

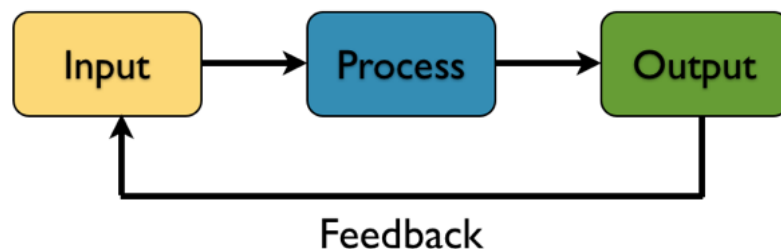
BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. Sistem

Sistem pada dasarnya merupakan sebuah entitas konseptual maupun fisik yang terdiri dari sekumpulan komponen, elemen, atau subsistem yang saling berinteraksi, bergantung, dan terintegrasi untuk mencapai suatu tujuan atau sasaran tertentu secara terpadu (Afriansyah & Syaripudin, 2022). Dalam lingkup ilmu komputer, sebuah sistem tidak dapat berdiri sendiri melainkan dibatasi oleh ruang lingkup (boundary) yang memisahkannya dari lingkungan luar, serta memiliki mekanisme umpan balik (feedback) untuk menjaga stabilitas kinerjanya. Keterikatan antarkomponen ini memastikan bahwa input yang diterima dapat diproses secara logis untuk menghasilkan luaran (output) yang memiliki nilai guna (Chotijah & Rosyid, 2021).



Gambar 2. 1 Ilustrasi Komponen dan Karakteristik Sistem

Karakteristik fundamental dari sebuah sistem yang baik meliputi adanya komponen pengolah, antarmuka penghubung (*interface*), serta

prosedur operasional yang jelas dalam mengatur aliran data. Tanpa adanya sinkronisasi yang kohesif antar komponen tersebut, sebuah sistem akan mengalami entropi atau penurunan kinerja yang berujung pada kegagalan operasional (*system failure*). Oleh karena itu, pendekatan pemikiran sistem (*system thinking*) sangat diperlukan dalam memandang sebuah masalah teknologi bukan sebagai elemen yang terpisah, melainkan sebagai satu kesatuan arsitektur yang holistic (Bakri & Nasution, 2024).

Dalam konteks penelitian rekayasa perangkat lunak ini, sistem diartikulasikan sebagai sebuah platform sistem informasi berbasis web terpusat. Sistem klasifikasi ini dirancang untuk mengakomodasi integrasi antara pangkalan data, antarmuka pengguna, dan mesin analitik kecerdasan buatan, sehingga alur pemrosesan opini publik dapat berjalan secara terotomatisasi, akurat, dan memiliki skalabilitas yang tinggi.

2. Rancang Bangun

Rancang bangun merupakan terminologi yang meleburkan dua fase esensial dalam daur hidup pengembangan perangkat lunak, yakni fase perancangan (*design*) dan fase pembangunan (*construction* atau *development*). Fase perancangan berfokus pada spesifikasi teknis, pemodelan arsitektur data, serta desain antarmuka logis untuk memberikan cetak biru visual mengenai bagaimana sebuah perangkat lunak seharusnya bekerja memenuhi kebutuhan pengguna (Miswadi et al., 2020). Di sisi lain, fase pembangunan merupakan proses penerjemahan cetak biru tersebut ke dalam

bahasa pemrograman komputasi, pembuatan skema basis data fisik, hingga pengujian fungsionalitas kode (Ruziq & Wayahdi, 2026).

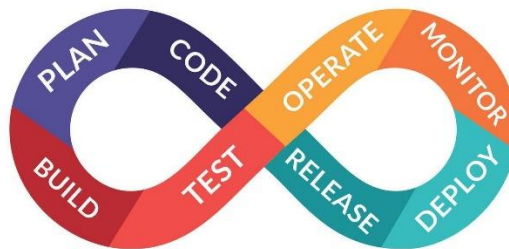
Secara metodologis, integrasi antara proses rancang dan bangun memastikan bahwa tidak ada kesenjangan (*gap*) antara kebutuhan analitis bisnis dengan produk akhir yang direalisasikan oleh pengembang. Dokumentasi rancangan yang matang, seperti melalui penggunaan *Unified Modeling Language* (UML), berfungsi krusial dalam menekan angka kesalahan logika (bug) sebelum fase penulisan kode program benar-benar dieksekusi. Hal ini terbukti secara empiris mampu meningkatkan efisiensi waktu, tenaga, dan biaya dalam rekayasa perangkat lunak.

Penerapan konsep rancang bangun pada penelitian ini merujuk pada konstruksi sistem klasifikasi opini yang diinisiasi dari pemodelan alur kerja sistem, desain arsitektur *Model-View-Template* (MVT) pada framework Django, hingga proses penulisan kode backend berbasis Python. Sinkronisasi kedua tahapan ini menjamin platform yang dihasilkan memiliki struktur yang kokoh dan beroperasi selaras dengan spesifikasi fungsional yang telah ditetapkan.

3. Metode Pengembangan Agile

Metode pengembangan *Agile* merupakan pendekatan dalam rekayasa perangkat lunak yang mengedepankan fleksibilitas, proses iteratif (perulangan), dan kolaborasi yang dinamis selama daur hidup pengembangan sistem. Berbeda dengan pendekatan tradisional yang kaku dan linear (seperti *Waterfall*), *Agile* memungkinkan pengembang untuk beradaptasi dengan

cepat terhadap perubahan kebutuhan atau penyesuaian spesifikasi tanpa harus merombak seluruh arsitektur dasar yang telah dibangun (Miswadi et al., 2020). Pendekatan iteratif ini terbukti secara empiris sangat ideal untuk diimplementasikan pada proyek pengembangan perangkat lunak berbasis kecerdasan buatan, di mana penyetelan model parameter algoritma dan penyesuaian antarmuka aplikasi seringkali membutuhkan pengujian dan umpan balik yang terus-menerus sebelum sistem dinyatakan stabil secara komputasional. Dalam penelitian ini, metode Agile yang diimplementasikan secara spesifik adalah pendekatan *Extreme Programming* (XP) yang berfokus pada kecepatan iterasi penulisan kode dan pengujian pada pengembangan sistem mandiri.



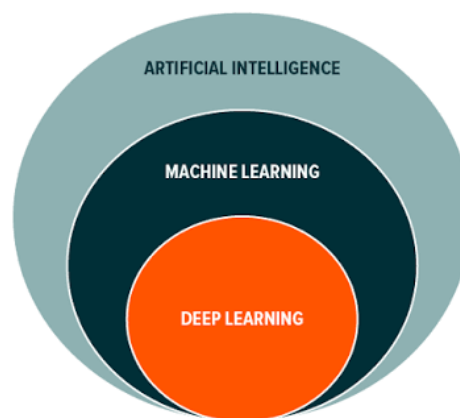
Gambar 2. 2 Ilustrasi Siklus Pengembangan Agile

Berdasarkan ilustrasi siklus pengembangan di atas, metode *Agile* beroperasi melalui rangkaian tahapan yang berkesinambungan dan tidak terputus. Siklus diawali dari tahap Perencanaan (*Plan*) untuk mengidentifikasi batasan masalah, disusul oleh tahap Perancangan (*Design*) guna memodelkan arsitektur basis data dan alur sistem. Tahap Pengembangan (*Develop*) bertindak sebagai fase penulisan kode program inti, yang

senantiasa berjalan secara paralel dengan tahap Pengujian (*Test*) untuk memvalidasi fungsi antarmuka maupun akurasi metrik algoritma. Apabila seluruh pengujian telah memenuhi standar kelayakan, sistem akan dieksekusi menuju tahap Peluncuran (*Release*), yang bermuara pada tahap Umpan Balik (*Feedback*) guna mengumpulkan hasil evaluasi operasional sebagai landasan perbaikan pada iterasi selanjutnya (Burhani et al., 2025).

4. *Artificial Intelligence (AI)*

Artificial Intelligence (AI) atau Kecerdasan Buatan merupakan cabang keilmuan komputer yang berfokus pada penciptaan mesin atau agen komputasi yang memiliki kemampuan untuk meniru fungsi kognitif dan kecerdasan rasional manusia, seperti kemampuan belajar, penalaran, dan koreksi diri (Jati Jatmika & Amalia, 2024). Tujuan utama dari disiplin ilmu ini adalah merancang algoritma matematis yang memungkinkan perangkat lunak untuk menyelesaikan masalah-masalah kompleks secara mandiri berdasarkan pengenalan pola data, tanpa harus diprogram secara eksplisit dengan aturan percabangan yang statis (Muhammad et al., 2023).



Gambar 2. 3 Ilustrasi Hierarki Konsep *Artificial Intelligence*

Berdasarkan ilustrasi hierarki pada gambar di atas, konsep kecerdasan komputasional terbagi ke dalam tiga lapisan (*layer*) utama yang saling beririsan, namun memiliki pendekatan matematis yang berbeda. Penjabaran dari masing-masing lapisan tersebut adalah sebagai berikut.

a. *Artificial Intelligence (AI)*

Lingkaran terluar yang merupakan payung besar dari seluruh teknologi pendelegasian kecerdasan. Konsep ini mencakup segala metode yang memungkinkan komputer meniru kecerdasan manusia, termasuk sistem pakar (*expert systems*) klasik yang murni beroperasi berdasarkan logika *if-then* (berbasis aturan) yang dikodekan secara manual oleh manusia.

b. *Machine Learning (ML)*

Lingkaran menengah yang merupakan sub-bidang utama dari AI. Pada tahap ini, mesin tidak lagi mengandalkan aturan statis, melainkan dilatih menggunakan metode statistik terhadap korpus data historis berskala besar. Mesin dibiarkan "belajar" (*learning*) untuk mengenali probabilitas, mengekstrak fitur klasifikasi, dan membangun model prediktif secara mandiri (Wibowo, 2023).

c. *Deep Learning (DL)*

Lingkaran terdalam yang merupakan evolusi spesifik dari Machine Learning. DL memecahkan masalah komputasi yang jauh lebih rumit, seperti pengenalan citra visual atau sintesis suara, dengan menggunakan arsitektur algoritma *Artificial Neural Networks* (Jaringan

Saraf Tiruan) berlapis-lapis yang meniru struktur kerja neuron otak biologis manusia.

Dalam konteks rancang bangun sistem klasifikasi opini publik ini, teknologi kecerdasan buatan yang diimplementasikan secara spesifik difokuskan pada lapisan *Machine Learning*, yakni melalui eksekusi algoritma *Support Vector Machine* (SVM). Algoritma tersebut dipilih karena efisiensi statistiknya yang tinggi dalam mengekstraksi matriks fitur dari data teks berdimensi besar, membedah karakteristik linguistik warganet, dan memprediksi orientasi polaritas sentimen dengan tingkat generalisasi yang jauh lebih tangguh dibandingkan metode AI berbasis aturan tradisional.

5. Analisis Sentimen

Analisis sentimen merupakan salah satu cabang komputasional dari disiplin ilmu penambangan teks (*text mining*) dan pemrosesan bahasa alami (*Natural Language Processing*) yang berfokus pada proses ekstraksi, identifikasi, dan klasifikasi opini atau polaritas emosi dari sebuah dokumen teks yang tidak terstruktur. Secara operasional, metode ini bekerja dengan cara membedah pola bahasa alami untuk mendeteksi kecenderungan persepsi subjektif penulisnya, yang kemudian dikategorikan ke dalam kelas polaritas secara terukur, seperti sentimen positif, negatif, maupun netral.

Dalam pengolahan data digital berskala masif, analisis sentimen bertindak sebagai instrumen teknis esensial yang mampu mentransformasikan data linguistik mentah menjadi informasi kuantitatif secara objektif, sehingga hasil analisis dapat terhindar dari bias yang sering

terjadi pada pemrosesan manual (Pratiwi, 2025). Implementasi metode ini sangat relevan untuk memproses opini masyarakat pada platform media sosial yang memiliki gaya bahasa dinamis, seperti komentar YouTube terkait peristiwa global, di mana sistem mampu mengidentifikasi dan memetakan dinamika persepsi warganet menjadi wawasan yang terstruktur secara efisien dan presisi (Kamsar & Khaer, 2024). Secara arsitektural, tingkat kedalaman komputasi analisis sentimen diklasifikasikan ke dalam tiga jenis tingkatan utama.

- a. Analisis Sentimen Tingkat Dokumen (*Document-Level Sentiment Analysis*): Pendekatan dasar yang bertujuan mengevaluasi dan mengklasifikasikan polaritas opini dari satu dokumen teks secara keseluruhan. Metode ini mengasumsikan bahwa keseluruhan dokumen tersebut hanya mengekspresikan opini pada satu entitas tunggal atau topik utama.
- b. Analisis Sentimen Tingkat Kalimat (*Sentence-Level Sentiment Analysis*). Pendekatan menengah yang bekerja dengan mengisolasi dan mengevaluasi setiap kalimat secara individual. Sistem akan menentukan terlebih dahulu apakah kalimat tersebut bersifat objektif (hanya menyajikan fakta informatif) atau subjektif (menyajikan opini). Kalimat subjektif kemudian dievaluasi untuk menentukan polaritas sentimennya.
- c. Analisis Sentimen Berbasis Aspek (*Aspect-Based Sentiment Analysis*). Pendekatan tingkat lanjut dan paling granular. Metode ini dirancang untuk mengatasi opini majemuk dengan cara mengekstraksi entitas target atau

topik spesifik (aspek) terlebih dahulu dari sebuah kalimat, lalu mengakumulasi nilai sentimen yang secara khusus ditujukan pada masing-masing aspek tersebut.

Untuk mempermudah pemahaman konseptual terkait pemetaan masukan dan keluaran pada analisis sentimen, gambar 2.4 adalah ilustrasi alur kerja analisis sentimen pada data.



Gambar 2. 4 Ilustrasi Analisis Sentimen

Berdasarkan Ilustrasi Analisis Sentimen, analisis sentimen tradisional yang beroperasi pada tingkat dokumen maupun kalimat seringkali menghadapi kebuntuan (ambiguitas) ketika dihadapkan pada opini yang memuat berbagai sudut pandang dalam satu waktu. Hal ini sangat lazim ditemukan pada diskusi isu-isu geopolitik seperti konflik Timur Tengah, di mana warganet kerap menyanjung satu sisi namun mengkritik sisi lainnya dalam satu tarikan napas. Oleh karena itu, untuk menutupi kelemahan klasifikasi konvensional tersebut, arsitektur pemodelan dalam penelitian ini tidak berhenti pada pemrosesan kalimat secara umum, melainkan difokuskan

pada tingkat granular menggunakan pendekatan *Aspect-Based Sentiment Analysis* (ABSA) agar setiap muatan dimensi (Kemanusiaan, Ekonomi, dan Politik) dapat dipetakan secara independen dan akurat.

6. Analisis Sentimen Berbasis Aspek (*Aspect-Based Sentiment Analysis*)

Analisis Sentimen Berbasis Aspek (*Aspect-Based Sentiment Analysis* atau ABSA) merupakan pendekatan komputasional tingkat lanjut yang tidak hanya mendeteksi polaritas teks secara keseluruhan, tetapi juga mengidentifikasi entitas atau topik spesifik yang dievaluasi di dalam teks tersebut. Berbeda dengan klasifikasi sentimen konvensional yang hanya memberikan satu nilai sentimen umum, ABSA bekerja melalui tahapan ekstraksi aspek untuk menemukan objek diskusi terlebih dahulu, disusul dengan penentuan polaritas sentimen (positif, negatif, atau netral) yang ditujukan khusus pada masing-masing aspek yang ditemukan (Mumtaz, 2026). Pendekatan granular ini sangat esensial untuk memproses teks yang mengandung opini majemuk, di mana seseorang dapat menyampaikan dukungan dan kritik secara bersamaan dalam satu kalimat yang sama terhadap komponen masalah yang berbeda.

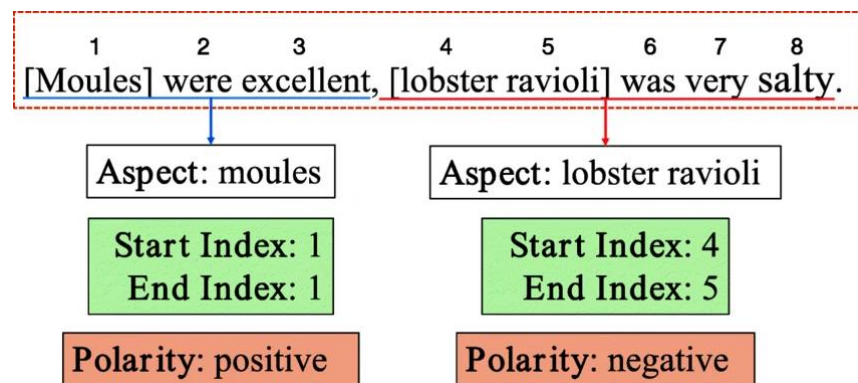
Secara operasional, arsitektur pemrosesan pada ABSA pada umumnya dibagi ke dalam dua sub-tugas (*sub-tasks*) utama yang bekerja secara berkesinambungan.

- a. Ekstraksi Aspek (*Aspect Term Extraction / ATE*): Merupakan tahapan sistem untuk mendeteksi kata atau frasa yang merepresentasikan objek diskusi (sebagai contoh, mengekstraksi target aspek ulasan).

b. Klasifikasi Sentimen Aspek (*Aspect Sentiment Classification / ASC*):

Merupakan tahapan untuk menentukan arah polaritas secara eksklusif pada masing-masing aspek yang telah diekstraksi.

Untuk merepresentasikan cara kerja kedua sub-tugas tersebut dalam menangani kalimat ulasan yang memiliki polaritas berlawanan, Gambar 2.5 adalah ilustrasi konseptual pemetaan ABSA.



Gambar 2. 5 Ilustrasi Konseptual Mekanisme Ekstraksi dan Klasifikasi pada ABSA

Sumber : (Zhou et al., 2020)

Berdasarkan Ilustrasi Konseptual Mekanisme Ekstraksi dan Klasifikasi pada ABSA, pemetaan matematis dari struktur kalimat ke dalam model *Aspect-Based Sentiment Analysis* (ABSA) dapat diartikulasikan melalui himpunan aspek. Diberikan sebuah himpunan kalimat ulasan sentimen, di mana di dalam sentimen terdapat himpunan aspek target A yang didefinisikan persamaan 2.1

$$A = \{ a_1, a_2, a_3, \dots, a_n \} \quad (2.1)$$

Model logis klasifikasi ABSA bertujuan untuk memprediksi fungsi probabilitas sentimen y sedemikian sehingga nilai polaritas untuk setiap aspek a_1 memenuhi kondisi pada Persamaan 2.2.

$$y(a_1) \in \{Positif, Negatif, Netral\} \quad (2.2)$$

Dalam konteks penelitian opini warganet Indonesia di platform YouTube terkait konflik Timur Tengah, metode ABSA memiliki peran krusial untuk membedah komentar-komentar publik yang seringkali kompleks dan bersifat ambigu. Melalui implementasi metode ini, opini publik yang panjang dan tidak terstruktur dapat didekomposisi secara sistematis ke dalam aspek-aspek spesifik yang telah ditentukan, seperti aspek kemanusiaan, ekonomi, dan politik, sehingga arah polaritas pada setiap dimensi persoalan geopolitik tersebut dapat dipetakan secara terukur (Setiawan, 2025). Pemisahan sentimen berdasarkan aspek ini akan menghasilkan wawasan analitik yang jauh lebih akurat dan representatif dalam memotret dinamika persepsi masyarakat, dibandingkan sekadar menyimpulkan keseluruhan kalimat komentar ke dalam satu nilai sentimen tunggal.

Meskipun memiliki keunggulan dekomposisi yang presisi, komputasi rumit dari algoritma klasifikasi pada ABSA tidak akan mampu mengekstraksi dan menilai fitur aspek secara optimal apabila data linguistik yang masuk masih berantakan dan penuh dengan derau teks (*noise*). Oleh karena itu, sebelum dikonversi menjadi matriks dan memasuki mesin pengklasifikasi sentimen, himpunan korpus data mentah tersebut wajib

dikondisikan terlebih dahulu melalui serangkaian tahapan pembersihan yang ketat pada tahapan pemrosesan teks (*text preprocessing*).

7. Pemrosesan Teks (*Text Preprocessing*)

Pemrosesan teks (*text preprocessing*) merupakan serangkaian tahapan rekayasa data yang krusial dalam disiplin pemrosesan bahasa alami, bertujuan untuk mengubah data teks mentah (*raw text*) yang tidak terstruktur dan penuh derau (*noise*) menjadi format matematis yang bersih dan terstandarisasi (Widianto, 2025). Mengingat sumber data pada penelitian ini berasal dari kolom komentar platform YouTube yang sangat dinamis, tidak baku, dan sarat akan singkatan maupun simbol, eksekusi pembersihan teks yang optimal akan berbanding lurus dengan kualitas fitur yang akan diekstraksi oleh algoritma (Soni, 2026).

Secara konseptual, tahapan preprocessing untuk teks berbahasa Indonesia bersumber media sosial umumnya diimplementasikan secara sekuensial melalui lima tahapan utama (Krouska et al., 2016), yaitu:

a. Pembersihan (*Cleansing*) dan Normalisasi

Proses eliminasi elemen non-alfabetik yang tidak relevan dengan analisis sentimen, seperti tautan (URL), *hashtag*, *username* (@), angka, tanda baca, serta emotikon. Tahap ini juga mencakup normalisasi kata-kata slang (bahasa gaul) atau singkatan menjadi kosa kata baku bahasa Indonesia (misalnya: kata "yg" diubah menjadi "yang", "tdk" menjadi "tidak").

b. Case Folding

Proses penyeragaman seluruh karakter alfabet di dalam dokumen menjadi huruf kecil (*lowercase*), guna memastikan mesin tidak menganggap kata "Konflik" dan "konflik" sebagai dua entitas fitur yang berbeda.

c. *Tokenisasi (Tokenizing)*

Proses pemecahan untaian kalimat yang panjang menjadi unit-unit kata tunggal atau token, yang umumnya dipisahkan berdasarkan karakter spasi.

d. *Stopword Removal*

Proses pemfilteran atau penghapusan kata hubung, kata ganti, dan kata depan yang memiliki frekuensi kemunculan tinggi namun tidak memiliki bobot sentimen atau makna leksikal yang signifikan (misalnya: "dan", "di", "ke", "dari", "yang").

e. *Stemming*

Proses morfologis untuk mereduksi kata-kata yang telah mengalami afiksasi (memiliki imbuhan) kembali ke bentuk kata dasarnya (misalnya: "menyerang" direduksi menjadi "serang"). Pada pemrosesan bahasa Indonesia, tahapan ini umumnya mengandalkan algoritma dan korpus dari pustaka eksternal seperti Sastrawi.

Untuk memberikan gambaran komprehensif mengenai transformasi data linguistik di dalam sistem, berikut adalah ilustrasi tahapan pemrosesan teks pada sebuah Tabel 2.1.

Tabel 2. 1 Ilustrasi Transformasi Pemrosesan Teks (Rajesh & Hiwarkar, 2023)

Tahapan Pemrosesan	Hasil Transformasi Teks
Teks Mentah (<i>Raw</i>)	"Wah kacau bgt!! Bantuan kemanusiaan di sana hancur lebur 🙏🙏 #Save"
<i>Cleansing & Normalisasi</i>	"Wah kacau banget Bantuan kemanusiaan di sana hancur lebur"
<i>Case Folding</i>	"wah kacau banget bantuan kemanusiaan di sana hancur lebur"
<i>Tokenizing</i>	["wah", "kacau", "banget", "bantuan", "kemanusiaan", "di", "sana", "hancur", "lebur"]
<i>Stopword Removal</i>	["kacau", "banget", "bantuan", "kemanusiaan", "hancur", "lebur"]
<i>Stemming</i>	["kacau", "banget", "bantu", "manusia", "hancur", "lebur"]

Setelah seluruh korpus komentar warganet berhasil melewati tahapan reduksi pada fase *stemming*, struktur kalimat telah berada dalam kondisi siap olah. Namun, algoritma klasifikasi seperti *Support Vector Machine* (SVM) merupakan model matematis yang tidak dapat memproses matriks data dalam format huruf (*string*). Oleh karena itu, kumpulan *token* kata dasar tersebut wajib ditransformasikan terlebih dahulu ke dalam bentuk representasi numerik menggunakan instrumen pembobotan statistik.

8. *Term Frequency-Inverse Document Frequency* (TF-IDF)

Metode *Term Frequency-Inverse Document Frequency* (TF-IDF) merupakan teknik pembobotan statistik ruang vektor yang secara luas

digunakan dalam komputasi penambangan teks untuk mengevaluasi seberapa esensial sebuah kata di dalam suatu dokumen relatif terhadap sekumpulan dokumen (korpus). Menurut (Septiani & Isabela, 2022), algoritma TF-IDF bekerja dengan mengombinasikan dua metrik utama: frekuensi kemunculan kata dalam satu unit teks (*Term Frequency*) dan inversi frekuensi dokumen yang mengandung kata tersebut dalam seluruh korpus (*Inverse Document Frequency*).

Perhitungan nilai *Term Frequency* (TF) dilakukan dengan mengukur rasio seberapa sering sebuah istilah atau kata (t) muncul dalam sebuah dokumen tertentu (d). Rumus matematis untuk normalisasi perhitungan TF dinyatakan pada persamaan (2.3).

$$TF(t, d) = \frac{f(t, d)}{\sum f(t^1, d)} \quad (2.3)$$

$TF(t, d)$ merupakan nilai *Term Frequency* dari kata t pada dokumen d , yang diperoleh dengan membagi $f(t, d)$ atau frekuensi kemunculan kata t dalam dokumen d , dengan $\sum f(t^1, d)$ yang menunjukkan total keseluruhan kata yang terdapat dalam dokumen d tersebut.

Berdasarkan perhitungan *Term Frequency* (TF) di atas, apabila kata "minyak" muncul sebanyak 2 kali dalam satu kalimat komentar yang memiliki total 10 kata, maka kalkulasi nilai TF adalah:

$$TF("minyak, d") = \frac{2}{10} = 0.2 \text{ (Term Frequency (TF))}$$

Di sisi lain, *Inverse Document Frequency* (IDF) berfungsi untuk mengendalikan bobot kata. Kata-kata yang sangat umum dan sering muncul

di hampir semua dokumen (seperti kata ganti) akan diberikan bobot rendah, sedangkan kata yang jarang muncul secara keseluruhan namun spesifik pada dokumen tertentu akan diberikan bobot tinggi. Mengacu pada (Mahendra et al., 2023), rumus perhitungan IDF sebagai persamaan (2.4) berikut.

$$IDF(t, d) = \log \left(\frac{N}{df(t)} \right) \quad (2.4)$$

Berdasarkan perhitungan IDF, apabila dataset penelitian memiliki total 1.000 baris komentar $N = 1000$ dan kata "minyak" muncul di 10 baris komentar yang berbeda ($df(t) = 10$), maka kalkulasi nilai IDF adalah:

$$IDF("minyak", D) = \log \left(\frac{1000}{10} \right) = \log(100) = 2 \text{ (IDF)}$$

Persamaan (2.4) menggunakan dua komponen utama dalam perhitungannya, yaitu variabel N dan $df(t)$. Variabel N merepresentasikan total keseluruhan jumlah dokumen atau baris komentar yang terdapat di dalam seluruh dataset D . Sementara itu, variabel $df(t)$ menunjukkan frekuensi dokumen, yaitu jumlah dokumen spesifik yang mengandung kata t di dalamnya. Bobot akhir ekstraksi fitur diperoleh dengan mengalikan nilai perhitungan TF dan IDF, menggunakan persamaan (2.5).

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D) \quad (2.5)$$

Berdasarkan persamaan di atas, bobot akhir representasi numerik untuk kata "minyak" pada dokumen tersebut adalah:

$$TFIDF(\text{minyak}) = 0.2 \times 2 = 0.4 \text{ (TF-IDF)}$$

Dalam konteks sistem analisis opini warganet terhadap konflik Timur Tengah, teks ulasan yang telah melalui tahapan preprocessing akan

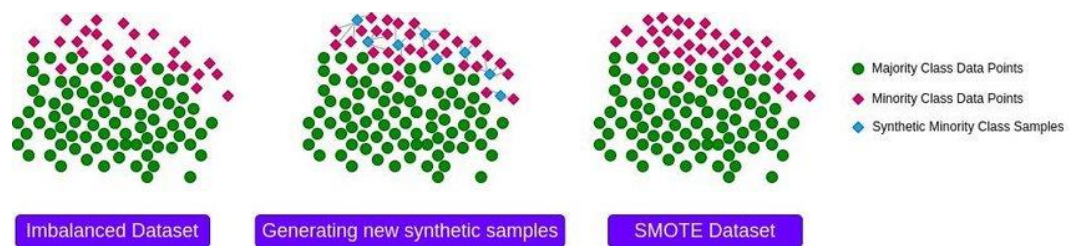
dikonversi menjadi representasi nilai TF-IDF. Himpunan nilai numerik berbentuk vektor matematis inilah yang selanjutnya diproses sebagai data masukan (*input*) bagi algoritma *Support Vector Machine* (SVM) untuk menentukan ruang pemisah kelas (*hyperplane*) dan memprediksi polaritas sentimen.

9. *Synthetic Minority Over-sampling Technique* (SMOTE)

Dalam studi klasifikasi teks, khususnya analisis sentimen opini publik terkait isu geopolitik, ketidakseimbangan kelas (*imbalanced data*) merupakan salah satu permasalahan fundamental yang sangat sering dijumpai. Kondisi ini muncul ketika jumlah sampel data pada satu kelas (misalnya kelas sentimen Netral atau aspek Politik) jauh lebih dominan dibandingkan dengan kelas lainnya (seperti kelas Positif atau aspek Ekonomi). Ketimpangan distribusi data ini secara matematis akan menyebabkan algoritma pembelajaran mesin, termasuk *Support Vector Machine* (SVM), mengalami bias prediksi. Algoritma akan cenderung mengenali dan memprediksi data uji sebagai bagian dari kelas mayoritas, sekaligus gagal menggeneralisasi karakteristik unik dari kelas minoritas akibat minimnya referensi fitur selama fase pelatihan.

Untuk mengatasi anomali distribusi tersebut, digunakan algoritma *Synthetic Minority Over-sampling Technique* (SMOTE). Berbeda dengan teknik random oversampling tradisional yang sekadar menduplikasi data minoritas secara persis sehingga memicu *overfitting*, SMOTE bekerja dengan cara menginterpolasi ruang fitur untuk menciptakan sampel data sintesis

(buatan) baru yang logis dan bervariasi. Representasi visual dari tahapan kerja algoritma SMOTE dalam menyeimbangkan distribusi kelas dapat dilihat pada Gambar 2.6.



Gambar 2. 6 Ilustrasi Mekanisme Interpolasi SMOTE

mekanisme kerja SMOTE dapat dibedah ke dalam tiga tahapan visual. Pada panel kiri (*Imbalanced Dataset*), terlihat kondisi awal pangkalan data di mana titik lingkaran hijau merepresentasikan sampel data dari kelas mayoritas yang jumlahnya sangat mendominasi, sedangkan titik belah ketupat merah muda merepresentasikan sampel data aktual dari kelas minoritas. Pada panel tengah (*Generating new synthetic samples*), algoritma SMOTE mulai beroperasi menggunakan pendekatan *K-Nearest Neighbors* (KNN). Algoritma ini memilih sebuah sampel acak dari kelas minoritas (merah muda), kemudian mengidentifikasi tetangga terdekatnya yang ditandai dengan garis-garis penghubung abu-abu. SMOTE kemudian menciptakan sampel sintetis baru yang direpresentasikan oleh titik belah ketupat berwarna biru dengan meletakkannya secara acak di sepanjang garis segmen yang menghubungkan sampel aktual dengan tetangganya tersebut. Hasil akhirnya terlihat pada panel kanan (*SMOTE Dataset*), di mana batas keputusan (*decision boundary*) kelas minoritas kini meluas dan distribusinya menjadi seimbang dengan kelas mayoritas. Proses pembentukan titik sampel

sintetis baru (belah ketupat biru) di sepanjang garis segmen tersebut direpresentasikan melalui komputasi matematis pada Persamaan (2.6).

$$X_{new} = X_i + \lambda(X_{zi} + X_i) \quad (2.6)$$

Komponen perhitungan pada persamaan di atas menjabarkan bahwa X_{new} merupakan vektor fitur dari sampel sintetis baru yang dihasilkan. Variabel X_i merepresentasikan titik vektor fitur dari sampel kelas minoritas awal yang dipilih secara acak. Variabel X_{zi} melambangkan titik vektor dari salah satu k tetangga terdekat (*nearest neighbors*) yang terpilih, sedangkan λ adalah besaran angka acak (*random number*) berdistribusi seragam dengan rentang nilai antara 0 hingga 1.

Sebagai contoh simulasi komputasi numerik, asumsikan sistem sedang mengevaluasi dua fitur kata pembobotan TF-IDF, yaitu kata "boikot" dan "krisis", pada sentimen kelas minoritas. Sebuah dokumen ulasan terpilih secara acak memiliki nilai vektor fitur $X_i = [0.4, 0.2]$. Algoritma KNN kemudian menemukan tetangga terdekat dari ulasan tersebut dengan nilai vektor $X_{zi} = [0.6, 0.3]$. Jika mesin membangkitkan nilai acak $\lambda = 0.5$, maka proses pembentukan sampel buatan adalah sebagai berikut:

$$X_{new} = [0.4, 0.2] + 0.5 \times ([0.6, 0.3] - [0.4, 0.2])$$

$$X_{new} = [0.4, 0.2] + 0.5 \times [0.2, 0.1]$$

$$X_{new} = [0.4, 0.2] + [0.1, 0.05]$$

$$X_{new} = [0.5, 0.25] \text{ (SMOTE)}$$

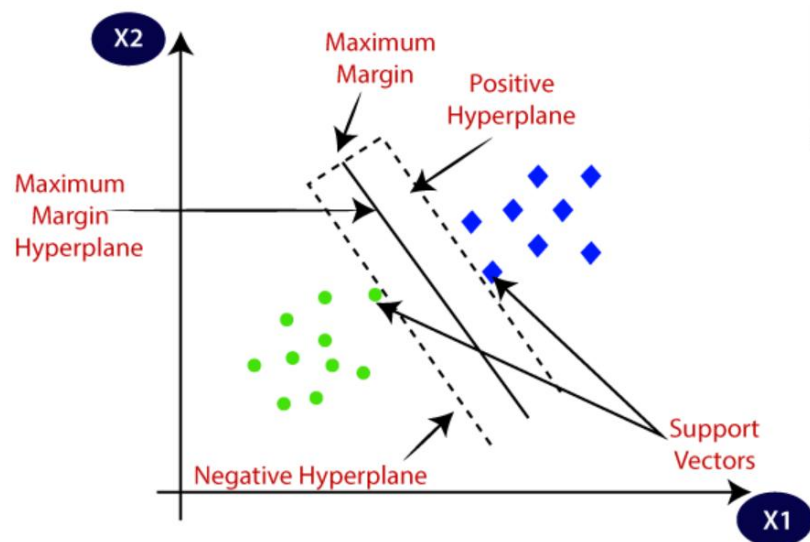
Berdasarkan kalkulasi matriks di atas, SMOTE berhasil menghasilkan sebuah sampel data latih sintetis baru dengan nilai vektor fitur

kata sebesar 0.5 dan 0.25. Sampel sintetik ini, beserta ribuan sampel lainnya yang digeneralisasi oleh mesin, akan disuntikkan ke dalam korpus data latih sentimen minoritas hingga distribusinya seimbang dan setara dengan jumlah data pada kelas mayoritas.

10. *Support Vector Machine (SVM)*

Menurut (Aulia Raihan & Yuliansyah, 2025), algoritma *Support Vector Machine (SVM)* merupakan salah satu metode pembelajaran mesin berbasis terbimbing (*supervised learning*) yang terbukti sangat tangguh dalam menangani komputasi klasifikasi teks berdimensi tinggi. Prinsip kerja fundamental dari SVM adalah merekayasa dan menemukan *hyperplane* (bidang pemisah) yang paling optimal di dalam ruang vektor untuk memisahkan himpunan data ke dalam kelas-kelas yang berbeda. Proses optimasi ruang ini dilakukan dengan cara memaksimalkan margin, yaitu jarak metrik tegak lurus antara *hyperplane* dengan *support vector*. *Support vector* sendiri merupakan titik-titik sampel data pelatihan yang posisinya paling dekat dengan garis batas keputusan (*decision boundary*). Semakin lebar dan terisolasi margin yang terbentuk, semakin tinggi tingkat generalisasi algoritma dalam mengklasifikasikan varian data teks baru yang belum pernah dijumpai sebelumnya (Putra et al., 2024).

Untuk memberikan visualisasi konseptual mengenai bagaimana algoritma ini memisahkan dua kelas data yang berbeda, Gambar 2.7 berikut adalah ilustrasi pembentukan *hyperplane* pada SVM.



Gambar 2. 7 Ilustrasi *Hyperplane*, *Margin*, dan *Support Vector* pada SVM

Dalam implementasi sistem analisis sentimen terhadap kompleksitas opini terkait konflik Timur Tengah, SVM dioperasikan untuk membedah vektor fitur teks (hasil pembobotan TF-IDF) menjadi polaritas positif maupun negatif secara spesifik pada masing-masing ulasan. Mengingat struktur data linguistik dari kolom komentar *YouTube* sangat dinamis dan bersifat non-linear, arsitektur SVM memanfaatkan fungsi kernel untuk mentransformasikan data tersebut ke dalam ruang dimensi baru tanpa memicu lonjakan beban komputasi secara eksponensial (Wardoyo et al., 2026). Secara umum, terdapat tiga jenis kernel yang sering diimplementasikan:

a. Kernel Linear

Kernel Linear merupakan jenis kernel yang paling sederhana di antara kernel lainnya. Pendekatan ini sangat berguna dan efisien ketika diterapkan pada vektor data yang besar dan renggang (*large sparse data vectors*). Kernel ini beroperasi murni berdasarkan perkalian titik antar

vektor masukan. Rumus matematis Kernel Linear didefinisikan sebagai persamaan 2.7.

$$K(x, u) = x^T \cdot u \quad (2.7)$$

Sebagai representasi perhitungan numerik, asumsikan terdapat vektor fitur pembobotan TF-IDF $x = [0.4, 0.2]$ dan vektor ulasan pembanding $u = [0.5, 0.1]$. Maka perhitungan kalkulasi Kernel Linear adalah:

$$K(x, u) = (0.4 \times 0.5) + (0.2 \times 0.1)$$

$$K(x, u) = 0.20 + 0.02 = 0.22 \text{ (Kernel)}$$

Dengan demikian, Kernel Linear terbukti memberikan hasil komputasi yang efisien secara linear, menjadikannya pilihan ideal untuk pemrosesan teks dasar tanpa perlu transformasi dimensi tingkat tinggi.

b. Kernel *Polynomial*

Kernel Polynomial berfungsi memetakan himpunan data awal ke dalam ruang dimensi komputasi yang lebih tinggi. Secara performa, algoritma ini ideal digunakan untuk menemukan batasan kelas yang melengkung dan dinilai lebih unggul dalam menghindari nilai nol pada perhitungan matriks. Rumus matematis fungsi polinomial dengan syarat nilai parameter $a > 0$ didefinisikan sebagai persamaan 2.8.

$$K(x, u) = (ax^T + c)^q \quad (2.8)$$

Sebagai representasi perhitungan numerik, dengan menggunakan nilai vektor masukan *dot product* yang sama serta ($ax^T u =$

0.22) diasumsikan sistem menggunakan parameter $a = 1$, konstanta $c = 1$, serta derajat fungsi $q = 2$, maka kalkulasinya adalah:

$$K(x, u) = (1 \times (0.22) + 1)^2 = (1.22)^2 = 1.4884$$

Hasil pemetaan matriks tersebut menunjukkan bahwa Kernel Polynomial secara efektif memperluas jarak metrik antar kelas data melalui perhitungan pangkat berpola, sehingga sangat cocok untuk memisahkan batasan kelas keputusan (*decision boundary*) yang melengkung.

c. Kernel RBF (*Radial Basis Function*)

Kernel Gaussian RBF merupakan fungsi matematis non-linear yang bersifat serbaguna (*general-purpose*) dan umumnya menjadi pilihan utama untuk diaplikasikan ketika peneliti tidak memiliki pengetahuan awal (*prior knowledge*) terkait sebaran distribusi datanya. Operasi perhitungan RBF didasarkan pada pengukuran jarak kuadrat antar vektor. Rumus matematis Kernel Gaussian RBF didefinisikan sebagai persamaan 2.9.

$$K(x, u) = \exp \left(- \frac{\|x - u\|^2}{2\sigma^2} \right) \quad (2.9)$$

Sebagai representasi perhitungan numerik, diasumsikan sistem menggunakan parameter skala varians $\sigma^2 = 2$. Kalkulasi kuadrat jarak antara vektor $x = [0.4, 0.2]$ dan $u = [0.5, 0.1]$ akan dihitung terlebih

dahulu, yang menghasilkan nilai 0.02. Hasil jarak tersebut kemudian dieksekusi ke dalam fungsi eksponensial Kernel Gaussian RBF:

$$K(x, u) = \exp\left(-\frac{0.02}{2}\right) = \exp(-0.01) = 0.9900 \text{ (Kernel)}$$

Melalui pendekatan pengukuran metrik Euclidean tersebut, Kernel RBF memproyeksikan data ke dalam dimensi tak terhingga, menjadikannya fungsi matematis yang sangat fleksibel dan andal untuk menangani sebaran opini yang sangat kompleks dan tumpang tindih. Representasi matematis dasar dari sebuah hyperplane linear pada ruang fitur klasifikasi ini didefinisikan melalui persamaan (2.10).

$$(f(x) = w \cdot x + b) \quad (2.10)$$

Pada Persamaan (2.10), $(f(x))$ merepresentasikan nilai kontinuitas atau jarak geometris dokumen terhadap garis pembatas. Variabel x merupakan vektor fitur masukan yang mewakili nilai bobot kata (seperti nilai TF-IDF) dari suatu dokumen ulasan. Sementara itu, variabel w melambangkan vektor bobot (*weight*) yang menentukan tingkat kontribusi masing-masing fitur kata, dan b adalah nilai bias yang berfungsi mengatur posisi orientasi *hyperplane*.

Model menentukan kelas sentimen menggunakan fungsi *sign* (tanda) berdasarkan hasil perhitungan Persamaan (2.11).

$$y = \text{sign}(w \cdot x + b) \quad (2.11)$$

Pada Persamaan (2.11), variabel y menunjukkan label kelas akhir hasil prediksi sentimen. Fungsi *sign* (tanda) berperan untuk mengevaluasi

hasil operasi linear di dalamnya, kemudian mengonversi nilai tersebut menjadi output diskret berupa nilai +1 jika hasilnya positif, atau -1 jika hasilnya negatif.

Sebagai representasi perhitungan numerik dari persamaan (2.8), asumsikan sebuah kalimat ulasan telah diproses menjadi vektor TF-IDF dengan dua fitur kata, sehingga nilai masukannya adalah $x = [0.4, 0.2]$. Berdasarkan hasil pelatihan, algoritma telah menetapkan bobot vektor $w = [0.5, -0.3]$ dan nilai bias $b = 0.1$. Maka, kalkulasi probabilitas kelas sentimen untuk kalimat tersebut:

$$f(x) = (0.4 \times 0.5) + (0.2 \times 0.3) + 0.1$$

$$f(x) = (0.20) + (-0.06) + 0.1 = 0.24$$

Karena nilai akhir matematis menghasilkan angka positif ($0.24 > 0$), maka fungsi sign akan membulatkannya menjadi +1. Dengan demikian, mesin secara otomatis mengklasifikasikan kalimat ulasan tersebut ke dalam sentimen Positif.

11. Pengujian Model (*Confusion Matrix*)

Pengujian keandalan performa model *Machine Learning*, khususnya pada algoritma klasifikasi seperti *Support Vector Machine* (SVM), mutlak memerlukan instrumen evaluasi yang terukur secara kuantitatif (Moningkey et al., 2025). Mengingat arsitektur sistem ini dirancang untuk mengklasifikasikan tiga label kelas (seperti sentimen Positif, Negatif, Netral, maupun aspek Ekonomi, Kemanusiaan, Politik), maka model pengujian matriks 2x2 konvensional tidak lagi relevan. Sistem ini mengadopsi struktur

Confusion Matrix berukuran 3x3 (multiclass classification) yang dievaluasi menggunakan pendekatan *One-vs-Rest* (OvR).

Pendekatan OvR bekerja dengan mengisolasi satu kelas target untuk dievaluasi secara biner dengan mengelompokkan sisa kelas lainnya sebagai kondisi Rest (bukan kelas target). Representasi matematis pemetaan koordinat sel pada matriks 3x3 dieksplisitkan melalui struktur skema persamaan 2.12 berikut.

$$\text{Confusion Matrix } (3 \times 3) = \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{pmatrix} \quad (2.12)$$

Indeks baris melambangkan kelas aktual (*actual class*) dan indeks kolom melambangkan kelas hasil prediksi mesin (*predicted class*). Berdasarkan struktur matriks di atas, ekstraksi parameter evaluasi seperti *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN) untuk suatu kelas tertentu (misalnya Kelas 1) didefinisikan secara matematis melalui rumus-rumus berikut (Harfaitulloh, 2026).

1. *True Positif* (TP)_{C1} = M₁₁(Data kelas 1 yang diprediksi benar sebagai kelas 1).
2. *False Positif* (FP)_{C1} = M₂₁ + M₃₁(Jumlah data kelas lain yang salah diprediksi sebagai kelas 1).
3. *False Negatif* (FN)_{C1} = M₁₂ + M₁₃(Jumlah data kelas 1 yang salah diprediksi sebagai kelas lain).
4. *True Negatif* (TN)_{C1} = M₂₂ + M₂₃ + M₃₂ + M₃₃ (Seluruh data di luar kelas 1 yang berhasil diprediksi di luar kelas 1).

Setelah komponen-komponen ekstraksi biner tersebut didapatkan dari matriks 3x3, barulah sistem menghitung performa klasifikasi menggunakan *Classification Performance Evaluation Metrics* yang mencakup parameter standar berikut.

a. *Accuracy* (Akurasi)

Accuracy digunakan untuk mengukur rasio ketepatan prediksi model secara keseluruhan di antara total dokumen yang diuji. Formula akurasi dinyatakan dalam Persamaan (2.13).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.13)$$

Komponen perhitungan pada Persamaan (2.8) terdiri dari empat variabel representasi hasil prediksi. Variabel *True Positive* (TP) menunjukkan jumlah data berlabel positif yang berhasil diprediksi secara tepat oleh model sebagai kelas positif, sedangkan *True Negative* (TN) menunjukkan jumlah data berlabel negatif yang berhasil diprediksi secara tepat sebagai kelas negatif. Sebaliknya, *False Positive* (FP) merupakan kondisi di mana data berlabel negatif salah diprediksi oleh model menjadi kelas positif, dan *False Negative* (FN) merupakan kondisi ketika data berlabel positif salah diprediksi menjadi kelas negatif. Berdasarkan asumsi numerik di atas, kalkulasi akurasinya adalah:

$$Accuracy = \frac{60 + 85}{60 + 85 + 40 + 15} = \frac{145}{200} = 0.725(72.5\%)$$

Nilai akurasi sebesar 72.5% ini menyimpulkan bahwa model SVM secara umum memiliki performa yang cukup baik dan dominan dalam

memprediksi keseluruhan data uji dengan benar, baik untuk sentimen positif maupun negatif.

b. *Precision* (Presisi)

Precision mengukur tingkat ketepatan antara informasi yang diminta pengguna dengan hasil prediksi yang dikeluarkan oleh sistem. Formula presisi dinyatakan dalam Persamaan (2.14).

$$Precision = \frac{TP}{TP + FP} \quad (2.14)$$

Berdasarkan asumsi numerik di atas, kalkulasi presisinya adalah:

$$Precision = \frac{60}{60 + 40} = \frac{60}{100} = 0.60(60\%)$$

Hasil presisi sebesar 60.0% ini menyimpulkan bahwa ketika model memprediksi suatu ulasan sebagai sentimen positif, tingkat kebenaran prediksi tersebut adalah sebesar 60.0%, sedangkan 40.0% sisanya merupakan kesalahan prediksi terhadap data yang sebenarnya bernilai negatif (false positive).

c. *Recall* (Sensitivitas)

Recall digunakan untuk mengukur keberhasilan model dalam menemukan kembali seluruh informasi dari kelas positif yang ada di dalam data aktual. Formula recall dinyatakan dalam Persamaan (2.15).

$$Recall = \frac{TP}{TP + FN} \quad (2.15)$$

Berdasarkan asumsi numerik di atas, kalkulasi sensitivitasnya adalah:

$$Recall = \frac{60}{60 + 15} = \frac{60}{75} = 0.80(80\%)$$

Capaian nilai *recall* sebesar 80.0% ini memberikan kesimpulan bahwa model memiliki sensitivitas yang tinggi dalam mengenali data aktual bersentimen positif, di mana model mampu menjaring 80.0% dari total seluruh data positif yang tersedia di dalam dataset.

d. *F1-Score* (Rata-rata Harmonis)

F1-Score menghitung rata-rata harmonis dari komponen presisi dan recall untuk memberikan gambaran performa model pada kondisi data yang tidak seimbang (*imbalanced data*). Formula F1-Score dinyatakan dalam Persamaan (2.16).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.6)$$

Berdasarkan asumsi numerik di atas, kalkulasi F1-Score adalah:

$$F1 = 2 \times \frac{0.60 \times 0.80}{0.60 + 0.80} = 2 \times \frac{0.48}{1.40} = 0.685(68.5\%)$$

Nilai F1-Score sebesar 68.6% ini memberikan kesimpulan akhir bahwa model klasifikasi memiliki keseimbangan performa akurasi yang cukup stabil dan tangguh, meskipun terdapat ketimpangan nilai yang cukup kontras antara hasil presisi dan recall sebelumnya.

Meskipun evaluasi menggunakan *Confusion Matrix* dan *Classification Performance Evaluation Metrics* telah mampu membuktikan keandalan matematis serta akurasi dari algoritma prediksi, keakuratan angka tersebut belum merepresentasikan kesiapan perangkat lunak secara utuh untuk dioperasikan oleh pengguna akhir. Sebuah platform analisis sentimen berbasis web yang tangguh tidak hanya

bergantung pada kecerdasan model prediktifnya, tetapi juga membutuhkan landasan arsitektur logika yang terstruktur serta jaminan bahwa seluruh fungsionalitas antarmukanya berjalan tanpa galat (error). Oleh karena itu, perancangan perangkat lunak ini dimodelkan secara visual menggunakan *Unified Modeling Language* (UML) sebelum pada akhirnya divalidasi kelayakan operasionalnya melalui metode pengujian *Blackbox*.

12. Flowchart

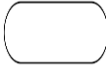
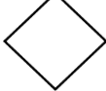
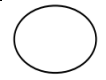
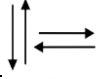
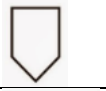
Flowchart atau diagram alir merupakan representasi grafis standar yang digunakan secara luas dalam rekayasa perangkat lunak untuk memvisualisasikan algoritma, urutan langkah-langkah prosedural, serta percabangan logika pengambilan keputusan dari suatu sistem atau program (Adnan et al, 2023). Penggunaan flowchart berperan sebagai jembatan komunikasi teknis yang mentranslasikan abstraksi proses bisnis ke dalam bentuk simbol-simbol geometris baku, sehingga kompleksitas alur komputasi dapat dipahami secara transparan oleh pengembang maupun pemangku kepentingan.

Dalam proses perancangan sistem, diagram ini disusun sebelum penulisan baris kode dieksekusi guna mendeteksi potensi cacat logika (*logical flaw*) atau kebuntuan alur (infinite loop). Keunggulan utama dari flowchart terletak pada kemampuannya untuk mendeskripsikan secara presisi di mana sebuah proses bermula, bagaimana aliran data dimanipulasi melalui berbagai

kondisi instruksi (seperti proses pembersihan teks atau klasifikasi mesin), dan di mana operasional sistem tersebut berakhir (Hidayanto, 2021).

Berdasarkan standarisasi perancangan sistem informasi, penyusunan diagram alir harus mematuhi konvensi simbol geometris yang telah ditetapkan secara universal. Penjabaran representasi simbol-simbol operasional baku yang digunakan dalam pembentukan Flowchart sistem dipaparkan secara rinci pada Tabel 2.2 berikut.

Tabel 2. 2 Simbol Standar *Flowchart*

No	Simbol	Nama Simbol	Fungsi Simbol
1.		<i>Terminal</i>	Awal atau akhir suatu program (Prosedur).
2.		<i>Output/Input</i>	Proses <i>input</i> atau <i>output</i> terlepas dari jenis perangkat.
3.		<i>Process</i>	Proses operasional <i>computer</i> .
4.		<i>Decision</i>	untuk menunjukkan bahwa situasi tertentu menghasilkan dua opsi, ya atau tidak
5.		<i>On Page Connector</i>	Proses terhubung satu sama lain pada halaman yang sama.
6.		<i>Flow</i>	Menyatakan jalannya arus suatu proses.
7.		<i>Database</i>	Tempat penyimpanan data dalam sistem.
8.		<i>Stored data</i>	Menunjukkan data atau hasil proses yang disimpan ke dalam <i>database</i> sistem.
9.		<i>Off Page Connector</i>	Menunjukkan penghubung alur <i>flowchart</i> ke halaman lain.
10.		<i>Document</i>	Menunjukkan keluaran berupa dokumen atau laporan.

Berdasarkan penjabaran fungsi dan standarisasi simbol pada tabel di atas, penerapan diagram alir (*flowchart*) dalam rancang bangun sistem

klasifikasi sentimen ini memiliki urgensi struktural yang tinggi. Diagram alir ini tidak sekadar mendokumentasikan urutan komputasi secara statis, melainkan bertindak sebagai instrumen validasi awal untuk memastikan bahwa seluruh logika pemrosesan beroperasi secara runut. Pemetaan yang presisimulai dari tahap ekstraksi data ulasan mentah, normalisasi teks melalui *Natural Language Processing* (NLP), hingga eksekusi prediksi kelas menggunakan algoritma *Support Vector Machine* (SVM) sangat krusial untuk mencegah terjadinya celah logika (*logical flaw*). Kesesuaian antara pemodelan diagram alir ini dengan kebutuhan arsitektur perangkat lunak pada akhirnya akan menjamin efisiensi penulisan baris kode program dan meminimalisir persentase galat (*error*) saat sistem dieksekusi di lingkungan produksi.

13. Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan bahasa pemodelan visual berorientasi objek yang digunakan untuk mendokumentasikan, menspesifikasikan, dan membangun rancangan perangkat lunak sehingga kebutuhan dan struktur sistem dapat dipahami secara konsisten oleh seluruh pemangku kepentingan (Paramita et al, 2024). Dalam praktik perancangan, UML membantu merumuskan kebutuhan fungsional melalui *use case diagram* serta mengarahkan turunan desain berikutnya agar pengembangan sistem lebih terstruktur, termasuk pada konteks sistem yang menuntut integrasi lintas aplikasi (Rospricilia, 2024).

Dengan demikian, penerapan UML tidak hanya memperjelas komunikasi desain, tetapi juga meningkatkan kualitas pemodelan (misalnya ketepatan notasi dan relasi pada *use case*) sehingga risiko kesalahan rancangan dapat ditekan sejak tahap analisis dan desain (Aryani et al., 2024). Sebagai lanjutan, UML memvisualisasikan kebutuhan sistem secara konsisten melalui representasi diagram perilaku dan struktur, agar arsitektur logika sesuai dengan spesifikasi pengguna (Paramita et al., 2025). Dalam perancangan sistem klasifikasi sentimen ini, UML bertindak sebagai cetak biru (*blueprint*) untuk menata alur komputasi, mulai dari mekanisme unggah dataset CSV, pemrosesan bahasa alami di sisi peladen, hingga penyajian luaran visualisasi kelas sentimen, sehingga meminimalisir risiko anomali saat fase penulisan kode program. Dalam integrasi sistem, ketepatan notasi UML serta pemodelan *Integration Use Case* (IUC) ditekankan agar kebutuhan integrasi tergambar jelas sejak awal.

a. *Use Case Diagram*

Use Case Diagram berfungsi untuk memetakan kebutuhan fungsional sistem dengan menonjolkan pihak yang berinteraksi (aktor) serta layanan yang diberikan sistem (*use case*). Tabel 2.3 adalah representasi simbol-simbol standar yang digunakan dalam penyusunan *Use Case Diagram*.

Tabel 2. 3 *Use Case Diagram* (Rospricilia, 2024)

NO	SIMBOL	NAMA	KETERANGAN
----	--------	------	------------

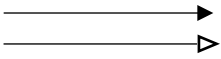
1		<i>Use Case</i>	Sistem bekerja dengan membagi tugas ke beberapa bagian, lalu saling berkomunikasi dan bertukar informasi, baik satu sama lain maupun dengan pengguna.
2		<i>Actor</i>	Orang, kegiatan, atau sistem lain yang ikut terlibat dan berinteraksi dengan sistem informasi, tapi tidak termasuk bagian dari sistem itu sendiri.
3		<i>Assosiation</i>	Hubungan antar aktor dan <i>Use Case</i> yang berpartisipasi.
4		Ekstensi / <i>extend</i>	Relasi <i>Use Case</i> tambahan ke sebuah <i>use case</i> yang dapat berdiri sendiri.
5		Generalisasi	Hubungan generalisasi dan spesialisasi antara dua buah <i>Use Case</i> yang mana fungsi yang satu lebih umum dari yang lainnya.
6		<i>Include Use Case</i>	Relasi tambahan pada sebuah <i>use case</i> , yaitu <i>use case</i> yang ditambahkan membutuhkan <i>use case</i> ini agar fungsinya dapat berjalan.

Diagram ini meringkas hubungan pengguna sistem secara jelas dan sering dijadikan landasan untuk menyusun desain lanjutan dalam pengembangan perangkat lunak (Dharmawan, 2023). Pada unsur simbolnya, aktor biasanya divisualisasikan sebagai gambar manusia sederhana (*stick figure*) untuk mewakili entitas eksternal, sedangkan *use case* digambarkan berbentuk elips yang menunjukkan fungsi/fitur yang

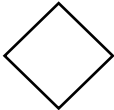
digunakan. Selain itu, *Use Case Diagram* membantu mengidentifikasi kebutuhan pengguna dan alur interaksi utama secara ringkas tanpa harus menjelaskan detail teknis implementasi.

b. *Activity Diagram*

Activity Diagram (Diagram Aktivitas) berfungsi untuk menggambarkan aliran kerja (*workflow*) maupun urutan kegiatan dalam suatu proses atau sistem dari tahap awal hingga akhir. Simbol-simbol standar yang digunakan dalam penyusunan *Activity Diagram* dijelaskan pada Tabel 2.3.

Tabel 2. 4 Simbol *Activity Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1	●	Status Awal	Setiap diagram aktivitas diawali dari kondisi awal, yaitu titik pertama saat proses atau alur kerja mulai berjalan. Kondisi awal tersebut direpresentasikan dengan simbol lingkaran hitam terisi dan berfungsi sebagai penanda dimulainya keseluruhan rangkaian aktivitas pada sistem yang sedang dimodelkan.

			Aktivitas dalam suatu sistem merepresentasikan rangkaian aksi atau prosedur yang dijalankan oleh sistem.
2		Aktivitas	Umumnya, aktivitas tersebut dituliskan dengan diawali kata kerja karena menegaskan bentuk tindakan yang dilakukan, misalnya “mengolah data” atau “mengirim
3		Percabangan	Relasi bercabang yang digunakan ketika terdapat lebih dari satu alternatif aktivitas.
4		Penggabungan	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5		Status Akhir	<i>Final state</i> yang menandakan seluruh rangkaian aktivitas telah selesai. Final state ditunjukkan dengan simbol lingkaran konsentris dan

berfungsi sebagai penutup proses. Keberadaannya memastikan alur kerja memiliki titik akhir yang jelas serta tidak berhenti pada kondisi yang menggantung.

Sumber: Nasution & Utami, 2024

Melalui diagram ini, setiap aktivitas dapat dipetakan secara terstruktur beserta relasi antar aktivitasnya, sehingga pengembang dan pemangku kepentingan lebih mudah memahami mekanisme serta dinamika proses yang dimodelkan. Dengan demikian, *Activity Diagram* menjadi representasi visual yang menonjolkan cara proses berlangsung dari tahap awal hingga akhir melalui rangkaian aktivitas yang saling terhubung (Amaliyah et al., 2023).

Pada kegiatan analisis dan perancangan sistem, *Activity Diagram* dimanfaatkan untuk memodelkan *workflow* yang terdapat pada sistem, proses bisnis, maupun fitur/menu dalam perangkat lunak. Penyusunan diagram ini membantu perancang menjabarkan langkah-langkah proses secara berurutan, menentukan titik pengambilan keputusan, serta mengidentifikasi kemungkinan adanya percabangan maupun aktivitas paralel yang perlu ditangani dalam rancangan. Oleh karena itu, *Activity Diagram* berperan penting dalam memperjelas kebutuhan fungsional

serta menyelaraskan pemahaman antara hasil analisis kebutuhan dengan rancangan proses yang akan diterapkan (Aryani et al., 2024).

c. *Class Diagram*

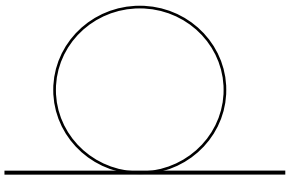
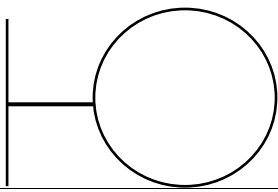
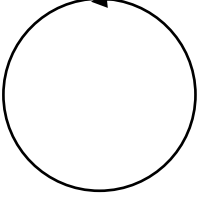
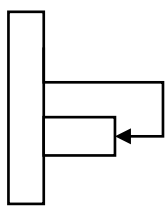
Class Diagram merupakan bagian dari diagram struktur (structural diagram) pada *Unified Modeling Language* (UML) yang berfungsi untuk menampilkan gambaran struktur statis sebuah sistem. Diagram ini menyajikan kelas-kelas beserta hubungan di antaranya melalui simbol dan notasi grafis secara ringkas, sehingga pemodel dapat menjelaskan komponen inti sistem (misalnya kelas) serta keterkaitan antarkomponen dalam pendekatan berorientasi objek. Oleh karena itu, *Class Diagram* menjadi dasar yang penting untuk memahami susunan elemen sistem sebelum dikembangkan lebih lanjut ke tahap perancangan rinci dan implementasi (Alturas, 2023).

Pada proses pengembangan perangkat lunak, *Class Diagram* tidak hanya dimanfaatkan sebagai dokumentasi desain, tetapi juga sebagai artefak yang dapat dievaluasi untuk meningkatkan mutu rancangan. Salah satu penerapannya ialah penggunaan metode berbasis *machine learning* untuk melakukan penilaian diagram secara otomatis, misalnya melalui analisis karakteristik kompleksitas desain seperti *coupling* dan kedalaman pewarisan, sehingga potensi kelemahan desain dapat terdeteksi sejak dini. Dengan demikian, *Class Diagram* dapat berperan sebagai objek analisis yang mendukung peningkatan *maintainability* serta kualitas perangkat lunak secara keseluruhan. (Shehzadi et al., 2025).

d. *Sequence Diagram*

Sequence Diagram berfungsi untuk memodelkan interaksi dinamis antarobjek atau komponen sistem melalui rangkaian pesan (*message*) yang berlangsung seiring berjalannya waktu. Diagram ini direpresentasikan dalam dua dimensi utama, yaitu dimensi vertikal yang menunjukkan urutan waktu dan dimensi horizontal yang melambangkan objek-objek atau entitas yang saling berinteraksi. Untuk mempermudah pemahaman mengenai komponen penyusunnya, Tabel 2.4 adalah representasi simbol-simbol standar yang digunakan dalam penyusunan *Sequence Diagram* beserta keterangan fungsinya.

Tabel 2. 5 Simbol *Sequence Diagram* (Septiansyah et al, 2024)

No	Simbol	Nama	Keterangan
1		<i>Entity Class</i>	Gambar Sistem sebagai landasan dalam Menyusun basis data
2		<i>Boundary Class</i>	Menangani komunikasi antar lingkungan sistem
3		<i>Control Class</i>	Bertanggungjawab terhadap kelas-kelas objek yang berisi logika
4		<i>Recursive</i>	Pesan untuk dirinya

5		<i>Activation</i>	Mewakili proses durasi aktivitas sebuah operasi
6		<i>Life Line</i>	Komponen yang digambarkan garis putus terhubung dengan objek

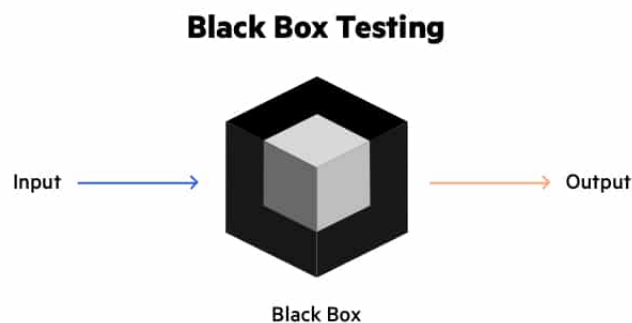
Diagram ini menonjolkan aspek urutan kejadian dengan memperlihatkan “kapan” serta “bagaimana” suatu pesan dikirim dan diterima oleh tiap objek, sehingga perilaku sistem pada suatu skenario dapat dipahami secara lebih terstruktur dan nyata. Oleh sebab itu, *Sequence Diagram* membantu menggambarkan alur komunikasi yang merepresentasikan fungsi serta respons sistem sesuai kebutuhan yang diharapkan (Kong et al., 2025).

Pada tahap analisis dan perancangan, *Sequence Diagram* digunakan untuk merinci *use case* menjadi urutan interaksi yang melibatkan aktor/objek, pemanggilan operasi, dan keterkaitan proses antarkomponen. Hasil pemodelan ini dapat dijadikan dasar penyusunan pengujian, misalnya melalui *path coverage* atau cakupan transisi yang kemudian diterjemahkan menjadi *test set* dan *test plan*. Selain itu, representasi kronologis pada *Sequence Diagram* membantu mengidentifikasi ketergantungan dan potensi titik kegagalan sejak dini sehingga skenario uji dapat disusun lebih terarah. Dengan demikian, *Sequence Diagram* berfungsi tidak hanya sebagai

dokumentasi rancangan, tetapi juga mendukung verifikasi dan perencanaan pengujian berbasis model (Górski & Stecz, 2024).

14. Pengujian *Black box*

Pengujian *Black Box* merupakan metode pengujian perangkat lunak yang berfokus pada validasi spesifikasi fungsional sistem tanpa melibatkan pemeriksaan terhadap struktur kode atau logika internal program. Pengujian ini dilakukan secara eksklusif dari perspektif pengguna akhir dengan mengevaluasi kesesuaian antara data masukan (*input*) yang dieksekusi melalui antarmuka dengan luaran (*output*) yang dihasilkan oleh sistem. Metode ini memastikan bahwa aplikasi berjalan sesuai dengan perancangan tanpa perlu mengintervensi atau membongkar mekanisme algoritma yang bekerja di latar belakang (Pratama et al., 2023).



Gambar 2. 8 Pengujian Blackbox

Untuk memberikan gambaran operasional mengenai bagaimana pengujian ini dieksekusi pada fungsi perangkat lunak, Tabel 2.6 berikut adalah ilustrasi tabel skenario pengujian *Black Box*.

Tabel 2. 6 Ilustrasi Skenario Pengujian *Black Box*

Skenario Pengujian	Data Masukan (<i>Input</i>)	Hasil yang Diharapkan (<i>Expected Output</i>)	Kesimpulan
Menguji fitur unggah dataset berformat CSV.	Pengguna mengunggah dokumen dengan ekstensi .csv.	Sistem menerima dokumen, menyimpannya ke dalam basis data, dan menampilkan notifikasi sukses.	Valid
Menguji validasi format dokumen.	Pengguna mengunggah dokumen dengan ekstensi .pdf atau .docx.	Sistem menolak dokumen dan memunculkan pesan peringatan "Format tidak didukung".	Valid
Menguji eksekusi tombol "Analisis Sentimen".	Pengguna menekan tombol "Proses Analisis" pada halaman Dashboard.	Sistem memproses dataset dan menampilkan visualisasi grafik secara interaktif.	Valid

Dalam konteks rancang bangun sistem klasifikasi sentimen berbasis aspek ini, pengujian *Black Box* memiliki peran krusial untuk memastikan bahwa seluruh modul aplikasi berbasis kerangka kerja Django dapat beroperasi sesuai dengan kebutuhan pengguna. Pengujian difokuskan pada kelancaran fungsi-fungsi utama, mulai dari fungsi impor data komentar YouTube melalui dokumen CSV, pemrosesan awal teks (*preprocessing*), hingga ketepatan visualisasi grafik sentimen secara granular (Permatasari et al., 2023). Melalui implementasi pengujian yang terperinci ini, tingkat keandalan, kegunaan, dan kualitas fungsionalitas antarmuka perangkat lunak dapat diukur secara optimal serta bebas dari ambiguitas operasional.

B. Kajian empiris

Banyak penelitian sebelumnya telah menguji penggunaan metode analisis teks untuk mengolah opini publik di media sosial secara terukur. Studi eksperimental yang dilakukan oleh Radiena dan Nugroho membuktikan bahwa implementasi *Aspect-Based Sentiment Analysis* (ABSA) yang dikombinasikan dengan algoritma *Support Vector Machine* (SVM) mampu memproses data teks berdimensi tinggi dengan tingkat akurasi yang konstan. Evaluasi kuantitatif dari penelitian-penelitian tersebut menunjukkan bahwa SVM mampu mencapai tingkat akurasi hingga 87,5% dalam memisahkan opini majemuk, di mana satu kalimat yang sama mengandung polaritas nilai sentimen yang berlawanan pada entitas aspek yang berbeda. Kapabilitas SVM dalam menghitung batas ruang kelas (*hyperplane*) dengan margin maksimal terbukti secara statistik mampu menekan persentase error klasifikasi saat dihadapkan pada struktur data opini warganet yang memiliki distribusi kata tidak seimbang (*imbalanced data*).

Penerapan arsitektur analisis sentimen ini memiliki urgensi komputasional ketika dihadapkan pada isu-isu geopolitik yang memicu polarisasi publik secara tajam. Penelitian yang dilakukan oleh Ardhi (2023) membuktikan bahwa platform media sosial, khususnya kolom komentar YouTube, menyediakan korpus data primer berskala masif, mencapai ribuan hingga jutaan token kata terkait konflik internasional seperti di Timur Tengah. Berdasarkan distribusi frekuensinya, karakteristik diskusi publik pada isu ini terpecah ke dalam kelas-kelas spesifik yang saling beririsan. Oleh karena itu,

penerapan dekomposisi tingkat aspek (ABSA) digunakan sebagai landasan pemodelan agar sentimen warganet tersebut dapat dipetakan secara terpisah ke dalam variabel ekonomi, kemanusiaan, maupun politik, sehingga meminimalisir bias akurasi yang terjadi apabila analisis hanya direduksi menjadi satu nilai polaritas tunggal.

Lebih lanjut, dari segi arsitektur sistem dan metodologi perolehan data, kerangka kerja *Django* memiliki kapabilitas konkurensi yang stabil untuk menangani beban eksekusi *machine learning* berbasis *Python* di sisi *backend* tanpa memicu *time-out server* (Pramudita, 2023). Meskipun penelitian sebelumnya mengandalkan *web scraping* otomatis secara *real-time* untuk akuisisi data komentar YouTube, metode tersebut terbukti mencatatkan tingkat kegagalan (*failure rate*) penarikan data yang tinggi akibat pembatasan akses API (*rate limiting failure rate*) serta perubahan struktur HTML dinamis dari pihak penyedia platform. Merespons anomali teknis tersebut, perancangan sistem ini mengadopsi ekstraksi data eksternal melalui mekanisme impor berkas format CSV ke dalam basis data sistem. Pendekatan arsitektur ini secara kuantitatif mampu menjamin integritas 100% korpus data awal terhindar dari *data loss* sebelum disalurkan ke dalam alur pembersihan teks (*text preprocessing*) dan mesin prediksi SVM. Arsitektur data tertutup ini sekaligus menggaransi stabilitas fungsionalitas antarmuka merespons input/output pengguna tanpa memicu galat (*error*) saat divalidasi melalui pengujian *Blackbox* (Permatasari et al., 2023).

C. Kerangka Berpikir

Kerangka berpikir dalam penelitian ini diawali dengan penetapan batasan masalah untuk memfokuskan ruang lingkup pengembangan sistem. Batasan utama mencakup sumber data yang berasal dari opini warganet Indonesia di platform YouTube mengenai konflik Timur Tengah, di mana perolehan data dilakukan di luar sistem menggunakan *web scraping* dan diunggah melalui dokumen berformat CSV. Selain itu, proses analisis difokuskan pada penerapan pendekatan analisis sentimen berbasis aspek yang dibatasi pada tiga kategori, yaitu kemanusiaan, ekonomi, dan politik. Algoritma pembelajaran mesin yang digunakan secara spesifik adalah *Support Vector Machine* (SVM) untuk mengklasifikasikan polaritas sentimen pada masing-masing aspek tersebut.

Model penelitian yang diterapkan berpusat pada perancangan arsitektur pemrosesan bahasa alami atau *Natural Language Processing* (NLP). Model ini dirancang untuk tidak hanya memproses teks secara utuh, tetapi juga mendekomposisinya menjadi unit-unit informasi yang lebih spesifik. Pendekatan komputasi teks ini diformulasikan untuk mengekstraksi aspek pembicaraan terlebih dahulu, lalu menentukan arah sentimen yang ditujukan pada objek tersebut. Pemodelan ini memastikan bahwa kalimat majemuk dari warganet yang memiliki nuansa polaritas saling bertolak belakang dapat dipetakan secara akurat tanpa kehilangan konteks aslinya.

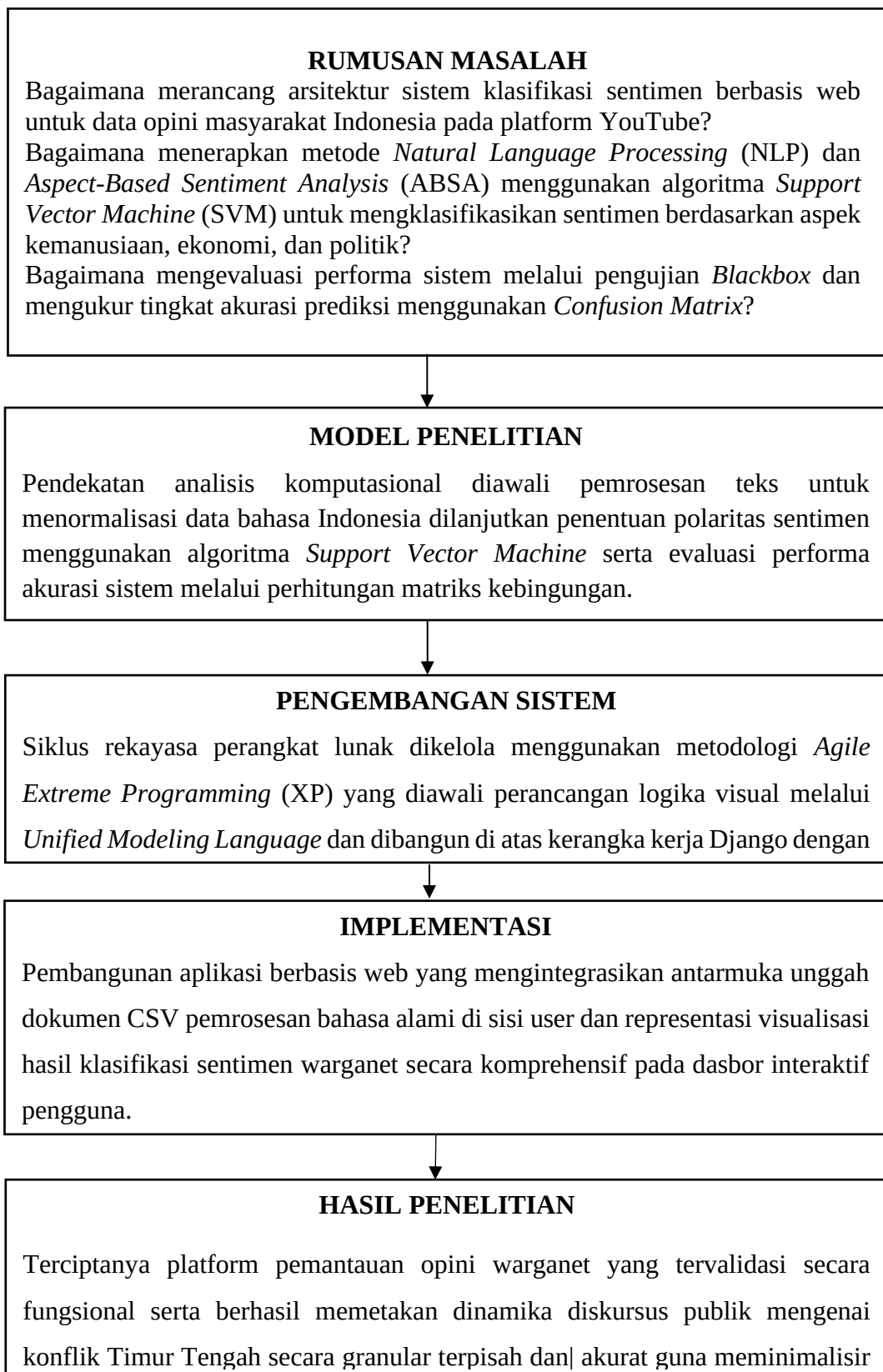
Pada tahap pengembangan sistem, metodologi *Agile* digunakan sebagai panduan rekayasa perangkat lunak agar proses pembangunan aplikasi

dapat berjalan secara adaptif dan iteratif. Sistem ini dibangun di atas kerangka kerja Django yang mengadopsi pola arsitektur *Model-View-Template* (MVT). Penerapan kerangka kerja ini dirancang untuk mengisolasi logika pengelolaan basis data masukan, pemrosesan algoritma di sisi pengontrol, dan visualisasi antarmuka pengguna, sehingga sistem tetap stabil dan aman saat menjalankan komputasi penambahan teks.

Tahap implementasi merupakan realisasi dari perancangan arsitektur perangkat lunak ke dalam bentuk operasional fungsional. Proses alur kerja dimulai ketika pengguna mengunggah dokumen CSV berisi korpus komentar mentah ke dalam sistem. Data teks tersebut selanjutnya dieksekusi melalui modul prapemrosesan otomatis yang meliputi penyeragaman huruf, pemotongan kalimat, penghilangan kata hubung, hingga standardisasi ejaan. Teks yang telah terstruktur secara matematis kemudian diproses oleh algoritma SVM untuk diklasifikasikan ke dalam kelas sentimen berdasarkan bobot vektornya, yang kemudian hasilnya disimpan di dalam basis data untuk disalurkan ke layar pengguna.

Hasil penelitian bermuara pada pembuktian keberhasilan rancang bangun sistem melalui serangkaian luaran dan evaluasi komprehensif. Luaran dari sistem ini berupa perangkat lunak berbasis web yang menyediakan dasbor visualisasi grafik sentimen publik secara granular berdasarkan pembagian aspek. Evaluasi kualitas sistem dibuktikan melalui pengukuran kinerja algoritma SVM menggunakan instrumen matriks kebingungan untuk mengetahui tingkat akurasi presisi klasifikasi, serta pelaksanaan pengujian

Blackbox yang memvalidasi bahwa seluruh fitur antarmuka, mulai dari unggah berkas hingga tayangan hasil, beroperasi sesuai dengan rancangan spesifikasi. Lebih jauh lagi, integrasi antara hasil pengujian algoritma dan validasi antarmuka ini akan memastikan bahwa perangkat lunak yang dihasilkan tidak hanya beroperasi secara fungsional, tetapi juga memiliki keandalan analitik yang dapat dipertanggungjawabkan secara saintifik. Representasi visual dari dekomposisi sentimen tersebut pada akhirnya akan bertransformasi menjadi instrumen pemantauan yang objektif guna meminimalisir bias interpretasi, serta memberikan wawasan strategis yang konkret terkait dinamika diskursus masyarakat Indonesia



Gambar 2. 9 Kerangka Berpikir