dan *Natural Language Processing* (NLP) yang bertujuan untuk membersihkan, menormalkan, dan mentransformasi data teks mentah menjadi format yang lebih terstruktur dan siap diolah oleh algoritma *machine learning*. Proses ini sangat penting karena data teks pada umumnya bersifat tidak terstruktur, mengandung variasi penulisan, serta memiliki banyak elemen yang tidak relevan untuk analisis. Menurut penelitian sebelumnya, kualitas preprocessing memiliki pengaruh signifikan terhadap performa model klasifikasi teks karena tahap ini menentukan kualitas representasi fitur yang dihasilkan [27]. Dengan demikian, preprocessing menjadi langkah krusial untuk memastikan bahwa informasi penting dalam teks dapat diekstraksi secara optimal.

Data teks mentah yang berasal dari sumber digital seperti ulasan aplikasi, media sosial, maupun forum daring umumnya mengandung *noise* dalam berbagai bentuk, seperti karakter khusus, URL, angka, emoji, serta tanda baca yang tidak relevan. Keberadaan *noise* ini dapat mengganggu proses ekstraksi fitur dan menurunkan akurasi model jika tidak ditangani dengan baik. Oleh karena itu, preprocessing dilakukan tidak hanya untuk membersihkan data, tetapi juga untuk meningkatkan kualitas dan konsistensi teks. Selain itu, proses

ini juga berperan dalam mereduksi kompleksitas data sehingga mempermudah tahap analisis selanjutnya [27].

Lebih lanjut, preprocessing juga berkontribusi dalam mengurangi dimensi ruang fitur yang dihasilkan dari representasi teks seperti TF-IDF. Representasi berbasis TF-IDF cenderung menghasilkan matriks berdimensi tinggi dan bersifat *sparse*, terutama ketika dataset memiliki ukuran besar dan kosakata yang beragam. Dengan menerapkan teknik preprocessing seperti *case folding*, *tokenization*, *stopword removal*, dan *stemming*, jumlah fitur yang tidak relevan dapat dikurangi secara signifikan tanpa menghilangkan makna utama dari teks. Hal ini memungkinkan model untuk bekerja lebih efisien serta meningkatkan kemampuan generalisasi terhadap data baru [28]. Oleh karena itu, penerapan preprocessing yang tepat menjadi faktor penting dalam membangun sistem klasifikasi teks yang akurat dan andal.

2.2.4.1 Case Folding

Case folding adalah proses mengubah seluruh karakter huruf dalam teks menjadi huruf kecil (*lowercase*) secara seragam dan konsisten. *Case folding* merupakan langkah *preprocessing* paling dasar namun sangat penting karena komputer secara inheren memperlakukan kata dengan kapitalisasi berbeda sebagai *token* yang sepenuhnya berbeda. Tanpa penerapan *case folding*, kata “Bagus”, “BAGUS”, dan “bagus” akan diperlakukan sebagai tiga entitas fitur yang berbeda meskipun secara semantik dan linguistik identik, sehingga frekuensi kemunculan kata yang seharusnya dijumlahkan menjadi terpecah ke dalam tiga fitur terpisah yang masing-masing memiliki bobot lebih rendah. Pada data ulasan pengguna yang diketik secara bebas tanpa aturan penulisan yang ketat, variasi kapitalisasi sangat umum terjadi, termasuk penggunaan huruf kapital seluruhnya untuk penekanan emosi atau penggunaan huruf kapital di awal kata secara acak. Proses *case folding* menyelesaikan

seluruh masalah ini dengan menyeragamkan representasi seluruh *token* menjadi *lowercase* sehingga frekuensi kemunculan kata dapat dihitung secara akurat dan konsisten[29].

2.2.4.2 Cleaning

Cleaning atau pembersihan teks adalah proses sistematis untuk menghapus elemen-elemen yang tidak relevan dan berpotensi mengganggu proses analisis dari data teks. Beberapa jenis *noise* yang umumnya ditemukan dalam data ulasan platform digital dan perlu dihapus pada tahap ini antara lain: (1) URL dan tautan web yang tidak mengandung informasi sentimen yang bermakna; (2) karakter *non-ASCII* termasuk *emoji* dan simbol *Unicode* yang tidak dapat direpresentasikan dalam format teks standar ASCII dan tidak dapat diproses oleh algoritma berbasis teks; (3) angka dan karakter numerik yang pada umumnya tidak berkontribusi secara langsung pada analisis sentimen berbasis kata; (4) tanda baca dan karakter khusus seperti tanda seru, tanda tanya, kurung, dan simbol lainnya yang dapat mengganggu proses *tokenisasi*; serta (5) spasi berlebih (*multiple whitespace*) yang timbul sebagai efek samping dari penghapusan elemen-elemen sebelumnya dan perlu dinormalisasi menjadi satu spasi tunggal. Proses *cleaning* yang komprehensif memastikan bahwa hanya *token-token* yang mengandung informasi linguistik bermakna yang tersisa untuk diproses pada tahapan selanjutnya[29].

2.2.4.3 Tokenisasi

Tokenisasi adalah proses memecah rangkaian teks menjadi unit-unit yang lebih kecil dan bermakna yang disebut *token*, di mana setiap *token* merepresentasikan satu unit analisis dasar dalam *pipeline NLP*. *Token*

pada umumnya berupa kata-kata individual, namun dapat pula berupa frasa, kalimat, atau bahkan subkata tergantung pada level tokenisasi yang diterapkan sesuai dengan kebutuhan analisis. Untuk teks berbahasa Indonesia, tokenisasi pada level kata dilakukan dengan memecah teks berdasarkan karakter spasi sebagai pemisah antar kata, menghasilkan daftar kata-kata individual dari setiap ulasan yang kemudian dapat diproses lebih lanjut. *Tokenisasi* merupakan prasyarat mutlak untuk semua tahapan *preprocessing* selanjutnya seperti *stopword removal* dan *stemming*, karena kedua proses tersebut bekerja pada level *token* individual satu per satu. Tanpa tokenisasi yang tepat, proses penghapusan *stopword* dan pencocokan kata dengan kamus *stemming* tidak dapat dilakukan secara akurat[29].

2.2.4.4 Stopword Removal

Stopword removal adalah proses menghilangkan kata-kata yang sangat umum dan sering muncul dalam teks namun tidak mengandung makna sentimen yang signifikan untuk keperluan klasifikasi. *Stopword* adalah kata-kata fungsional dalam bahasa yang berfungsi sebagai penghubung gramatikal tanpa membawa muatan semantik yang informatif untuk analisis sentimen, seperti kata sandang, kata depan, kata ganti, konjungsi, dan kata bantu. Dalam bahasa Indonesia, contoh *stopword* yang umum meliputi kata-kata seperti “yang”, “dan”, “di”, “ke”, “dari”, “ini”, “itu”, “juga”, “sudah”, “akan”, “dengan”, “untuk”, “pada”, “adalah”, dan sejenisnya. Penghapusan *stopword* memberikan dua manfaat utama yang saling berkaitan: pertama, mereduksi dimensi ruang fitur secara signifikan sehingga model dapat dilatih dengan lebih efisien dan tidak terbebani oleh fitur-fitur yang tidak informatif; kedua, meningkatkan kualitas fitur yang tersisa dengan memfokuskan bobot *TF-IDF* pada kata-kata yang benar-benar membawa informasi sentimen seperti kata sifat, kata kerja bermakna, dan *adverbia intensifikasi*[29].

2.2.4.5 Stemming

Stemming adalah proses mereduksi kata berimbuhan ke bentuk dasar atau akar katanya (*stem*) melalui penghapusan afiks secara sistematis berdasarkan aturan morfologi bahasa yang bersangkutan. Afiks yang dihapus mencakup awalan (*prefiks*) seperti *me-*, *di-*, *ber-*, *ter-*, *ke-*, *se-*, akhiran (*sufiks*) seperti *-kan*, *-an*, *-i*, *-nya*, serta kombinasi awalan dan akhiran (*konfiks*) seperti *pe-an*, *per-an*, dan *ke-an*. Tujuan utama *stemming* adalah menggabungkan berbagai bentuk morfologis dari kata yang sama ke dalam satu representasi tunggal, sehingga kata "menggunakan", "digunakan", "penggunaan", dan "gunakan" semuanya dipetakan ke bentuk dasar "guna". Dengan demikian, seluruh variasi morfologis dari kata yang sama tidak lagi menghasilkan fitur-fitur terpisah yang redundan dalam matriks *TF-IDF*, melainkan berkontribusi pada satu fitur tunggal yang memiliki bobot lebih tinggi dan lebih representatif. Dalam penelitian ini, *stemming* dilakukan menggunakan library *PySastrawi* yang merupakan implementasi algoritma *Enhanced Confix Stripping* (*ECS*) yang dirancang khusus untuk menangani kompleksitas morfologi bahasa Indonesia yang kaya dengan sistem pengimbuhan yang sangat beragam[30].

2.2.5 Google Play Store

Google Play Store adalah platform distribusi konten digital resmi yang dikembangkan dan dioperasikan oleh Google LLC, diperuntukkan bagi seluruh perangkat yang menjalankan sistem operasi *Android*. Platform ini merupakan *marketplace* digital terbesar untuk aplikasi *mobile* di dunia dengan jutaan aplikasi yang tersedia dari berbagai kategori, mulai dari produktivitas, hiburan, pendidikan, kesehatan, hingga *game*. *Google Play Store* memungkinkan pengembang dari seluruh dunia untuk mendistribusikan aplikasi mereka kepada

ratusan juta pengguna *Android* secara global melalui satu platform yang terpusat dan mudah diakses.

Salah satu fitur terpenting *Google Play Store* yang menjadi relevan dengan penelitian ini adalah sistem *rating* dan ulasan pengguna yang terintegrasi. Setiap pengguna yang telah mengunduh dan menggunakan sebuah aplikasi dapat memberikan penilaian berupa bintang pada skala 1 hingga 5 bintang, di mana 1 bintang berarti sangat buruk dan 5 bintang berarti sangat baik, disertai dengan ulasan tekstual yang mendeskripsikan pengalaman penggunaan secara kualitatif dan personal. Setiap ulasan yang tersimpan di *Google Play Store* memiliki beberapa atribut yang dapat diakses secara publik, antara lain konten teks ulasan, skor *rating* bintang, nama pengguna, tanggal ulasan dibuat, jumlah *thumbs up* dari pengguna lain, dan versi aplikasi yang sedang digunakan saat ulasan dibuat.

Data ulasan di *Google Play Store* bersifat publik dan dapat diakses oleh siapapun melalui halaman aplikasi, sehingga dapat dikumpulkan untuk keperluan penelitian akademis menggunakan teknik *web scraping*. Karakteristik data ulasan *Google Play Store* yang perlu diperhatikan dalam penelitian adalah variasi panjang teks yang sangat besar mulai dari satu kata hingga beberapa paragraf, distribusi *rating* yang umumnya tidak seimbang dengan dominasi *rating* tinggi (4–5 bintang), serta kandungan bahasa informal yang kaya dengan singkatan dan variasi ejaan[31]. Dalam penelitian ini, *Google Play Store* berfungsi sebagai sumber data utama untuk pengumpulan 10.000 ulasan pengguna aplikasi *Webtoon* berbahasa Indonesia.

2.2.6 Webtoon

Webtoon adalah format komik digital yang dirancang secara khusus untuk dikonsumsi melalui perangkat *mobile* dengan orientasi vertikal yang mengharuskan pembaca melakukan *scroll* ke bawah untuk membaca panel-

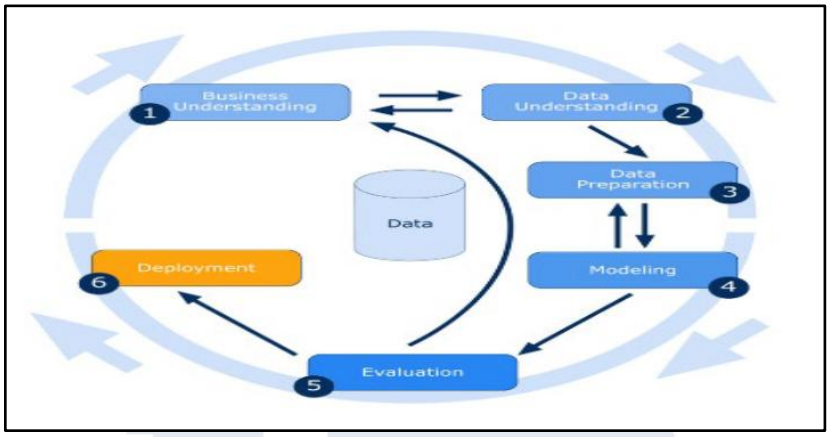
panel komik secara berurutan. Format ini berbeda secara fundamental dari komik cetak konvensional yang menggunakan orientasi halaman horizontal dan dibaca secara manual halaman per halaman. Aplikasi *Webtoon* dikembangkan dan dioperasikan oleh NAVER Corporation, sebuah perusahaan teknologi multinasional berbasis di Korea Selatan, dan telah berkembang menjadi salah satu platform distribusi komik digital terbesar dan paling berpengaruh di dunia sejak pertama kali diluncurkan pada tahun 2004.

Platform *Webtoon* menyediakan konten komik digital dari berbagai genre yang sangat beragam, termasuk *romance*, *action*, *thriller*, *fantasy*, *horror*, *comedy*, *mystery*, dan *slice of life*, yang dapat diakses pengguna secara gratis maupun melalui sistem pembelian koin untuk episode premium. *Webtoon* telah memiliki lebih dari 35 juta pengguna aktif di seluruh dunia dengan sekitar 6 juta pengguna aktif berasal dari Indonesia, menjadikan Indonesia sebagai salah satu pasar terbesar platform ini di kawasan Asia Tenggara. Data *Google Play Store* menunjukkan bahwa *Webtoon* menduduki posisi pertama dalam kategori *Top Free Comics* di Indonesia, mengungguli puluhan aplikasi komik digital lainnya.

Karakteristik khas yang membedakan ulasan pengguna aplikasi *Webtoon* dari ulasan aplikasi digital lainnya adalah keberagaman topik yang dibahas dalam setiap ulasan. Pengguna tidak hanya mengomentari aspek teknis aplikasi seperti performa, stabilitas, kecepatan *loading*, dan antarmuka pengguna, tetapi juga sangat sering mengekspresikan pendapat mereka tentang kualitas konten cerita dari judul-judul komik tertentu, kualitas ilustrasi, sistem pembayaran koin untuk episode premium, ketersediaan atau penghapusan judul favorit, dan berbagai aspek pengalaman membaca lainnya. Keberagaman topik ini menghasilkan variasi sentimen yang kaya dan kompleks dalam dataset ulasan, menjadikan *Webtoon* sebagai objek penelitian yang sangat menarik sekaligus menantang untuk penelitian klasifikasi sentimen berbasis *machine learning*[32].

2.3 Framework/Algoritma yang digunakan

2.3.1 CRISP-DM



Gambar 2. 1 Alur Proses CRISP-DM [33]

CRISP-DM (*Cross Industry Standard Process for Data Mining*) merupakan metodologi standar yang dikembangkan secara kolaboratif oleh sejumlah perusahaan terkemuka dan telah banyak diadopsi baik di industri maupun akademisi sebagai kerangka kerja dalam pelaksanaan proyek *data mining* dan *machine learning* secara sistematis dari awal hingga akhir. Metodologi ini menyediakan struktur proses yang bersifat iteratif, fleksibel, serta tidak bergantung pada *tools*, bahasa pemrograman, maupun domain industri tertentu, sehingga dapat diterapkan pada berbagai jenis proyek analitik data. Dengan demikian, seluruh aspek penting dalam proyek *data mining*, mulai dari pemahaman konteks hingga implementasi solusi, dapat direncanakan dan dijalankan secara terstruktur serta terdokumentasi dengan baik.

CRISP-DM terdiri dari enam tahapan utama yang saling berkaitan dan berjalan secara siklus. Tahap pertama adalah *Business Understanding*, yang bertujuan untuk memahami secara mendalam tujuan dan kebutuhan penelitian dari sudut pandang permasalahan. Tahap kedua adalah *Data Understanding*, yang meliputi pengumpulan data awal, eksplorasi data untuk memahami karakteristiknya, serta identifikasi permasalahan kualitas data. Tahap ketiga adalah *Data Preparation*, yang mencakup proses pengolahan data mentah menjadi dataset yang siap digunakan untuk pemodelan, termasuk pembersihan

data, seleksi fitur, dan transformasi data. Tahap keempat adalah *Modeling*, yang melibatkan penerapan algoritma *machine learning* yang sesuai dengan tujuan penelitian. Tahap kelima adalah *Evaluation*, yang bertujuan untuk menilai sejauh mana model yang dibangun mampu memenuhi tujuan penelitian berdasarkan metrik evaluasi yang digunakan. Tahap terakhir adalah *Deployment*, yang mencakup implementasi model terbaik ke dalam lingkungan nyata agar dapat dimanfaatkan oleh pengguna akhir.

Sifat iteratif dari *CRISP-DM* memungkinkan peneliti untuk kembali ke tahapan sebelumnya apabila diperlukan perbaikan pada tahap selanjutnya. Sebagai contoh, apabila hasil evaluasi menunjukkan performa model yang kurang optimal, maka peneliti dapat kembali ke tahap *Data Preparation* untuk memperbaiki proses *preprocessing*, atau ke tahap *Modeling* untuk mencoba algoritma maupun konfigurasi parameter yang berbeda [33]. Dalam penelitian ini, *CRISP-DM* digunakan sebagai metodologi utama yang memandu seluruh tahapan penelitian, mulai dari pengumpulan data ulasan Webtoon hingga implementasi aplikasi berbasis *Streamlit* ke dalam enam fase yang terstruktur.

2.3.2 Support Vector Machine (SVM)

Support *Support Vector Machine* (SVM) merupakan salah satu algoritma dalam *machine learning* yang termasuk ke dalam kategori *supervised learning* dan dapat digunakan untuk tugas klasifikasi maupun regresi. SVM bekerja dengan mencari *hyperplane* optimal yang mampu memisahkan data dari kelas-kelas yang berbeda dengan memaksimalkan jarak antar kelas (*margin*) di dalam ruang fitur multidimensi. *Hyperplane* merupakan bidang pemisah berupa fungsi linear yang membagi ruang fitur menjadi dua wilayah, masing-masing merepresentasikan satu kelas, yaitu kelas positif (+1) dan kelas negatif (−1).

Hyperplane optimal adalah *hyperplane* yang memiliki jarak (*margin*) maksimum terhadap titik data terdekat dari masing-masing kelas. Titik-titik

data yang berada paling dekat dengan *hyperplane* dan secara langsung menentukan posisi serta orientasinya disebut sebagai *support vectors*, yang menjadi dasar penamaan algoritma ini. Dengan memaksimalkan *margin*, SVM cenderung memiliki kemampuan generalisasi yang baik terhadap data yang belum pernah dilihat sebelumnya.

Secara matematis, untuk permasalahan klasifikasi biner dengan data latih {(x1,y1),...,(xn,yn)}, di mana yi∈{−1,+1}, SVM linear berusaha menemukan *hyperplane* yang dinyatakan sebagai:

$$w \cdot x + b = 0$$

Rumus 2. 1 Persamaan Hyperplane pada Support Vector Machine (SVM)

Keterangan:

a) w = vektor bobot (weight) yang tegak lurus terhadap hyperplane
b) x = vektor fitur data input
c) b = bias (intercept), yaitu jarak hyperplane dari titik origin

dengan tujuan memaksimalkan *margin* sebesar:

$$\frac{2}{|w|}$$

Rumus 2. 2 Margin pada Support Vector Machine (SVM)

Keterangan:

a) $|w|$ = norma (panjang) dari vektor bobot $|w|$
b) Margin merupakan jarak antara hyperplane pemisah dengan titik data terdekat (*support vector*) dari masing-masing kelas

dengan batasan:

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i$$

Rumus 2. 3 Kendala (Constraint) pada SVM

Keterangan:

a) y_i = label kelas dari data ke-i, bernilai +1 atau −1

b) x_i = vektor fitur dari data ke- i

c) w = vektor bobot hyperplane

d) b = bias hyperplane

e) $\forall i$ = berlaku untuk seluruh data ke- i dalam dataset

Prinsip dasar yang digunakan oleh SVM adalah *structural risk minimization*, yaitu pendekatan yang bertujuan untuk meminimalkan batas atas dari *generalization error*, bukan hanya meminimalkan *training error* seperti pada banyak algoritma *machine learning* lainnya. Pendekatan ini memberikan jaminan teoritis yang lebih baik terhadap kemampuan model dalam melakukan generalisasi.

Untuk data yang tidak dapat dipisahkan secara linear (*non-linearly separable*), SVM memanfaatkan konsep *kernel trick*, yaitu teknik untuk memetakan data dari ruang fitur asli ke ruang fitur berdimensi lebih tinggi sehingga pemisahan linear dapat dilakukan.

Beberapa fungsi kernel yang umum digunakan dalam SVM antara lain *kernel linear*, *polynomial*, *radial basis function* (RBF), dan *sigmoid*, dengan persamaan sebagai berikut:

a) Kernel Linear

$$K(x_i, z) = x_i^T z$$

Rumus 2. 4 Kernel Linear

Keterangan:

a) $K(x_i, z)$ = fungsi kernel yang mengukur kemiripan (similarity) antara dua vektor
b) x_i = vektor fitur dari data ke- i
c) z = vektor fitur dari data lain yang dibandingkan
d) x_i^T = transpos dari vektor x_i

b) Kernel Radial Basis Function (RBF)

$$K(x_i, z) = \exp(-\gamma|x_i - z|^2), \gamma > 0$$

Rumus 2. 5 Kernel Radial Basis Function (RBF)

Keterangan:

a) $K(x_i, z)$ = fungsi kernel RBF
b) x_i = vektor fitur dari data ke- i
c) z = vektor fitur dari data lain yang dibandingkan
d) $|x_i - z|^2$ = kuadrat jarak Euclidean antara x_i dan z
e) γ = parameter yang mengontrol pengaruh satu titik data terhadap titik lainnya; nilai γ gamma γ harus lebih besar dari 0

c) Kernel Polynomial

$$K(x_i, z) = (1 + x_i^T z)^d$$

Rumus 2. 6 Kernel Polynomial

Keterangan:

a) $K(x_i, z)$ = fungsi kernel polynomial
b) x_i = vektor fitur dari data ke- i
c) z = vektor fitur dari data lain yang dibandingkan

d) $x_i^T z$ = hasil dot product antara vektor x_i dan z

e) d = derajat (degree) polinomial yang menentukan kompleksitas fungsi pemisah

d) Kernel Sigmoid

$$K(x_i, z) = \tan \mathrm{h}(\gamma x_i^T z + r)$$

Rumus 2. 7 Kernel Sigmoid

Keterangan:

a) $K(x_i, z)$ = fungsi kernel sigmoid

b) x_i = vektor fitur dari data ke- i

c) z = vektor fitur dari data lain yang dibandingkan

d) γ = parameter skala (slope) fungsi sigmoid

e) r = konstanta koefisien (intercept) pada fungsi sigmoid

f) Tanh = fungsi tangen hiperbolik

Dalam penelitian ini, digunakan SVM dengan *kernel linear* karena representasi fitur menggunakan *Term Frequency-Inverse Document Frequency* (*TF-IDF*) menghasilkan ruang fitur berdimensi sangat tinggi yang umumnya bersifat *linearly separable*. Oleh karena itu, penggunaan *kernel linear* dinilai lebih efisien secara komputasi sekaligus tetap mampu memberikan performa yang optimal.SVM memiliki beberapa keunggulan yang menjadikannya efektif untuk tugas klasifikasi teks. Pertama, SVM sangat baik dalam menangani data berdimensi tinggi seperti representasi *TF-IDF* yang dapat menghasilkan ribuan fitur. Kedua, SVM relatif tidak rentan terhadap *curse of dimensionality* karena prinsip *margin maximization* yang digunakannya. Ketiga, SVM memiliki dasar teoritis yang kuat melalui *Vapnik–Chervonenkis (VC) theory* yang menjamin kemampuan generalisasi model. Keempat, SVM tetap efektif meskipun jumlah dimensi fitur lebih besar dibandingkan jumlah data latih.Namun demikian, SVM juga memiliki keterbatasan, yaitu kompleksitas komputasi yang relatif tinggi pada fase pelatihan, terutama untuk dataset berukuran sangat besar.

Meskipun demikian, untuk ukuran dataset dalam penelitian ini, SVM dengan *kernel linear* masih dapat dilatih dalam waktu yang relatif efisien[34].

2.3.3 Random Forest

Random Forest adalah algoritma *ensemble learning* yang membangun sejumlah besar pohon keputusan (*decision tree*) secara paralel dan independen satu sama lain selama fase *training*, kemudian menggabungkan prediksi dari seluruh pohon yang telah dibangun melalui mekanisme *majority voting* untuk menghasilkan prediksi akhir yang lebih akurat, stabil, dan *robust* dibandingkan prediksi dari pohon keputusan tunggal manapun. *Ensemble learning* adalah paradigma *machine learning* yang menggabungkan prediksi dari beberapa model yang lebih lemah (*weak learner*) untuk menghasilkan prediksi kolektif yang lebih kuat (*strong learner*). *Random Forest* mengimplementasikan konsep ini melalui metode *bagging* (*Bootstrap Aggregating*) yang membangun pohon-pohon yang bervariasi dan tidak berkorelasi tinggi satu sama lain.

Kunci keberhasilan *Random Forest* terletak pada dua sumber randomisasi yang diterapkan secara simultan dalam proses pembangunan setiap pohon. Randomisasi pertama adalah *bootstrap sampling*, di mana setiap pohon dilatih menggunakan subset data *training* yang dipilih secara acak dengan pengembalian (*sampling with replacement*) dari keseluruhan *dataset training*. Akibatnya, setiap pohon hanya melihat sekitar 63% dari data *training* yang unik (sisanya adalah duplikat akibat *sampling with replacement*), sehingga setiap pohon memiliki perspektif yang berbeda terhadap data. Data yang tidak terpilih dalam proses *bootstrap* (sekitar 37%) disebut sebagai *out-of-bag* (*OOB*) *samples* dan dapat digunakan untuk estimasi *error model* tanpa memerlukan *validation set* terpisah. Randomisasi kedua adalah *random feature selection*, di mana pada setiap *node* pemisahan dalam sebuah pohon, hanya sejumlah subset fitur yang dipilih secara acak yang dipertimbangkan sebagai kandidat untuk

pemisahan terbaik. Umumnya jumlah fitur yang dipertimbangkan diatur sebesar akar kuadrat dari total jumlah fitur untuk tugas klasifikasi.

Random Forest memiliki sejumlah keunggulan yang membuatnya menjadi salah satu algoritma *machine learning* paling populer dan sering digunakan dalam berbagai domain. Pertama, *Random Forest* sangat tahan terhadap *overfitting* karena mekanisme *averaging* yang secara efektif mereduksi *variance model* tanpa meningkatkan *bias* secara signifikan. Kedua, algoritma ini mampu menangani *dataset* dengan dimensi fitur yang sangat tinggi secara efisien. Ketiga, *Random Forest* menghasilkan metrik *feature importance* yang dapat digunakan untuk mengidentifikasi dan menginterpretasikan fitur-fitur yang paling berkontribusi pada keputusan klasifikasi. Keempat, *Random Forest* relatif tidak sensitif terhadap *hyperparameter* sehingga konfigurasi *default* sudah menghasilkan performa yang baik untuk sebagian besar kasus[35].

2.3.4 XGBoost

XGBoost (*eXtreme Gradient Boosting*) adalah algoritma *ensemble learning* berbasis *gradient boosting* yang membangun model secara sekuensial dan iteratif, di mana setiap model baru yang ditambahkan ke *ensemble* secara khusus dilatih untuk memperbaiki kesalahan prediksi dari kumpulan model yang telah dibangun sebelumnya. Berbeda dari *Random Forest* yang membangun pohon-pohon secara paralel dan independen, *XGBoost* membangun pohon-pohon keputusan secara berurutan (*sekuensial*), di mana setiap pohon baru difokuskan pada *residual error* dari iterasi sebelumnya. Pendekatan *boosting* ini secara bertahap dan terus-menerus memperbaiki kesalahan *ensemble* melalui serangkaian iterasi, menghasilkan model akhir yang merupakan akumulasi dari kontribusi banyak model lemah.

XGBoost memiliki beberapa inovasi teknis kunci yang membedakannya dari implementasi *gradient boosting* konvensional dan menjadikannya lebih

efisien dan akurat. Pertama, *XGBoost* menambahkan regularisasi *L1* (*Lasso*) dan *L2* (*Ridge*) secara eksplisit langsung pada fungsi objektif yang dioptimalkan selama *training*, sehingga secara aktif mengontrol kompleksitas model dan mencegah *overfitting*. Kedua, *XGBoost* menggunakan pendekatan *second-order Taylor expansion* untuk mengaproksimasi fungsi *loss*, yang menghasilkan konvergensi yang lebih cepat dan akurat dibandingkan *first-order gradient descent* yang digunakan oleh implementasi *boosting* tradisional. Ketiga, *XGBoost* mendukung pemrosesan paralel dalam pembangunan setiap pohon individual pada level pemilihan *split point* terbaik, sehingga waktu *training* jauh lebih efisien meskipun secara keseluruhan proses *boosting* tetap bersifat sekuensial. Keempat, *XGBoost* memiliki mekanisme *built-in* yang dapat menangani *missing values* secara otomatis dengan mempelajari arah *default* yang optimal untuk data yang hilang.

Prediksi akhir model *XGBoost* adalah jumlah dari prediksi seluruh pohon yang telah dibangun secara iteratif:

$$\widehat{y_i} = \sum f_t(x_i)$$

Rumus 2. 8 Persamaan Prediksi pada Model XGBoost

Keterangan:

a) $\widehat{y_i}$ = nilai prediksi akhir untuk data ke- i

b) $f_t(x_i)$ = hasil prediksi dari *tree* ke-t terhadap vektor fitur data ke- i (x_i)

c) $\sum t$ = penjumlahan hasil prediksi dari seluruh *tree* yang dibentuk secara bertahap (*boosting*) dalam model XGBoost

di mana ftf_tft adalah pohon ke-ttt dalam *ensemble*. Pada setiap iterasi ttt, sebuah pohon baru dibangun untuk meminimalkan fungsi objektif yang menggabungkan *training loss* (mengukur seberapa baik prediksi mendekati label sebenarnya) dan *regularization term* (mengontrol kompleksitas pohon). *XGBoost* secara konsisten mengungguli algoritma *machine learning* konvensional pada berbagai *benchmark dataset*, khususnya untuk data tabular

dan *sparse* seperti representasi *TF-IDF* dalam analisis teks, menjadikannya pilihan yang sangat kuat sebagai algoritma ketiga dalam penelitian ini yang mewakili paradigma *ensemble* berbasis *boosting*[36].

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

Term Frequency–Inverse Document Frequency atau yang disingkat *TF-IDF* adalah metode pembobotan fitur teks berbasis statistik yang digunakan untuk mengukur dan merepresentasikan kepentingan relatif suatu kata dalam sebuah dokumen terhadap keseluruhan koleksi dokumen atau yang dikenal sebagai *corpus*. *TF-IDF* menghasilkan representasi vektor numerik dari setiap dokumen teks yang dapat langsung digunakan sebagai input oleh algoritma *machine learning*, karena algoritma *machine learning* hanya dapat memproses data dalam bentuk numerik dan tidak dapat bekerja langsung dengan data teks mentah. *TF-IDF* merupakan salah satu metode representasi teks yang paling banyak digunakan dalam penelitian *information retrieval* dan *text classification* karena kesederhanaannya, efisiensinya secara komputasional, serta efektivitasnya yang telah terbukti secara empiris dalam berbagai tugas klasifikasi teks lintas domain dan bahasa.

TF-IDF terdiri dari dua komponen utama yang dikalikan untuk menghasilkan bobot akhir setiap kata dalam setiap dokumen. Komponen pertama adalah *Term Frequency* (*TF*) yang mengukur seberapa sering sebuah kata ttt muncul dalam dokumen ddd tertentu. Semakin tinggi frekuensi kemunculan kata dalam suatu dokumen, semakin tinggi nilai *TF*-nya, yang mengindikasikan bahwa kata tersebut menjadi topik yang penting dalam konteks dokumen tersebut. *Term Frequency* dihitung menggunakan rumus:

$$TF(t,d) = \frac{f(t,d)}{\sum f(t',d)}$$

Rumus 2. 9 Term Frequency (TF) pada TF-IDF

Keterangan:

a) $TF(t,d)$= nilai *Term Frequency* dari term (kata) *t* pada dokumen *d*

b) $f(t,d)$ = jumlah kemunculan term *t* pada dokumen *d*

c) $\sum f(t',d)$ = jumlah total kemunculan seluruh term t' pada dokumen *d* (total kata dalam dokumen tersebut)

di mana f(t,d) adalah frekuensi kemunculan kata t dalam dokumen d, dan Σ f(t',d) adalah jumlah total seluruh kata dalam dokumen d.

Komponen kedua adalah *Inverse Document Frequency* (*IDF*) yang mengukur seberapa jarang atau unik sebuah kata muncul di seluruh dokumen dalam *corpus*. Kata yang muncul di hampir semua dokumen, seperti kata-kata umum yang tidak informatif, dianggap memiliki nilai diskriminatif yang rendah sehingga mendapat bobot *IDF* yang rendah pula. Sebaliknya, kata yang jarang muncul di seluruh *corpus* tetapi sering muncul di dokumen tertentu dianggap sangat informatif dan khas untuk merepresentasikan dokumen tersebut, sehingga mendapat bobot *IDF* yang tinggi. *Inverse Document Frequency* dihitung menggunakan rumus:

$$IDF(t,D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

Rumus 2. 10 Inverse Document Frequency (IDF) pada TF-IDF

Keterangan:

a) $IDF(t,D)$ = nilai *Inverse Document Frequency* dari term *t* pada kumpulan dokumen *D*

b) *N* = jumlah total dokumen dalam korpus/kumpulan dokumen *D*

c) $|\{d \in D : t \in d\}|$ = jumlah dokumen dalam *D* yang mengandung term *t*

di mana N adalah jumlah total dokumen dalam corpus D, dan |{d ∈ D : t ∈ d}| adalah jumlah dokumen yang mengandung kata t. Nilai akhir TF-IDF diperoleh dari perkalian kedua komponen tersebut:

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

Rumus 2. 11 Pembobotan TF-IDF

Keterangan:

a) $TF\text{-}IDF(t,d,D)$ = nilai bobot akhir term *t* pada dokumen *d* dalam kumpulan dokumen *D*

b) $TF(t,d)$ = nilai *Term Frequency* term *t* pada dokumen *d* (dari Rumus 2.9)

c) $IDF(t,D)$ = nilai *Inverse Document Frequency* term *t* pada kumpulan dokumen *D* (dari Rumus 2.10)

Dalam implementasi praktis menggunakan *library scikit-learn*, seringkali digunakan varian *sublinear TF* dengan rumus TF = 1 + log(f(t,d)) untuk mengurangi dominasi token-token yang muncul sangat sering dalam satu dokumen sehingga tidak mendominasi representasi secara berlebihan. Dalam penelitian ini, *TF-IDF* diimplementasikan menggunakan kelas *TfidfVectorizer* dari *scikit-learn* dengan parameter *sublinear_tf=True* dan *max_features=5000*. *TF-IDF* dikombinasikan dengan berbagai konfigurasi *n-gram* yang berbeda sebagai variabel utama penelitian untuk menganalisis pengaruh representasi kontekstual terhadap performa klasifikasi sentimen secara komprehensif [10].

2.3.6 N-Gram

N-gram adalah model representasi teks berbasis statistik yang memecah teks menjadi urutan nnn unit berturut-turut, di mana setiap urutan nnn unit tersebut diperlakukan sebagai satu fitur tunggal dalam representasi vektor teks. Unit yang digunakan dalam *n-gram* dapat berupa karakter, suku kata, atau yang paling umum digunakan adalah kata (*word n-gram*). Model *n-gram* memungkinkan representasi teks untuk menangkap informasi kontekstual dan urutan kata yang tidak dapat ditangkap oleh model *unigram* yang hanya mempertimbangkan satu kata pada satu waktu tanpa memperhatikan kata-kata di sekitarnya. *N-gram* adalah salah satu metode representasi fitur teks yang paling fundamental dalam *NLP* karena kemampuannya menangkap dependensi lokal antara kata-kata yang berdekatan tanpa memerlukan komputasi yang sangat kompleks seperti yang diperlukan oleh model berbasis *neural network*.

Terdapat beberapa jenis konfigurasi *n-gram* yang umum digunakan dalam analisis teks, masing-masing dengan karakteristik dan keunggulan yang berbeda. *Unigram* (n=1n=1n=1) adalah model paling dasar yang mempertimbangkan setiap kata sebagai fitur yang sepenuhnya independen satu sama lain. *Bigram* (n=2n=2n=2) menangkap pasangan kata berurutan sehingga mampu merepresentasikan hubungan langsung antar dua kata yang berdampingan. *Trigram* (n=3n=3n=3) menangkap kelompok tiga kata berurutan sehingga mampu merepresentasikan konteks yang lebih luas namun dengan risiko *sparsity* yang lebih tinggi. Konfigurasi kombinasi seperti *unigram+bigram* (1,2)(1,2)(1,2) dan *unigram+bigram+trigram* (1,3)(1,3)(1,3) menggabungkan representasi dari *multiple level n-gram* secara simultan dalam satu vektor fitur yang lebih kaya dan komprehensif.

Keunggulan utama penggunaan *n-gram* dengan nnn lebih dari satu dalam analisis sentimen terletak pada kemampuannya menangkap pola negasi dan modifikasi yang sangat umum dalam bahasa natural. Sebagai contoh konkret pada data ulasan berbahasa Indonesia, frasa bernuansa negatif seperti *"tidak bagus"*, *"kurang memuaskan"*, *"sangat mengecewakan"*, atau *"sering error"* hanya dapat direpresentasikan secara utuh dan akurat melalui *bigram* atau *trigram*. Jika diproses sebagai *unigram* saja, komponen kata *"bagus"* dan *"memuaskan"* dalam frasa-frasa tersebut akan memberikan sinyal sentimen positif yang bertentangan dengan makna aktual frasa secara keseluruhan, sehingga model dapat salah mengklasifikasikan ulasan yang sebenarnya negatif. Di sisi lain, penggunaan *n-gram* dengan nilai nnn yang terlalu besar dapat menghasilkan representasi yang sangat *sparse* karena kombinasi kata yang muncul berulang semakin jarang seiring bertambahnya nilai nnn, sehingga terdapat *trade-off* antara kekayaan konteks dan *sparsity* yang perlu dioptimalkan melalui eksperimen.

Dalam implementasi menggunakan *scikit-learn*, konfigurasi *n-gram* diatur melalui parameter *ngram_range* pada *TfidfVectorizer*, di mana nilai

(min_n,max_n)(min_n, max_n)(min_n,max_n) menentukan rentang ukuran *n-gram* yang akan digunakan. Penggunaan (1,2)(1,2)(1,2) misalnya berarti semua *unigram* dan *bigram* akan digabungkan menjadi satu representasi fitur tunggal, bukan hanya *bigram* saja. Penelitian ini menguji lima konfigurasi *n-gram* yaitu (1,1), (2,2), (3,3), (1,2), dan (1,3) untuk menganalisis secara komprehensif bagaimana setiap konfigurasi memengaruhi performa model klasifikasi sentimen ulasan *Webtoon*[37].

2.3.7 Data Labeling

Data labeling atau pelabelan data adalah proses memberikan anotasi atau label kategori yang bermakna pada setiap sampel data mentah sehingga algoritma *machine learning supervised* dapat mempelajari hubungan antara pola dalam data input dengan kategori output yang diinginkan. Dalam konteks *sentiment analysis*, *data labeling* berarti memberikan label kelas sentimen pada setiap teks ulasan yang akan digunakan sebagai data *training* dan *testing* model. Label yang diberikan mencerminkan kategori sentimen yang terkandung dalam teks tersebut, misalnya positif, negatif, atau netral. Kualitas dan konsistensi label yang diberikan menjadi faktor penentu utama batas atas performa model yang dapat dicapai, karena model *machine learning* hanya dapat mempelajari pola yang tersedia dalam data berlabel yang disediakannya.

Terdapat dua pendekatan utama dalam *data labeling* untuk *sentiment analysis* yang masing-masing memiliki kelebihan dan kekurangan. Pendekatan pertama adalah *manual labeling*, di mana anotator manusia membaca setiap teks ulasan secara individual dan memberikan label berdasarkan pemahaman subjektif mereka tentang isi dan sentimen teks. Pendekatan ini menghasilkan label yang paling akurat dan dapat menangkap nuansa bahasa yang halus, namun sangat memakan waktu, tenaga, dan biaya terutama untuk *dataset* berskala besar seperti ribuan hingga jutaan ulasan. Pendekatan kedua adalah *heuristic-based labeling* atau pelabelan otomatis berdasarkan aturan heuristik

tertentu, di mana label ditentukan secara otomatis berdasarkan atribut yang sudah tersedia pada data tanpa intervensi manusia secara langsung.

Dalam penelitian ini, pendekatan *heuristic-based labeling* diterapkan menggunakan skor rating bintang yang diberikan pengguna sebagai dasar pemberian label sentimen. Pendekatan ini secara luas diterima dan banyak digunakan dalam komunitas penelitian *sentiment analysis* karena rating bintang merupakan ekspresi eksplisit dari sentimen dan kepuasan pengguna yang diberikan secara sukarela bersamaan dengan ulasan tekstual. Skema *labeling* yang diterapkan dalam penelitian ini adalah sebagai berikut: ulasan dengan rating 4 dan 5 bintang dikategorikan sebagai sentimen positif (label 1) karena mencerminkan kepuasan pengguna; ulasan dengan rating 1 dan 2 bintang dikategorikan sebagai sentimen negatif (label 0) karena mencerminkan ketidakpuasan pengguna; dan ulasan dengan rating 3 bintang dikeluarkan dari *dataset* karena rating ini bersifat ambigu dan dapat mencerminkan sentimen yang campuran atau netral sehingga sulit diklasifikasikan ke salah satu dari dua kategori dengan keyakinan yang cukup[38].

2.3.8 Evaluation Metrics

Evaluation metrics atau metrik evaluasi adalah ukuran-ukuran kuantitatif yang digunakan untuk menilai, mengukur, dan membandingkan performa model *machine learning* pada tugas klasifikasi secara objektif dan terstandarisasi. Pemilihan metrik evaluasi yang tepat sangat penting karena setiap metrik mengukur aspek yang berbeda dari performa model, dan tidak ada satu metrik pun yang secara universal optimal untuk semua skenario dan jenis *dataset*. Dalam konteks klasifikasi sentimen biner yang terdiri dari dua kelas (positif dan negatif), metrik evaluasi standar yang digunakan adalah *accuracy*, *precision*, *recall*, dan *F1-score*, yang semuanya diturunkan dari empat komponen dasar yang terdapat dalam *confusion matrix*.

Confusion matrix merupakan tabel ringkasan yang menampilkan jumlah prediksi benar dan salah yang dibuat oleh model untuk setiap kelas secara komprehensif, sehingga memberikan gambaran yang lebih mendetail tentang performa model per kelas dibandingkan hanya melihat satu angka *accuracy* keseluruhan. Empat komponen dasar *confusion matrix* didefinisikan sebagai berikut: *True Positive* (*TP*) adalah jumlah sampel positif yang berhasil diprediksi dengan benar sebagai positif; *True Negative* (*TN*) adalah jumlah sampel negatif yang berhasil diprediksi dengan benar sebagai negatif; *False Positive* (*FP*) adalah jumlah sampel negatif yang salah diprediksi sebagai positif, yang juga disebut sebagai *error* tipe I atau *false alarm*; dan *False Negative* (*FN*) adalah jumlah sampel positif yang salah diprediksi sebagai negatif, yang juga disebut sebagai *error* tipe II atau *miss*[39].

2.3.8.1 Accuracy

Accuracy adalah metrik evaluasi yang mengukur proporsi prediksi yang benar dari model terhadap keseluruhan jumlah prediksi yang dilakukan, tanpa mempertimbangkan distribusi kelas. *Accuracy* memberikan gambaran umum tentang seberapa sering model membuat prediksi yang benar secara keseluruhan. Rumus perhitungan *accuracy* adalah sebagai berikut:

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

Rumus 2. 12 Accuracy pada Evaluasi Model Klasifikasi

Keterangan:

a) TP(*True Positive*) = jumlah data positif yang diprediksi benar sebagai positif

b) TN(*True Negative*) = jumlah data negatif yang diprediksi benar sebagai negatif

c) FP(*False Positive*) = jumlah data negatif yang salah diprediksi sebagai positif

d) FN(*False Negative*) = jumlah data positif yang salah diprediksi sebagai negatif

Meskipun mudah diinterpretasikan, *accuracy* dapat menjadi metrik yang menyesatkan ketika distribusi kelas dalam *dataset* tidak seimbang. Pada *dataset* yang *imbalanced* seperti dalam penelitian ini di mana ulasan positif jauh lebih dominan dibandingkan ulasan negatif, model yang selalu memprediksi kelas mayoritas pun dapat mencapai nilai *accuracy* yang tinggi secara artifisial, meskipun sama sekali tidak mampu mengenali kelas minoritas dengan benar. Oleh karena itu, *accuracy* perlu selalu diinterpretasikan bersama dengan metrik-metrik lain yang lebih sensitif terhadap distribusi kelas[40].

2.3.8.2 Precision

Precision adalah metrik yang mengukur proporsi prediksi positif yang benar-benar merupakan data positif dari keseluruhan prediksi positif yang dibuat oleh model. *Precision* mencerminkan kemampuan model untuk menghindari *false alarm* atau *false positive*, yaitu kondisi di mana model salah mengidentifikasi sampel negatif sebagai positif. Rumus perhitungan *precision* adalah sebagai berikut:

$$\boldsymbol{Precision = \frac{TP}{TP + FP}}$$

Rumus 2. 13 Precision pada Evaluasi Model Klasifikasi

Keterangan:

a) TP(*True Positive*) = jumlah data positif yang diprediksi benar sebagai positif

b) FP(*False Positive*) = jumlah data negatif yang salah diprediksi sebagai positif

Nilai *precision* yang tinggi mengindikasikan bahwa ketika model memprediksi suatu sampel sebagai positif, prediksi tersebut sangat kemungkinan besar benar dan dapat dipercaya. *Precision* menjadi metrik yang sangat penting dalam skenario di mana biaya dari *false positive* sangat tinggi, misalnya ketika ulasan negatif yang salah dikategorikan sebagai positif dapat menyesatkan pengembang dalam mengambil keputusan terkait peningkatan kualitas aplikasi[40].

2.3.8.3 Recall

Recall atau yang juga dikenal sebagai *sensitivity* adalah metrik yang mengukur proporsi data positif aktual yang berhasil diidentifikasi dengan benar oleh model dari keseluruhan data positif yang sebenarnya ada dalam dataset. *Recall* mencerminkan kemampuan model untuk menemukan seluruh *instance* positif yang ada, dengan sesedikit mungkin *false negative*. Rumus perhitungan *recall* adalah sebagai berikut:

$$\boldsymbol{Recall = \frac{TP}{TP + FN}}$$

Rumus 2. 14 Recall pada Evaluasi Model Klasifikasi

Keterangan:

a) TP(*True Positive*) = jumlah data positif yang diprediksi benar sebagai positif
b) FN(*False Negative*) = jumlah data positif yang salah diprediksi sebagai negatif

Nilai *recall* yang tinggi menandakan bahwa model mampu mendeteksi sebagian besar data positif yang ada dalam dataset, dengan sangat sedikit kasus positif yang terlewat dan tidak terdeteksi. *Recall* dan *precision* memiliki hubungan *trade-off* yang inheren: meningkatkan nilai *recall* umumnya akan menurunkan nilai *precision* dan sebaliknya, karena meningkatkan *sensitivity* model cenderung meningkatkan jumlah *false*

positive. Oleh karena itu, diperlukan metrik yang mampu menyeimbangkan kepentingan kedua metrik ini secara bersamaan, yaitu *F1-score*[40].

2.3.8.4 F1-Score

F1-Score adalah metrik evaluasi yang merupakan rata-rata harmonik dari *precision* dan *recall*, dirancang khusus untuk memberikan satu nilai tunggal yang menyeimbangkan kepentingan kedua metrik tersebut secara simultan. Rata-rata harmonik dipilih karena lebih sensitif terhadap nilai ekstrem dibandingkan rata-rata aritmatika biasa, sehingga *F1-score* akan rendah jika salah satu dari *precision* atau *recall* rendah meskipun yang lainnya tinggi. Rumus perhitungan *F1-score* adalah sebagai berikut:

$$\boldsymbol{F1\text{-}Score} = \frac{\boldsymbol{2 \times (Precision \times Recall)}}{\boldsymbol{Precision + Recall}}$$

Rumus 2. 15 F1-Score pada Evaluasi Model Klasifikasi

Keterangan:

a) *Precision* = nilai *precision* model (dari Rumus 2.13)

b) *Recall* = nilai *recall* model (dari Rumus 2.14)

c) $F1\text{-}Score$ = nilai rata-rata harmonik (*harmonic mean*) antara precision dan recall, yang menyeimbangkan kedua metrik tersebut

F1-score sangat berguna dan relevan pada dataset yang memiliki distribusi kelas tidak seimbang seperti dalam penelitian ini, di mana distribusi ulasan positif jauh lebih dominan dibandingkan ulasan negatif. Dalam konteks evaluasi model dengan dataset tidak seimbang, *F1-score* dengan pendekatan *weighted average* yang memperhitungkan jumlah sampel per kelas sebagai bobot lebih representatif dibandingkan *macro average* karena memberikan kontribusi yang proporsional sesuai dengan

ukuran relatif setiap kelas terhadap nilai akhir[40]. Dalam penelitian ini, *F1-score weighted average* digunakan sebagai metrik utama untuk membandingkan dan menentukan model terbaik dari 15 kombinasi konfigurasi *n-gram* dan algoritma yang diuji.

2.3.9 IndoBERT

IndoBERT merupakan salah satu model *pre-trained language model* berbasis arsitektur *Bidirectional Encoder Representations from Transformers (BERT)* yang dikembangkan khusus untuk bahasa Indonesia. Model ini dilatih menggunakan korpus teks berbahasa Indonesia dalam jumlah besar sehingga mampu memahami konteks bahasa secara lebih mendalam dibandingkan metode tradisional berbasis *machine learning*. Arsitektur *BERT* sendiri menggunakan mekanisme *transformer encoder* yang memungkinkan model untuk memahami hubungan antar kata dalam suatu kalimat secara dua arah (*bidirectional*). Hal ini berbeda dengan model sebelumnya yang hanya membaca teks dari kiri ke kanan atau sebaliknya. Dengan pendekatan ini, *IndoBERT* mampu menangkap makna kontekstual, termasuk penggunaan slang, negasi, serta variasi bahasa informal yang sering muncul dalam ulasan pengguna.

Dalam penerapannya, *IndoBERT* dapat digunakan untuk berbagai tugas *Natural Language Processing (NLP)* seperti klasifikasi teks, analisis sentimen, *named entity recognition*, dan lain sebagainya. Untuk tugas klasifikasi sentimen, model ini biasanya dilakukan proses *fine-tuning* menggunakan dataset spesifik agar dapat menyesuaikan dengan konteks domain tertentu. Keunggulan utama *IndoBERT* terletak pada kemampuannya dalam memahami konteks kalimat yang kompleks serta sensitivitas terhadap struktur bahasa Indonesia. Hal ini menjadikan *IndoBERT* lebih unggul dibandingkan metode klasik seperti *Naive Bayes* atau *Support Vector Machine (SVM)* dalam banyak

kasus, terutama pada data yang bersifat tidak terstruktur dan mengandung variasi Bahasa[41].

Dalam penelitian ini, *IndoBERT* digunakan sebagai salah satu metode klasifikasi sentimen untuk membandingkan performanya dengan algoritma *machine learning* tradisional dalam menganalisis ulasan pengguna aplikasi Webtoon.

2.3.10 Cohen's Kappa

Cohen's Kappa merupakan salah satu metode statistik yang digunakan untuk mengukur tingkat kesepakatan (*inter-rater agreement*) antara dua annotator dalam proses pelabelan data kategorikal. Berbeda dengan pengukuran akurasi sederhana, *Cohen's Kappa* memperhitungkan kemungkinan kesepakatan yang terjadi secara kebetulan (*chance agreement*), sehingga memberikan hasil evaluasi yang lebih objektif dan reliabel. Nilai *Cohen's Kappa* dinyatakan dalam rentang -1 hingga 1, di mana nilai 1 menunjukkan kesepakatan sempurna, nilai 0 menunjukkan bahwa kesepakatan yang terjadi setara dengan peluang acak, dan nilai negatif menunjukkan adanya ketidaksepakatan yang lebih buruk dari peluang acak. Secara umum, interpretasi nilai *Cohen's Kappa* dapat dikategorikan sebagai berikut: nilai di bawah 0,20 menunjukkan kesepakatan yang sangat lemah, 0,21–0,40 lemah, 0,41–0,60 sedang, 0,61–0,80 kuat, dan di atas 0,80 menunjukkan kesepakatan yang sangat kuat.

Penggunaan *Cohen's Kappa* sangat penting dalam penelitian yang melibatkan pelabelan manual, seperti analisis sentimen, karena kualitas label sangat memengaruhi performa model yang dihasilkan. Dengan adanya pengukuran ini, peneliti dapat memastikan bahwa data yang digunakan memiliki tingkat konsistensi yang baik antar annotator, sehingga hasil penelitian menjadi lebih valid dan dapat dipercaya[42]. Dalam penelitian ini, *Cohen's Kappa* digunakan untuk mengukur tingkat kesepakatan antara dua

annotator dalam proses validasi label sentimen pada sebagian data, sehingga dapat memastikan bahwa dataset yang digunakan memiliki kualitas label yang reliabel sebelum digunakan pada tahap modeling.

2.4 Tools/software yang digunakan

2.4.1 Python

Python adalah bahasa pemrograman tingkat tinggi yang bersifat *interpreted*, dinamis, dan mendukung berbagai paradigma pemrograman termasuk berorientasi objek, fungsional, dan prosedural. *Python* pertama kali dikembangkan oleh *Guido van Rossum* dan telah berkembang menjadi bahasa pemrograman paling dominan dan banyak digunakan dalam ekosistem *data science*, *machine learning*, dan kecerdasan buatan secara global. Popularitas *Python* dalam bidang ini didorong oleh beberapa faktor utama yang saling mendukung, yaitu sintaks yang bersih, ekspresif, dan mudah dibaca sehingga mempercepat proses pengembangan kode, ekosistem *library* yang sangat kaya dan terus berkembang untuk berbagai keperluan komputasi numerik, visualisasi data, dan *machine learning*, komunitas pengembang yang sangat besar dan aktif yang terus mengembangkan solusi dan *library-library* baru, serta dokumentasi yang komprehensif dan mudah diakses untuk semua level pengguna.

Python menjadi pilihan standar dalam hampir seluruh proyek *data science* dan *machine learning* modern karena kemudahan integrasinya dengan berbagai *library* spesialis seperti *NumPy*, *Pandas*, *scikit-learn*, dan *TensorFlow* yang masing-masing menyediakan implementasi yang sangat dioptimalkan untuk tugas-tugas spesifik[43]. Dalam penelitian ini, *Python* digunakan sebagai bahasa pemrograman tunggal untuk seluruh *pipeline* penelitian, mulai dari pengumpulan data melalui *web scraping*, *preprocessing* teks, ekstraksi fitur *TF-IDF* dengan berbagai konfigurasi *n-gram*, *training* dan evaluasi seluruh 15 model eksperimen, hingga implementasi aplikasi web berbasis *Streamlit*.

Python versi 3.10 ke atas digunakan untuk memastikan kompatibilitas dengan seluruh *library* yang diperlukan dalam penelitian.

2.4.2 Google Colaboratory (Google Colab)

Google Colaboratory atau yang lebih dikenal sebagai *Google Colab* adalah platform komputasi berbasis *cloud* yang disediakan oleh *Google* secara gratis, yang memungkinkan pengguna untuk menulis, mengeksekusi, dan berbagi kode *Python* melalui antarmuka *Jupyter Notebook* langsung di *browser web* tanpa memerlukan instalasi perangkat lunak apa pun di komputer lokal pengguna. *Google Colab* menyediakan lingkungan *notebook* yang dihosting sepenuhnya di infrastruktur *cloud Google* dengan akses ke sumber daya komputasi seperti *CPU, GPU* (*Graphics Processing Unit*), dan *TPU* (*Tensor Processing Unit*) yang dapat digunakan secara gratis dengan batasan waktu dan sumber daya tertentu.

Google Colab menawarkan beberapa keunggulan signifikan yang menjadikannya pilihan ideal untuk penelitian *machine learning*. Pertama, aksesibilitas yang sangat tinggi karena dapat dijalankan dari *browser* mana pun pada perangkat apa pun tanpa instalasi lokal, memastikan konsistensi *environment* pengembangan. Kedua, akses ke *GPU* gratis yang dapat mempercepat proses komputasi secara signifikan untuk *training* model *machine learning*. Ketiga, integrasi langsung dengan *Google Drive* yang memudahkan penyimpanan dan pengaksesan dataset, model yang telah dilatih, serta hasil eksperimen secara persisten lintas sesi. Keempat, kemampuan berbagi *notebook* dengan mudah yang mendukung reprodusibilitas penelitian dan kolaborasi antar peneliti[44]. Dalam penelitian ini, *Google Colab* digunakan sebagai platform pengembangan dan eksperimen utama untuk seluruh proses, termasuk *preprocessing* 10.000 ulasan dan *training* 15 model dari kombinasi *n-gram* dan algoritma.

2.4.3 Scikit-learn

Scikit-learn adalah *library machine learning open-source* untuk *Python* yang menyediakan implementasi yang efisien, konsisten, dan terdokumentasi dengan sangat baik untuk ratusan algoritma *machine learning supervised* dan *unsupervised*, teknik *preprocessing* data, metode seleksi fitur, dan berbagai alat evaluasi model. *Scikit-learn* dibangun di atas tiga *library* komputasi saintifik *Python* yang fundamental, yaitu *NumPy* untuk komputasi array numerik multidimensi, *SciPy* untuk algoritma saintifik dan matematis, serta *Matplotlib* untuk visualisasi data. Keunggulan utama *scikit-learn* terletak pada *API* yang konsisten, intuitif, dan mudah digunakan, di mana semua *estimator* mengimplementasikan antarmuka yang seragam berupa metode *fit()* untuk *training*, *predict()* untuk prediksi, dan *transform()* untuk transformasi data, sehingga sangat mudah untuk mencoba, membandingkan, dan mengganti berbagai algoritma dalam *pipeline* yang sama[45].

Dalam penelitian ini, *scikit-learn* berperan sebagai *library* inti untuk beberapa komponen krusial. *TfidfVectorizer* digunakan untuk mengimplementasikan *TF-IDF* dengan berbagai konfigurasi *ngram_range* yang menjadi variabel utama penelitian. *SVC* (*Support Vector Classifier*) digunakan untuk implementasi algoritma *SVM* dengan *kernel* linear. *RandomForestClassifier* digunakan untuk implementasi algoritma *Random Forest* dengan 100 pohon. Fungsi *train_test_split* digunakan untuk pembagian data *training* dan *testing* dengan *stratified split*. Seluruh fungsi metrik evaluasi (*accuracy_score*, *precision_score*, *recall_score*, *f1_score*, *classification_report*, *confusion_matrix*) juga bersumber dari modul *sklearn.metrics*.

2.4.4 PySastrawi

PySastrawi adalah *library stemming* bahasa Indonesia untuk *Python* yang merupakan *port* dari *project Sastrawi*, sebuah *library stemming* berbahasa *Java* yang dikembangkan untuk menangani kompleksitas morfologi bahasa Indonesia. *PySastrawi* mengimplementasikan algoritma *Enhanced Confix Stripping* (*ECS*) yang dirancang khusus untuk menangani sistem pengimbuhan bahasa Indonesia yang kaya dan kompleks, mencakup berbagai jenis awalan (*prefiks*) seperti *me-*, *di-*, *ber-*, *ter-*, *ke-*, *se-*, berbagai jenis akhiran (*sufiks*) seperti *-kan*, *-an*, *-i*, *-nya*, kombinasi awalan dan akhiran secara bersamaan, serta *konfiks* (gabungan *prefiks* dan *sufiks* yang bekerja sebagai satu kesatuan) seperti *pe-an*, *per-an*, *ke-an*, dan *me-kan*.

Stemming bahasa Indonesia memerlukan pendekatan khusus yang sangat berbeda dari *stemming* bahasa Inggris atau bahasa Latin lainnya karena bahasa Indonesia memiliki sistem morfologi aglutinatif yang kaya, di mana sebuah kata dasar dapat menghasilkan puluhan variasi bentuk berimbuhan yang berbeda namun memiliki makna dasar yang sama. Sebagai contoh, kata dasar *"tulis"* dapat menghasilkan bentuk *"menulis"*, *"ditulis"*, *"menuliskan"*, *"dituliskan"*, *"penulisan"*, *"penulis"*, *"tulisan"*, *"tertulis"*, dan masih banyak lagi. Tanpa *stemming*, seluruh variasi morfologis ini akan direpresentasikan sebagai fitur yang berbeda dalam matriks *TF-IDF*, meningkatkan dimensi secara tidak perlu dan menurunkan efektivitas model[46]. Dengan *PySastrawi*, seluruh variasi tersebut direduksi ke bentuk dasar *"tulis"* yang tunggal, mengurangi *sparsity* dan meningkatkan kualitas representasi fitur dalam penelitian ini.

2.4.5 XGBoost Library

XGBoost Library adalah implementasi perangkat lunak *open-source* dari algoritma *XGBoost* yang tersedia sebagai *library Python* yang dapat diintegrasikan secara mulus dengan ekosistem *scikit-learn*. *Library* ini menyediakan antarmuka yang sepenuhnya kompatibel dengan *API scikit-learn* melalui kelas *XGBClassifier* untuk tugas klasifikasi dan *XGBRegressor* untuk tugas regresi, sehingga dapat digunakan dalam *pipeline scikit-learn* yang sama dengan algoritma-algoritma lainnya tanpa perlu penyesuaian kode yang signifikan. *XGBoost library* didesain dengan mempertimbangkan efisiensi komputasi secara mendalam, mendukung komputasi paralel pada level pemilihan *split point* dalam setiap pohon, komputasi terdistribusi di *kluster* komputer, serta pemrosesan data di luar memori untuk *dataset* yang sangat besar yang tidak dapat dimuat seluruhnya ke dalam *RAM*.

Parameter utama *XGBClassifier* yang dikonfigurasi dalam penelitian ini meliputi *n_estimators* yang menentukan jumlah pohon yang dibangun secara iteratif, *random_state* yang ditetapkan sebesar 42 untuk memastikan reprodusibilitas hasil eksperimen, serta *eval_metric* yang diatur ke *'logloss'* sebagai fungsi *loss* untuk tugas klasifikasi biner. Selain parameter-parameter tersebut, parameter *default XGBoost library* digunakan sebagai *baseline* untuk memastikan perbandingan yang adil dan konsisten antar seluruh 15 skenario eksperimen yang dilakukan dalam penelitian ini[47].

2.4.6 Pandas dan NumPy

Pandas adalah *library Python open-source* yang menyediakan struktur data yang fleksibel, efisien, dan *powerful* untuk manipulasi dan analisis data, terutama dalam bentuk tabel. Struktur data utama *Pandas* adalah *DataFrame*, yaitu tabel data dua dimensi dengan baris dan kolom yang masing-masing dapat

memiliki nama (*label*), serta *Series*, yaitu array satu dimensi berlabel yang merupakan kolom dari sebuah *DataFrame*. *Pandas* dirancang untuk memudahkan berbagai operasi manipulasi data tabular seperti *filtering* baris berdasarkan kondisi tertentu, *grouping* dan agregasi data, *merging* dan *joining* beberapa tabel, *reshaping* format data, serta penanganan *missing values* secara efisien dengan sintaks yang intuitif dan mudah dibaca.

NumPy (*Numerical Python*) adalah *library* fundamental untuk komputasi numerik dalam *Python* yang menyediakan objek array multidimensi (*ndarray*) beserta koleksi fungsi matematika dan statistik tingkat tinggi yang dioptimalkan untuk operasi pada array tersebut secara efisien menggunakan implementasi *C* yang dikompilasi. *NumPy* menjadi fondasi komputasi dari hampir seluruh ekosistem ilmiah *Python*, termasuk *Pandas*, *scikit-learn*, dan *Matplotlib*. Dalam penelitian ini, *Pandas* digunakan untuk *loading dataset CSV* dari hasil *scraping*, manipulasi dan *filtering* kolom teks dan label, serta penyimpanan hasil evaluasi seluruh eksperimen dalam format yang terstruktur. *NumPy* digunakan untuk operasi numerik dalam proses ekstraksi fitur, perhitungan metrik evaluasi, dan manipulasi array yang dihasilkan oleh *TfidfVectorizer*[48].

2.4.7 Google Play Scraper

Google Play Scraper adalah *library Python open-source* yang menyediakan antarmuka pemrograman yang mudah digunakan dan andal untuk mengakses serta mengekstraksi berbagai data dari *Google Play Store* secara terprogram dan otomatis, tanpa memerlukan *API key* resmi dari *Google* atau registrasi akun khusus. *Library* ini bekerja dengan mengimitasi perilaku *browser web* untuk mengakses *endpoint* internal *Google Play Store*, kemudian mengurai (*parse*) respons yang diterima dan mengembalikan data dalam format *Python native* (*dictionary* dan *list*) yang terstruktur dan siap diproses lebih lanjut.

Fungsi utama yang digunakan dalam penelitian ini adalah fungsi *reviews()* yang memungkinkan pengambilan ulasan pengguna dari sebuah aplikasi dalam jumlah besar sekaligus dengan berbagai parameter yang dapat dikonfigurasi. Parameter *app_id* digunakan untuk menentukan identitas unik aplikasi yang ulasannya ingin diambil, parameter *lang* digunakan untuk memfilter ulasan berdasarkan bahasa tertentu, parameter *country* digunakan untuk memfilter berdasarkan negara pengguna, parameter *sort* digunakan untuk menentukan metode pengurutan ulasan, dan parameter *count* digunakan untuk menentukan jumlah ulasan yang ingin dikumpulkan[49]. Dalam penelitian ini, *Google Play Scraper* digunakan untuk mengumpulkan 10.000 ulasan pengguna aplikasi *Webtoon* berbahasa Indonesia dari pengguna di Indonesia secara otomatis, menghasilkan *dataset* mentah yang kemudian disimpan dalam format *CSV* untuk diproses pada tahap selanjutnya.

2.4.8 Streamlit

Streamlit adalah *framework* pengembangan aplikasi web *open-source* berbasis *Python* yang dirancang secara khusus untuk membangun dan men-*deploy* aplikasi *data science* dan *machine learning* secara cepat, mudah, dan efisien tanpa memerlukan pengetahuan mendalam tentang teknologi pengembangan web *frontend* seperti *HTML, CSS, JavaScript*, atau *framework* web seperti *Flask* dan *Django*. *Streamlit* mengadopsi pendekatan yang sangat sederhana dan intuitif: skrip *Python* yang ditulis secara linear dari atas ke bawah dieksekusi oleh *Streamlit* untuk menghasilkan halaman web interaktif secara otomatis, di mana setiap perintah *Streamlit* (seperti *st.title()*, *st.text_input()*, *st.button()*, *st.dataframe()*, dan *st.pyplot()*) langsung menghasilkan komponen *UI* yang sesuai pada halaman web.

Streamlit menawarkan beberapa fitur kunci yang membuatnya sangat cocok untuk implementasi model *machine learning* dalam penelitian ini. Fitur *hot-reloading* otomatis memperbarui tampilan aplikasi secara *real-time* setiap kali

kode *Python* diubah dan disimpan, sehingga mempercepat siklus pengembangan dan iterasi. *Decorator @st.cache_resource* memungkinkan *caching* objek berat seperti model *machine learning* dan *TF-IDF vectorizer* yang dimuat dari file *pickle*, sehingga objek-objek tersebut hanya dimuat sekali saat aplikasi pertama kali dijalankan dan disimpan di *cache* untuk permintaan selanjutnya, mencegah *overhead loading* yang berulang. Dalam penelitian ini, *Streamlit* digunakan untuk mengimplementasikan model klasifikasi sentimen terbaik dalam aplikasi web interaktif yang memiliki tiga halaman utama: halaman beranda yang menampilkan informasi penelitian, halaman prediksi sentimen untuk input teks tunggal, dan halaman *dashboard* evaluasi yang menampilkan perbandingan performa seluruh 15 model secara visual dan interaktif[50].