BAB 2
LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian ini mengacu pada beberapa penelitian sebelumnya yang memiliki keterkaitan dengan analisis sentimen, penggunaan algoritma SVM, data dari media sosial, serta penanganan ketidakseimbangan kelas. Ringkasan penelitian terdahulu yang digunakan sebagai rujukan disajikan pada Tabel 2.1.

Tabel 2.1. Penelitian terdahulu

No	Penulis	Topik	Algoritma	Hasil
1	AminiMotlagh, dkk.	*A reliable sentiment analysis for classification of tweets in social networks*	KNN, DT, SVM, NB, *Ensemble*	Algoritma SVM menunjukkan hasil yang paling unggul dibandingkan algoritma lainnya dengan tingkat akurasi 86,42% pada pengujian polaritas dua kelas.
2	Cam, dkk.	*Sentiment analysis of financial Twitter posts on Twitter with the machine learning classifiers*	*Lexicon-based*, NB, LR, SVM, KNN, DT, MLP	SVM dan MLP memberikan performa terbaik, dengan SVM memimpin dengan tingkat akurasi mencapai 89% dan AUC 0,8729.

Tabel 2.1 Penelitian terdahulu (lanjutan)

No	Penulis	Judul	Topik	Hasil
3	Rohmatun & Baita	*Machine Learning-Based Sentiment Analysis on Twitter (X): A Case Study of the "Kabur Aja Dulu" Issue Using SVM*	SVM, SMOTE, TF-IDF, Grid Search	Model SVM dengan menggunakan kernel linear terbukti mampu mengklasifikasikan sentimen dengan sangat baik pada data cuitan berbahasa Indonesia, mencapai tingkat akurasi tertinggi sebesar 85,56%
4	Andriyani, dkk.	*The Effect of SMOTE Application on Support Vector Machine Performance in Sentiment Classification on Imbalanced Datasets*	SVM, SMOTE, TF-IDF	Penerapan metode SMOTE sukses mengatasi masalah data yang tidak seimbang dan melonjakkan akurasi SVM secara signifikan sebesar 12,48%, dari 71,41% menjadi 83,89%.

AminiMotlagh dkk. melakukan penelitian analisis sentimen terhadap 6.980 cuitan berbahasa Inggris di Twitter dengan topik COVID-19. Penelitian tersebut membandingkan beberapa algoritma klasifikasi, yaitu *K-Nearest Neighbor* (KNN), *Decision Tree* (DT), *Support Vector Machine* (SVM), *Naive Bayes* (NB), serta pendekatan *ensemble*. Hasil pengujian menunjukkan bahwa SVM mampu memberikan performa paling baik dibandingkan algoritma lainnya, dengan nilai

akurasi sebesar 86,42% pada klasifikasi dua polaritas [11]. Temuan tersebut relevan dengan penelitian ini karena sama-sama menggunakan data cuitan dan berfokus pada klasifikasi sentimen dua kelas.

Penelitian Cam dkk. membahas analisis sentimen pada 9.076 cuitan finansial berbahasa Turki di Twitter. Penelitian tersebut menggabungkan pendekatan berbasis leksikon dan *machine learning* untuk mendukung prediksi pergerakan pasar saham. Beberapa algoritma yang diuji meliputi NB, LR, SVM, KNN, DT, dan MLP. Berdasarkan hasil pengujian, SVM dan MLP memperoleh performa terbaik, dengan SVM mencapai akurasi sebesar 89% dan nilai *Area Under the Curve* (AUC) sebesar 0,8729 [12]. Penelitian tersebut menunjukkan bahwa SVM tetap kompetitif pada data Twitter meskipun digunakan pada domain yang berbeda.

Rohmatun dan Baita menerapkan SVM pada analisis sentimen cuitan berbahasa Indonesia mengenai isu "Kabur Aja Dulu" di platform Twitter (X). Data yang digunakan berjumlah 4.768 cuitan. Penelitian tersebut menggunakan TF-IDF sebagai metode ekstraksi fitur, SMOTE untuk menangani ketidakseimbangan kelas, serta *Grid Search* untuk mencari kombinasi parameter terbaik. Hasil pengujian menunjukkan bahwa SVM dengan *kernel linear* dan parameter C=10 memperoleh performa terbaik, dengan akurasi sebesar 85,56%, *precision* 85,19%, *recall* 85,56%, dan *F1-score* 85,30% [14]. Penelitian ini menjadi rujukan karena menggunakan kombinasi TF-IDF, SVM, SMOTE, dan *hyperparameter tuning* pada data dari platform X.

Andriyani dkk. meneliti pengaruh penggunaan SMOTE terhadap performa SVM pada klasifikasi sentimen dengan distribusi kelas tidak seimbang. Penelitian tersebut menunjukkan bahwa penerapan SMOTE dapat meningkatkan performa model, ditunjukkan melalui kenaikan akurasi dari 71,41% menjadi 83,89%. Selain itu, nilai *precision*, *recall*, dan *F1-score* juga meningkat setelah SMOTE diterapkan [13]. Penelitian tersebut menjadi dasar pertimbangan dalam penelitian ini untuk menguji pengaruh SMOTE pada beberapa skenario pembentukan model.

Berdasarkan beberapa penelitian tersebut, SVM dipilih karena memiliki performa yang baik dalam berbagai kasus klasifikasi sentimen, khususnya pada data media sosial. Selain itu, penggunaan TF-IDF, SMOTE, dan *hyperparameter tuning* juga menjadi pendekatan yang relevan untuk diuji pada penelitian ini. Perbedaan utama penelitian ini terletak pada topik yang dianalisis, yaitu sentimen terhadap *Agentic AI* dalam pengembangan perangkat lunak pada *platform* X, dengan perbandingan beberapa skenario pembagian data, penggunaan SMOTE,

serta metode optimasi parameter.

2.2 Analisis Sentimen

Analisis sentimen, atau yang juga dikenal sebagai *opinion mining*, merupakan salah satu bidang studi penting di dalam *Natural Language Processing* (NLP) yang dirancang secara khusus untuk mengekstraksi dan menganalisis sentimen serta pandangan dari suatu teks secara otomatis. Di era ledakan informasi saat ini, tujuan utama dari analisis sentimen adalah untuk menyediakan alat yang mampu memproses puluhan ribu opini atau komentar publik secara otomatis, cepat, efisien, dan berbiaya rendah. Melalui kemampuannya dalam menangkap pemikiran psikologis dari para pembuat komentar, teknologi ini memegang peranan krusial untuk membantu pengawasan opini publik serta memfasilitasi pengambilan keputusan yang lebih baik di berbagai industri [10].

2.3 X

X, yang sebelumnya dikenal luas sebagai *Twitter*, adalah *platform* media sosial berkonsep *microblogging* yang memfasilitasi interaksi dan komunikasi publik secara terbuka. *Platform* ini bertindak layaknya sebuah ruang publik yang demokratis bagi setiap individu untuk saling terhubung dengan mudah, memiliki suara, serta mengekspresikan pandangan terhadap berbagai macam isu. Berkat kemampuannya sebagai *platform* siaran yang cepat, X memungkinkan penggunanya untuk menyebarkan informasi dan terlibat dalam ragam diskusi berskala global secara instan atau *real-time* [15]. Karakteristik tersebut menjadikan X sebagai sumber data yang relevan untuk analisis sentimen karena opini pengguna terhadap suatu isu dapat diamati melalui cuitan yang bersifat terbuka, aktual, dan beragam.

2.4 Agentic AI

Kecerdasan Buatan (AI) dalam rekayasa perangkat lunak sebelumnya lebih banyak berperan sebagai asisten pasif yang beroperasi dalam skala terbatas, seperti fitur *autocomplete* untuk satu baris atau satu fungsi tertentu [1]. Namun, inovasi teknologi *Large Language Models* (LLMs) telah memicu disrupsi besar dengan lahirnya *Agentic AI*. Berbeda dengan generasi pendahulunya yang hanya merespons instruksi singkat secara tunggal, *Agentic AI* merupakan sistem dengan tingkat

otonomi tinggi yang mampu merencanakan, bernalar, dan mengeksekusi tugas pemrograman secara mandiri [16]. Disrupsi ini mengubah interaksi *developer* dengan AI, dari yang awalnya murni menulis kode secara manual dengan bantuan saran AI, kini beralih menjadi mendelegasikan pemecahan masalah kepada *Agentic AI* [1].

Kecanggihan utama dari *Agentic AI* terletak pada kemampuannya untuk melakukan penalaran multi-langkah dan berinteraksi langsung dengan alat-alat pengembangan eksternal [16]. Sistem pemrograman terdepan saat ini, seperti Claude Code, Codex, dan Cursor, tidak lagi sekadar menghasilkan cuplikan teks kode, tetapi mampu beroperasi pada skala keseluruhan repositori proyek. Alat-alat agen ini dibekali kapabilitas untuk memecah masalah yang rumit menjadi langkah-langkah sistematis, mengeksekusi perintah terminal, membaca dan memodifikasi berkas, hingga menjalankan pengujian aplikasi secara otomatis [1]. Kemampuan agen untuk mengevaluasi kegagalan kodenya sendiri dan melakukan perbaikan berulang tanpa panduan manusia secara terus-menerus inilah yang mewakili puncak disrupsi AI dalam praktik rekayasa perangkat lunak saat ini.

2.5 Cohen's Kappa

Cohen's Kappa merupakan metode yang digunakan untuk mengukur tingkat kesepakatan antara dua penilai atau dua hasil pelabelan pada data berbentuk kategori nominal. Berbeda dengan persentase kesesuaian biasa, *Cohen's Kappa* memperhitungkan kemungkinan kesepakatan yang terjadi secara kebetulan [17].

Secara umum, *Cohen's Kappa* dihitung menggunakan Persamaan 2.1.

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{2.1}$$

Keterangan:

- κ = nilai *Cohen's Kappa*,
- P_o = proporsi kesepakatan aktual atau *observed agreement*,
- P_e = proporsi kesepakatan yang terjadi karena peluang atau *chance agreement*.

Nilai P_o diperoleh dari jumlah data yang memiliki label sama dibagi dengan jumlah seluruh data. Sementara itu, P_e dihitung berdasarkan distribusi label

dari masing-masing penilai. Semakin tinggi nilai Kappa, semakin tinggi tingkat kesepakatan antara kedua penilai.

Interpretasi nilai *Cohen's Kappa* dalam penelitian ini mengacu pada kategori interpretasi yang ditunjukkan pada Tabel 2.2 [18].

Tabel 2.2. Interpretasi Nilai Cohen's Kappa

Nilai Kappa	**Tingkat Kesepakatan**
< 0.00	Buruk
$0.00 - 0.20$	Sangat rendah
$0.21 - 0.40$	Rendah
$0.41 - 0.60$	Sedang
$0.61 - 0.80$	Kuat
$0.81 - 1.00$	Hampir sempurna

Dalam penelitian ini, *Cohen's Kappa* digunakan untuk mengukur tingkat kesesuaian hasil pelabelan sentimen secara manual. Mengingat pentingnya kualitas data berlabel untuk menghasilkan model klasifikasi yang baik, proses pelabelan secara ideal mengharuskan keterlibatan minimal dua orang pakar (*raters* atau *annotators*) sehingga hasilnya dapat saling dibandingkan [18]. Keterlibatan dua penilai atau lebih ini umumnya memunculkan tantangan baru berupa kemungkinan adanya perbedaan objektivitas hasil pelabelan antar individu. Oleh karena itu, karena label sentimen yang digunakan berbentuk kategori nominal (positif dan negatif), metode statistik deskriptif *Cohen's Kappa* sesuai digunakan untuk mengatasi permasalahan tersebut dengan menilai reliabilitas serta tingkat kesepakatan (*agreement*) di antara kedua penilai [18].

2.6 Data Preprocessing

Dalam *Natural Language Processing* (NLP), data *preprocessing* adalah tahapan awal untuk membersihkan dan mentransformasi data teks mentah yang tidak terstruktur menjadi format yang lebih rapi dan seragam [11]. Tujuan utama dari proses ini adalah untuk menghilangkan informasi yang tidak relevan (*noise*) serta menyederhanakan bentuk kata, sehingga kualitas data meningkat dan model klasifikasi dapat bekerja secara lebih efisien dan akurat [19, 20]. Secara umum, tahapan preprocessing pada teks terdiri dari beberapa proses utama sebagai berikut:

1. Case Folding

Langkah ini bertujuan untuk menyeragamkan format teks dengan cara mengubah seluruh huruf kapital menjadi huruf kecil guna menjaga konsistensi dan mencegah ambiguitas data [20].

2. Normalization

   Tahapan untuk menormalkan kata-kata yang tidak baku, seperti ejaan yang salah, singkatan, atau bahasa gaul (slang), menjadi bentuk kata aslinya yang standar agar maknanya lebih mudah diinterpretasikan oleh algoritma [20].

3. Cleaning

   Proses ini berfokus pada penghapusan elemen-elemen yang tidak memiliki nilai informasi penting untuk proses analisis, seperti penghapusan tautan web (URL), tanda baca, angka, serta karakter khusus [19, 21].

4. Stopwords Removal

   Proses penyaringan dan penghapusan kata-kata bising yang memiliki frekuensi kemunculan tinggi namun tidak memberikan informasi atau makna sentimen yang signifikan (seperti kata depan atau kata hubung) guna mengurangi dimensi fitur [21].

5. *Lemmatization*

   Tahap ini digunakan untuk mengembalikan variasi bentuk kata ke dalam bentuk dasarnya. *Lemmatization* mengubah kata ke bentuk dasar kamusnya dengan mempertimbangkan konteks linguistik, seperti posisi kata dalam kalimat, sehingga variasi kata yang memiliki makna serupa dapat direpresentasikan secara lebih konsisten [19].

2.7 Data Splitting

Dalam pengembangan model *machine learning*, pembagian data (*data splitting*) merupakan metodologi esensial yang bertujuan untuk menjamin kualitas, keandalan, dan kemampuan generalisasi model prediksi. Secara mendasar, pemisahan dataset dilakukan untuk menciptakan batas yang jelas antara fase pembelajaran dan fase evaluasi. Data latih (*training data*) bertindak sebagai fondasi utama bagi algoritma untuk menganalisis dan mengenali pola fitur secara mendalam guna membangun fungsi matematis [22]. Sementara itu, data uji (*testing data*) diisolasi secara khusus untuk mengukur performa model secara objektif

saat dihadapkan pada sampel data baru yang belum pernah dikenali sebelumnya. Pemisahan yang ketat ini sangat krusial untuk mendeteksi serta mencegah terjadinya fenomena *overfitting*, yakni kondisi ketika model terbiasa menghafal detail data latih namun gagal melakukan prediksi akurat pada skenario nyata [12, 22].

2.8 TF-IDF

Dalam pemodelan *machine learning*, data teks harus ditransformasikan menjadi representasi numerik terlebih dahulu agar dapat diproses oleh algoritma. *Term Frequency-Inverse Document Frequency* (TF-IDF) adalah salah satu metode ekstraksi fitur yang paling andal untuk mengevaluasi tingkat kepentingan sebuah kata di dalam suatu dokumen spesifik relatif terhadap keseluruhan korpus data [23].

Secara konseptual, pembobotan TF-IDF didasarkan pada dua prinsip utama. Komponen pertama, yakni *Term Frequency* (TF), berfungsi untuk mengukur seberapa sering sebuah kata muncul di dalam satu dokumen tertentu. Sementara itu, komponen kedua, yakni *Inverse Document Frequency* (IDF), bertugas mengukur seberapa langka kata tersebut di seluruh kumpulan dokumen. Hasil perpaduan dari kedua metrik ini akan memberikan bobot tertinggi pada kata-kata yang sering muncul di suatu dokumen namun sangat jarang ditemukan di dokumen lainnya. Kata-kata unik dengan skor tinggi inilah yang nantinya dijadikan pedoman oleh model untuk membedakan kelas sentimen secara akurat.

2.9 SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) adalah sebuah metode prapemrosesan data yang dirancang khusus untuk mengatasi masalah ketidakseimbangan kelas (*class imbalance*) di dalam sebuah dataset, yang sering kali menyebabkan model klasifikasi menjadi bias terhadap kelas mayoritas dan mengabaikan kelas minoritas. Untuk menyeimbangkan kembali distribusi data, SMOTE tidak sekadar menduplikasi sampel yang sudah ada, melainkan beroperasi pada ruang fitur (*feature space*) untuk menghasilkan sampel-sampel sintetis baru yang ditujukan bagi kelas minoritas. Secara teknis, algoritma ini bekerja dengan cara mencari tetangga terdekat (*k-nearest neighbors*) dari sebuah titik data kelas minoritas, menghitung selisih jaraknya, lalu mengalikannya dengan angka acak untuk menyisipkan titik data buatan baru di sepanjang garis yang menghubungkan titik-titik tersebut. Melalui teknik penambahan data ini, proporsi kelas minoritas

akan menjadi setara dengan kelas mayoritas, sehingga model dapat mengenali pola secara lebih komprehensif, mengurangi bias, dan pada akhirnya meningkatkan performa serta akurasi prediksi secara keseluruhan [13].

2.10 SVM

Support Vector Machine (SVM) merupakan metode pembelajaran mesin yang digunakan untuk klasifikasi dengan mencari *hyperplane* optimal sebagai batas pemisah antarkelas. SVM bekerja berdasarkan prinsip margin maksimum, yaitu membentuk batas pemisah yang memiliki jarak sebesar mungkin terhadap data terdekat dari masing-masing kelas. Dengan demikian, SVM tidak hanya berusaha memisahkan data latih, tetapi juga membentuk model yang memiliki kemampuan generalisasi yang baik [24].

Pada klasifikasi dua kelas, data latih SVM dapat dinyatakan sebagai pasangan data dan label. Bentuk data latih SVM ditunjukkan pada Persamaan 2.2.

$$D = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \ldots, N\}, \quad y_i \in \{-1, +1\} \tag{2.2}$$

Keterangan:

- D adalah himpunan data latih.
- $\mathbf{x}_i$ adalah vektor fitur data ke-i.
- y_i adalah label kelas data ke-i.
- N adalah jumlah data latih.

Dalam penelitian ini, label $+1$ merepresentasikan sentimen positif, sedangkan label -1 merepresentasikan sentimen negatif. Data teks yang telah melalui proses ekstraksi fitur direpresentasikan dalam bentuk vektor numerik sehingga dapat diproses oleh SVM.

SVM membentuk fungsi keputusan untuk menentukan posisi data terhadap batas pemisah. Fungsi keputusan SVM ditunjukkan pada Persamaan 2.3.

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i + b \tag{2.3}$$

Keterangan:

- $f(\mathbf{x}_i)$ adalah nilai fungsi keputusan untuk data ke-i.
- $\mathbf{w}$ adalah vektor bobot.
- $\mathbf{x}_i$ adalah vektor fitur data ke-i.
- b adalah bias.

Batas pemisah atau *hyperplane* diperoleh ketika nilai fungsi keputusan sama dengan nol. Bentuk *hyperplane* SVM ditunjukkan pada Persamaan 2.4.

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.4}$$

Pada data nyata, pemisahan secara sempurna tidak selalu dapat dilakukan. Oleh karena itu, SVM menggunakan pendekatan *soft margin* yang memungkinkan sebagian data melanggar margin dengan penalti tertentu. Kondisi *soft margin* SVM ditunjukkan pada Persamaan 2.5 [24].

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{2.5}$$

Keterangan:

- ξ_i adalah *slack variable* pada data ke-i.
- $y_i(\mathbf{w}^T\mathbf{x}_i + b)$ adalah nilai yang menunjukkan posisi data terhadap margin.

Slack variable digunakan untuk menunjukkan tingkat pelanggaran margin. Jika data memenuhi margin, maka tidak terdapat pelanggaran. Sebaliknya, jika data tidak memenuhi margin, maka pelanggaran tersebut akan diperhitungkan melalui fungsi *loss*.

Pada implementasi SVM linear, salah satu bentuk *loss* yang dapat digunakan adalah *squared hinge loss*. Rumus *squared hinge loss* ditunjukkan pada Persamaan 2.6.

$$loss_i = \left[\max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)\right]^2 \tag{2.6}$$

Keterangan:

- $loss_i$ adalah nilai *loss* pada data ke-i.

- $\max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$ menunjukkan besar pelanggaran margin.

Fungsi *loss* tersebut bernilai nol apabila data telah memenuhi margin. Sebaliknya, apabila data berada di dalam margin atau berada pada sisi yang salah, maka fungsi *loss* akan menghasilkan nilai lebih besar dari nol.

Berdasarkan *squared hinge loss*, fungsi objektif SVM linear digunakan untuk menghitung nilai yang harus diminimalkan dalam proses pelatihan. Fungsi objektif SVM ditunjukkan pada Persamaan 2.7.

$$\min J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\left[\max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)\right]^2 \tag{2.7}$$

Keterangan:

- $J(\mathbf{w})$ adalah fungsi objektif yang diminimalkan.
- $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ adalah komponen regularisasi yang berkaitan dengan pengaturan margin.
- C adalah parameter regularisasi.
- $\sum_{i=1}^{N}\left[\max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\right]^2$ adalah total penalti pelanggaran margin pada data latih.

Parameter C berfungsi mengatur keseimbangan antara margin dan penalti terhadap pelanggaran margin. Nilai C yang besar memberikan penalti lebih besar terhadap pelanggaran, sedangkan nilai C yang kecil membuat model lebih toleran terhadap pelanggaran margin. Lebih lanjut, meskipun optimasi dapat dilakukan secara langsung pada fungsi *primal*, implementasi komputasi algoritma SVM sering kali mentransformasikan masalah ke dalam bentuk *dual* menggunakan metode pengali Lagrange (*Lagrange multiplier*). Transformasi dari *primal* ke *dual* ini dilakukan untuk menyederhanakan perhitungan matematis, sehingga model dapat mengidentifikasi *support vectors* dan batas keputusan (*decision boundary*) secara lebih presisi dan efisien.

2.11 Hyperparameter Tuning

Dalam pemelajaran mesin, kinerja dan akurasi sebuah model sangat bergantung pada parameter yang digunakan. Oleh karena itu, diperlukan

proses *hyperparameter tuning* untuk menemukan konfigurasi optimal dari sebuah model [25, 26]. Terdapat berbagai metode yang dapat digunakan untuk melakukan optimasi ini, di antaranya adalah *grid search*, *random search*, *Bayesian optimization*, *particle swarm optimization*, hingga algoritma genetika [25]. Di antara berbagai metode tersebut, *Grid Search* dan *Bayesian Optimization* merupakan dua teknik yang diigunakan dalam penelitian ini.

A Grid Search Cross-Validation

Grid Search adalah teknik optimasi yang bekerja dengan cara menguji seluruh kombinasi model dan *hyperparameter* yang telah ditentukan di dalam sebuah kisi (*grid*) secara satu per satu. Tujuan utama dari metode ini adalah untuk mengevaluasi setiap kombinasi guna menentukan konfigurasi mana yang menghasilkan performa model terbaik untuk digunakan dalam prediksi. Dalam praktiknya, teknik ini umumnya digabungkan dengan validasi silang (*k-fold cross-validation*), sehingga setiap kombinasi parameter dievaluasi secara iteratif menggunakan lipatan data untuk menciptakan indeks performa yang valid dan menghindari *overfitting* [25].

B Bayesian Optimization

Bayesian Optimization merupakan metode optimasi yang memanfaatkan prinsip inferensi Bayesian untuk membangun model probabilistik dari suatu fungsi objektif. Berbeda dengan *grid search* yang menguji seluruh kombinasi parameter secara menyeluruh, metode ini bekerja dengan mempertimbangkan informasi dari hasil pengujian sebelumnya sehingga pencarian dapat dilakukan secara lebih terarah. Pendekatan probabilistik semacam ini membuat proses pencarian *hyperparameter* terbaik menjadi lebih efisien dan berkontribusi pada peningkatan performa model klasifikasi [26].

Cara kerja *Bayesian Optimization* pada dasarnya digerakkan oleh dua komponen matematis, yaitu model pengganti (*surrogate model*) dan fungsi akuisisi (*acquisition function*) [27].

1. Model Pengganti (*Surrogate Model*)

   Proses pencarian *hyperparameter* (seperti nilai *C* dan jenis *kernel* pada SVM) sesungguhnya membutuhkan komputasi pelatihan model yang cukup mahal dan memakan waktu. Untuk itu, *Bayesian Optimization* menggunakan

surrogate model, yang umumnya berbasis *Gaussian Process* (GP), sebagai representasi fungsi objektif yang jauh lebih ringan untuk dihitung. Model ini kemudian diperbarui secara berkelanjutan sehingga terbentuk distribusi *posterior* setiap kali terdapat data hasil evaluasi *hyperparameter* baru yang diobservasi.

2. Fungsi Akuisisi (*Acquisition Function*)

   Berdasarkan distribusi *posterior* yang dihasilkan oleh *surrogate model*, fungsi akuisisi digunakan untuk memberikan skor pada tiap titik *hyperparameter* sekaligus menentukan titik mana yang paling potensial untuk dievaluasi pada iterasi berikutnya. Fungsi ini bekerja dengan menyeimbangkan dua aspek yang saling bertolak belakang, yaitu eksploitasi, yakni memilih area *hyperparameter* yang dari data sebelumnya sudah terbukti memberikan akurasi tinggi, dan eksplorasi, yakni memilih area yang belum banyak diuji atau masih memiliki tingkat ketidakpastian tinggi. Beberapa metrik fungsi akuisisi yang umum digunakan antara lain *Expected Improvement* (EI), *Probability of Improvement* (PI), dan *Upper Confidence Bound* (UCB).

Secara keseluruhan, siklus pencarian *hyperparameter* pada *Bayesian Optimization* berlangsung secara iteratif. Algoritma mencari kombinasi *hyperparameter* yang memaksimalkan fungsi akuisisi, kemudian melakukan pelatihan dan evaluasi model klasifikasi yang sesungguhnya pada parameter tersebut, lalu hasilnya digunakan untuk memperbarui *surrogate model*. Siklus ini terus berulang hingga kriteria penghentian atau batas iterasi maksimum tercapai, sehingga proses optimasi menjadi jauh lebih terarah dan efisien dibandingkan metode tradisional [27, 26].

2.12 Cross Validation

Cross-validation (validasi silang) adalah metode evaluasi yang banyak digunakan dalam pembelajaran mesin untuk mengukur seberapa baik kinerja sebuah model ketika dihadapkan pada data yang belum pernah dilihat oleh model saat pelatihan. Secara umum, metode *k-fold cross-validation* bekerja dengan membagi dataset menjadi *k* bagian yang sama besar, kemudian proses pelatihan dan pengujian diulang sebanyak *k* kali. Pendekatan ini memberikan kesempatan yang sama bagi setiap data untuk digunakan sebagai data uji. Kinerja akhir dari model kemudian

dievaluasi berdasarkan nilai rata-rata metrik akurasi dari keseluruhan *k* pengujian tersebut [28].

Salah satu variasi dari metode ini adalah *Stratified K-Fold Cross-Validation* (SKCV). Berbeda dengan metode *k-fold* biasa yang membagi dataset secara acak, SKCV menggunakan teknik pengambilan sampel bertingkat (*stratified sampling*) untuk memastikan bahwa rasio distribusi sampel antar kelas tetap dipertahankan secara merata di setiap *fold*. Metode ini lebih sesuai digunakan pada kasus klasifikasi dengan distribusi kelas yang tidak seimbang (*imbalanced dataset*) karena proporsi masing-masing kelas tetap terjaga pada setiap proses pelatihan dan validasi. Dengan demikian, SKCV dapat membantu menghasilkan evaluasi model yang lebih stabil dan representatif [28].

2.13 Confusion Matrix

Confusion matrix adalah sebuah alat evaluasi kinerja yang digunakan untuk mengukur seberapa baik sebuah model klasifikasi dalam memprediksi data. Evaluasi ini dilakukan dengan cara membandingkan nilai prediksi yang dihasilkan oleh model terhadap nilai aktual (sebenarnya) pada *dataset*. Matriks ini menyajikan informasi mendetail terkait keputusan klasifikasi yang benar maupun yang keliru [28]. Secara umum, struktur *confusion matrix* untuk klasifikasi dua kelas (seperti sentimen positif dan negatif) dibangun berdasarkan empat komponen dasar, yaitu:

- *True Positive* (TP): Jumlah data dari kelas positif yang berhasil diprediksi secara tepat sebagai kelas positif oleh model.
- *True Negative* (TN): Jumlah data dari kelas negatif yang berhasil diprediksi secara tepat sebagai kelas negatif oleh model.
- *False Positive* (FP): Jumlah data kesalahan prediksi, yaitu data dari kelas aktual negatif secara keliru diprediksi oleh model sebagai kelas positif.
- *False Negative* (FN): Jumlah data kesalahan prediksi, yaitu data dari kelas aktual positif secara keliru diprediksi oleh model sebagai kelas negatif.

Dari keempat komponen *confusion matrix* tersebut, berbagai metrik evaluasi dapat dihitung untuk menilai performa dari model klasifikasi. Rincian metrik evaluasi tersebut adalah sebagai berikut:

1. *Accuracy*: Merupakan persentase jumlah data yang diprediksi secara benar oleh model dari keseluruhan data uji. Persamaan matematis untuk menghitung nilai akurasi ditunjukkan pada Persamaan 2.8.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.8}$$

2. *Precision*: Mengukur tingkat ketepatan model dalam memprediksi suatu kelas, yaitu proporsi data yang benar diprediksi pada kelas tersebut dibandingkan dengan seluruh data yang diprediksi sebagai kelas tersebut oleh model. Perhitungan nilai *precision* dirumuskan pada Persamaan 2.9.

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

3. *Recall*: Mengukur kemampuan model dalam mengenali seluruh data yang secara aktual termasuk ke dalam suatu kelas, yaitu proporsi data yang benar diprediksi pada kelas tersebut dibandingkan dengan seluruh data aktual pada kelas tersebut. Perhitungan nilai *recall* dapat dilihat pada Persamaan 2.10.

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

4. *F1-Score*: Merupakan nilai rata-rata harmonik (*harmonic mean*) dari nilai *precision* dan *recall*. Metrik ini merangkum kedua nilai tersebut untuk memberikan keseimbangan evaluasi, terutama pada *dataset* yang distribusinya tidak seimbang. Formulasi perhitungan *F1-Score* ditunjukkan pada Persamaan 2.11.

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.11}$$