UNIVERSITAS GUNADARMA

FAKULTAS ILMU KOMPUTER & TEKNOLOGI INFORMASI

PERBANDINGAN METODE NAIVE BAYES DAN SUPPORT VECTOR MACHINE PADA ANALISIS SENTIMEN ULASAN APLIKASI ACCESS BY KAI DI GOOGLE PLAY STORE

Disusun Oleh :

Nama	**: Dhasa Satria Nugraha**
NPM	**: 10120301**
Program Studi	**: Sistem Informasi**
Pembimbing	**: Dr. Aviarini Indrati, SKom., MMSI., MIKom.**

Diajukan Guna Melengkapi Sebagian Syarat

Dalam Mencapai Gelar Sarjana Strata Satu (S1)

JAKARTA

2024

PERNYATAAN ORISINALITAS DAN PUBLIKASI

Saya yang bertanda tangan di bawah ini,

Nama	: Dhasa Satria Nugraha
NPM	: 10120301
Judul Tulisan Ilmiah	: Perbandingan Metode Naive Bayes dan Support Vector Machine Pada Analisis Sentimen Ulasan Aplikasi Access by KAI di Google Play Store
Tanggal Sidang	: 26-Agustus-2024
Tanggal Lulus	: 26-Agustus-2024

Menyatakan bahwa tulisan ini merupakan hasil karya saya sendiri dan dapat dipublikasikan sepenuhnya oleh Universitas Gunadarma. Segala kutipan dalam bentuk apa pun telah mengikuti kaidah, etika yang berlaku. Mengenai isi dan tulisan adalah merupakan tanggung jawab penulis, bukan Universitas Gunadarma.

Demikian pernyataan ini dibuat dengan sebenarnya dan dengan penuh kesadaran.

Jakarta, 16 Agustus 2024

Dhasa Satria Nugraha

LEMBAR PENGESAHAN

KOMISI PEMBIMBING

NO.	NAMA	KEDUDUKAN
1.	Dr. Aviarini Indrati, SKom., MMSI., MIKom.	Ketua
2.	Dr. Dyah Cita Irawati, SSi., MM.	Anggots
3.	Dr. Yuti Dewita Arimbi, ST., MMSI	Anggota

Tanggal Sidang : 26/08/2024

PANITIA UJIAN

NO.	NAMA	KEDUDUKAN
1.	Dr. Ravi Ahmad Salim	Ketua
2.	Prof. Dr. Wahyudi Priyono	Sekretaris
3.	Dr. Aviarini Indrati, SKom., MMSI., MIKom.	Anggota
4.	Dr. Dyah Cita Irawati, SSi., MM.	Anggota
5.	Dr. Yuti Dewita Arimbi, ST., MMSI	Anggota

Tanggal Lulus : 26/08/2024

Mengetahui,

Pembimbing **Bagian Sidang Ujian**

(Dr. Aviarini Indrati, SKom., MMSI., MIKom.) **(Dr. Edi Sukirman, SSi., MM., M.I.Kom.)**

ABSTRAK

Dhasa Satria Nugraha. 10120301
PERBANDINGAN METODE NAIVE BAYES DAN SUPPORT VECTOR MACHINE PADA ANALISIS SENTIMEN ULASAN APLIKASI ACCESS BY KAI DI GOOGLE PLAY STORE
Skripsi. Jurusan Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma, 2024.
Kata kunci: Naive Bayes, Perbandingan, Sentimen, *Support Vector Machine*
(xiii + 101 + Lampiran)

Analisis sentimen merupakan suatu proses pengumpulan, pengolahan, dan analisis data untuk menentukan sentimen atau opini yang terkandung dalam teks atau data. Proses analisis sentimen dapat dilakukan dengan menggunakan beberapa pendekatan yaitu pendekatan berbasis aturan, pendekatan *machine learning* dan pendekatan *neural network.* Pendekatan *machine learning* memiliki dua teknik yang digunakan yaitu *Naive Bayes* dan *Support Vector Machine* kedua metode tersebut memiliki kelebihan dan kekurangannya tersendiri dalam analisis sentimen. Penelitian ini bertujuan untuk membandingkan kinerja antara *Support Vector Machine* dan *Naive Bayes* dalam menganalisis sentimen ulasan aplikasi Access by KAI, berdasarkan nilai evaluasi yaitu akurasi, *precision, recall,* dan *F1-Score*. Metode yang digunakan untuk memproses analisis sentimen dilakukan melalui beberapa tahapan, yaitu akuisisi data ulasan aplikasi Access by KAI di Google Play Store, *preprocessing* data, pemodelan data menggunakan *Support Vector Machine* dan *Naive Bayes,* serta evaluasi kinerja dengan menggunakan *confusion matrix, dan* melakukan perbandingan kinerja berdasarkan nilai evaluasi yang telah dihasilkan. Hasil dari penelitian ini menghasilkan nilai dari evaluasi kinerja metode *Support Vector Machine* pada perbandingan 60 : 40 adalah 81,4% dan pada metode *Naive Bayes* adalah 74,8%, pada perbandingan 70 : 30 *Support Vector Machine* menghasilkan akurasi 84,2 dan *Naive Bayes* 77%, sedangkan pada perbandingan 80 : 20 *Support Vector Machine* menghasilkan akurasi 85,5 dan *Naive Bayes* menghasilkan akurasi 78,1%. Selain itu berdasarkan hasil *confusion matrix* yang telah dilakukan yang menghasilkan nilai *precision,* nilai *recall,* dan nilai *F1-Score* menunjukkan hasil metode *Support Vector Machine* lebih unggul dibandingkan dengan metode *Naive Bayes.* Berdasarkan hasil tersebut maka menunjukkan metode *Support Vector Machine* lebih unggul dalam melakukan analisis sentimen terhadap ulasan Access by KAI dibandingkan dengan metode *Naive Bayes.*

Daftar Pustaka (2019-2024)

ABSTRACT

Dhasa Satria Nugraha. 10120301
PERBANDINGAN METODE NAIVE BAYES DAN SUPPORT VECTOR MACHINE PADA ANALISIS SENTIMEN ULASAN APLIKASI ACCESS BY KAI DI GOOGLE PLAY STORE
Thesis. Department of Information Systems, Faculty of Computer Science and Information Technology, Gunadarma University, 2024.
Keywords: Comparison, Naive Bayes, Sentiment, Support Vector Machine
(xii + 101 + attachment)

Sentiment analysis is a process of collecting, processing, and analyzing data to determine the sentiment or opinion contained within text or data. Sentiment analysis can be conducted using various approaches, including rule-based methods, machine learning approaches, and neural network approaches. Machine learning approaches often utilize two techniques: Naive Bayes and Support Vector Machine, each with its own strengths and weaknesses in sentiment analysis. This study aims to compare the performance of Support Vector Machine and Naive Bayes in analyzing the sentiment of reviews for the Access by KAI application, based on evaluation metrics such as accuracy, precision, recall, and F1-Score. The sentiment analysis process was conducted in several stages: acquiring review data for the Access by KAI application from the Google Play Store, preprocessing the data, modeling the data using Support Vector Machine and Naive Bayes, and performing a confusion matrix to obtain evaluation metrics for both methods, followed by a performance comparison. The results of this study indicate that the performance evaluation of the Support Vector Machine method yielded an accuracy of 81.4% with a 60:40 data split, compared to 74.8% for Naive Bayes. With a 70:30 data split, Support Vector Machine achieved an accuracy of 84.2% compared to 77% for Naive Bayes, while a split of 80:20 saw Support Vector Machine achieving 85.5% accuracy compared to 78.1% for Naive Bayes. Additionally, based on the confusion matrix, which provided precision, recall, and F1-Score metrics, the Support Vector Machine method outperformed the Naive Bayes method. Therefore, the results suggest that the Support Vector Machine method is more effective in performing sentiment analysis on reviews for the Access by KAI application compared to the Naive Bayes method.

Bibiliography (2019-2024)

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan berkat, anugerah dan karunia yang melimpah, sehingga penulis dapat menyelesaikan tugas akhir ini.

Tugas akhir ini disusun guna melengkapi Sebagian syarat dalam mencapai gelar Setara Sarjana Strata Satu pada jurusan Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma. Adapun judul skripsi ini adalah "Perbandingan Metode Naive Bayes dan Support Vector Machine Pada Analisis Sentimen Ulasan Aplikasi Access by KAI di Google Play Store".

Walaupun banyak kesulitan yang penulis harus hadapi ketika menyusun tugas akhir ini, namun berkat bantuan dan dorongan dari berbagai pihak akhirnya tugas ini dapat diselesaikan dengan baik. Untuk itu penulis mengucapkan terima kasih, kepada:

1. Prof. Dr. E.S. Margianti, SE., MM., selaku Rektor Universitas Gunadarma.
2. Prof. Dr. Rer. Nat. Achmad Benny Mutiara, SSi.., SKom., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Gunadarma.
3. Dr. Setia Wirawan, SKom., MMSI., selaku Ketua Program Studi Sistem Informasi Universitas Gunadarma.
4. Dr. Edi Sukirman, SSi., MM., MIKom., selaku Kepala Bagian Sidang Ujian Universitas Gunadarma.
5. Dr. Aviarini Indrati, SKom., MMSI., MIKom., selaku Dosen Pembimbing yang telah memberikan arahan dan waktunya kepada penulis selama penulisan ini berlangsung hingga selesai.
6. Bapak dan Ibu Dosen Universitas Gunadarma yang telah memberikan ilmu pengetahuan kepada penulis.
7. Kedua orang tua, Hayu Oprasito dan Tri Sumarningsih, yang selalu memberikan kasih sayang, doa, nasehat, serta kesabarannya yang luar biasa dalam setiap

langkah hidup penulis, yang merupakan anugrah terbesar dalam hidup. Penulis berharap dapat menjadi anak yang dapat dibanggakan.

8. Teman-teman kelas 4KA03 yang selalu memberikan dukungan serta arahan kepada penulis.
9. Semua pihak yang telah membantu penulis namun tidak dapat disebutkan satu persatu.

Jakarta, 16 Agustus 2024

Dhasa Satria Nugraha

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

1. PENDAHULUAN

1.1. Latar Belakang

Dalam Beberapa tahun terakhir, penggunaan aplikasi *mobile* telah menjadi bagian yang tidak terpisahkan dari kehidupan sehari-hari. Aplikasi *mobile* memberikan kemudahan bagi pengguna untuk melakukan berbagai aktivitas, mulai dari memesan makanan, melakukan perjalanan, hingga berbelanja secara *online*, dengan mudah dan cepat melalui perangkat mereka. Google Play Store adalah salah satu *platform* utama yang menyediakan berbagai macam aplikasi untuk diunduh oleh pengguna, serta memberikan kesempatan bagi pengguna untuk berbagi pengalaman mereka melalui ulasan yang memang disediakan oleh *platform* Google Play store. Melalui ulasan ini, pengguna dapat memberikan masukan, saran, pujian, atau bahkan keluhan terhadap aplikasi yang mereka gunakan. Ulasan pengguna mengenai aplikasi berguna sebagai masukan terhadap pengembang aplikasi dalam melakukan pembaharuan selanjutnya. Ketika terdapat banyak data dan bervariasi, maka pengolahan data menjadi rumit dan memakan waktu cukup lama. Oleh karena itu, analisis sentimen diperlukan untuk menentukan lebih banyak ulasan positif atau ulasan negatif.

Analisis sentimen merupakan suatu proses pengumpulan, pengolahan, dan analisis data untuk menentukan sentimen atau opini yang terkandung dalam teks atau data. Pendekatan ini memanfaatkan teknik-teknik *Natural Language Processing* (NLP) untuk mengidentifikasikan dan memahami emosi atau pendapat yang tersembunyi dalam suatu teks. Proses analisis sentimen dapat dilakukan pada berbagai jenis data, termasuk teks, gambar, dan suara. Dalam praktiknya, terdapat beberapa pendekatan yang dapat digunakan untuk melakukan analisis sentimen, yaitu menggunakan pendekatan berbasis aturan, pendekatan *Machine Learning* dan pendekatan *Neural Network.*

Pada pendekatan berbasis aturan, terdapat aturan-aturan linguistik atau statistik digunakan untuk menglasifikasikan teks berdasarkan sentimen yang terkandung di dalamnya, melibatkan penggunaan leksikon, tokenisasi, dan teknik

parsing. Pendekatan *Machine Learning* merupakan pendekatan model-model pembelajaran mesin dipelajari dari data latih untuk mengklasifikasikan sentimen pada teks yang belum pernah dilihat sebelumnya, menggunakan teknik seperti *Naive Bayes, Support Vector Machines*. Pendekatan terakhir adalah pendekatan *Neural Network,* yang menggunakan jaringan saraf tiruan untuk memahami struktur dan makna teks, pendekatan ini menggunakan model seperti *Recurrent Neurak Network* (RNN) dan *Long Short-Term Memory* (LSTM). Setiap pendekatan memiliki kelebihan dan kekurangan tersendiri, dan pemilihan pendekatan yang tepat tergantung pada karakteristik data dan kompleksitas masalah yang dihadapi.

Berdasarkan karakteristik data yang digunakan yaitu data memiliki jumlah data yang relatif tidak besar, data yang seimbang antara ulasan positif dan negatif, terdapat pemisahan antara data latih dan data uji. Berdasarkan karakteristik tersebut pendekatan yang sesuai adalah pendekatan *Machine Learning,* pendekatan *Machine Learning* memiliki keunggulan pada kecepatan pelatihan karena memiliki struktur yang lebih sederhana dan tidak memerlukan jumlah parameter yang sangat besar, dapat dengan mudah memperbaharui model dengan data latih yang dimiliki, serta dapat memberikan hasil yang baik bahkan dengan jumlah data yang relatif sedikit (Ilmawan and Mude, 2020).

Pendekatan *Machine Learning* memiliki dua teknik yang umum digunakan yaitu *Naive Bayes* dan *Support Vector Machine,* kedua metode ini memiliki kelebihan dan kekurangannya masing-masing. Metode *Naive Bayes* merupakan klasifikasi yang paling sederhana dan paling umum digunakan dalam analisis sentimen. Secara prinsip, *Naive Bayes* menghitung probabilitas kelas berdasarkan distribusi kata-kata yang terdapat dalam data. Keunggulan utama dari metode *Naive Bayes* adalah kesederhanaannya, kecepatan komputasi yang tinggi, dan akurasi yang baik. Namun, metode ini memiliki keterbatasan utama, yaitu asumsi bahwa setiap kata dalam data adalah independen, asumsi tersebut sering kali tidak terpenuhi dalam praktiknya sehingga mempengaruhi tingkat akurasinya (Md et al., 2024)

Sedangkan metode *Support Vector Machine* telah menjadi salah satu metode klasifikasi dan regresi yang populer, terutama untuk masalah linear dan nonlinear.

Salah satu Keistimewaan dari *Support Vector Machine* berasal dari kemampuan untuk menerapkan pemisahan linear pada *input* data non linear dengan dimensi yang tinggi. Hal ini dicapai dengan menggunakan fungsi kernel yang diperlukan untuk mentransformasikan data ke dalam ruang fitur yang lebih tinggi. Keunggulan utama *Support Vector Machine* adalah kemampuannya untuk membedakan antara kelas positif dan negatif dengan lebih baik, terutama ketika data negatif memiliki karakteristik yang lebih beragam daripada data positif, SVM menggunakan teknik kernel yang mampu memetakan data ke dalam ruang dimensi yang lebih tinggi, sehingga dapat memodelkan hubungan antar data dengan lebih baik untuk mendapatkan nilai akurasi yang lebih tinggi (Rokhman et al., 2021).

Meskipun demikian, *Support Vector Machine* juga memiliki keterbatasan. Salah satunya adalah kinerjanya yang dapat menurun ketika kumpulan data memiliki banyak gangguan, seperti adanya tumpang tindih antara kelas target atau distribusi data yang tidak merata. Hal ini dapat mengakibatkan penurunan akurasi dan stabilitas klasifikasi.

Kedua metode, *Naive Bayes* dan *Support Vector Machine,* merupakan metode yang umum digunakan dalam analisis sentimen dan merupakan metode yang tergabung dalam pendekatan *Machine Learning*. Setiap metode memiliki kekurangan dan kelebihannya masing-masing, sehingga keduanya akan memberikan nilai akurasi yang berbeda. Dengan menggunakan ulasan pengguna, dapat dilakukan analisis sentimen untuk mencari nilai akurasi dari kedua metode. Selain itu, juga untuk melakukan perbandingan kinerja metode dengan menggunakan ukuran kinerja seperti akurasi, *precision, recall*, dan F1-S*core*. Hal ini dilakukan untuk mengetahui metode yang memiliki tingkat kinerja tertinggi pada *dataset* yang sama.

Dalam melakukan perbandingan kedua metode tersebut, dibutuhkan sebuah objek, yaitu sebuah aplikasi yang terdapat di Google Play Store dengan kriteria aplikasi tersebut memiliki jumlah pengguna yang banyak, mendapatkan banyak ulasan dari pengguna, dikelola oleh pengembang yang aktif dalam melakukan pembaharuan, namun memiliki *rating* yang tidak memuaskan. Salah satu aplikasi yang memenuhi kriteria tersebut adalah Access by KAI. Aplikasi ini telah diunduh

lebih dari sepuluh juta unduhan, dengan *rating* 2.2, dan telah menerima lebih dari 204 ribu ulasan dari pengguna, dan pembaharuan terakhir dilakukan pada tanggal 19 Maret 2024.

Access by KAI adalah aplikasi yang dikembangkan oleh PT Kereta Api Indonesia. Aplikasi ini diluncurkan pada tanggal 4 September 2014 dengan nama KAI Access yang memiliki 8 fitur utama, yaitu profil, berita, pesan tiket, jadwal, cek *booking*, riwayat, peta lokasi, dan tentang. KAI Acces mengalami pembaharuan pertama pada September 2016 yaitu dengan melakukan perubahan tampilan dan perubahan terhadap versi android yang dapat mengunduh aplikasi tersebut. Seiring berjalannya waktu pembaharuan terus dilakukan pada tahun 2017 terdapat 3 pembaharuan besar yang dilakukan yaitu melakukan penambahan fitur baru seperti e-moda, porter, makanan, serta *chat with* loko, pembaharuan selanjutnya pada bulan Oktober di tahun yang sama yaitu menambah fitur *boarding pass* elektronik serta terdapat fitur pembatalan, pengubahan jadwal, pembelian tiket kereta lokal, dan dompet elektronik. Pada tahun 2019 terdapat perubahan yang cukup banyak pada aplikasi KAI Acces, fitur-fitur yang dimiliki oleh KAI Access semakin banyak salah satunya yang terbaru yaitu seluruh tiket kereta lokal sudah dapat dipesan melalui aplikasi tersebut, pembayaran melalui QRIS, serta uang elektronik bernama KAIPAY yang dapat digunakan untuk melakukan pembayaran tiket kereta.

Pembaharuan terakhir sekaligus membuat KAI Access berganti nama menjadi Access by KAI, perilisan aplikasi Access by KAI dilakukan pada tanggal 10 Agustus 2023. Pembaharuan yang terdapat pada aplikasi Acces by KAI tidak hanya penambahan fitur tetapi mengubah seluruh *user interface* dengan tampilan yang lebih segar dan *youthful*. Dalam Access by KAI, semua layanan yang dioperasikan oleh KAI diintegrasikan dalam satu sistem pemesanan tiket, yaitu terdiri dari LRT, KA Bandara, Commuter Line, dan whoosh. Penambahan fitur yang sangat berbeda dari pembaharuan-pembaharuan sebelumnya yaitu adanya pendaftaran sistem pengenalan wajah, sehingga calon penumpang kereta api dapat melakukan pendaftaran wajah melalui aplikasi Access by KAI, dengan melakukan pendaftaran tersebut maka pada saat penumpang sudah berada di stasiun tidak perlu lagi mencetak *boarding pass* dan dapat langsung melewati *gate* yang terdapat

sensor pemindai wajah. Selain itu fitur baru yang ditambahkan pada *rebranding* tersebut adalah transfer tiket, *single sign-on*, pembelian pulsa, paket data, *payment point online* bank, token PLN, serta *railfood*.

Pengguna aplikasi Access by KAI pada akhir tahun 2023 tercatat sebanyak 12.419.711 *register user* dengan jumlah pengguna aktif sebanyak 6.101.343. Dengan banyaknya pengguna aplikasi Access by KAI, dan begitu sering terjadi pembaharuan terhadap aplikasi tersebut maka terdapat banyak sekali ulasan yang ditulis oleh pengguna pada kolom ulasan Google Play Store.

Penelitian tentang analisis sentimen telah dilakukan pada penelitian sebelumnya dengan beberapa metode yang berbeda-beda, seperti pada penelitian yang dilakukan oleh Ayu Sri Rahayu, Ahmad Fauzi, dan Rahmat (2022) dengan menggunakan metode *Support Vector Machine* dan *Naive Bayes* pada aplikasi Spotify mendapatkan hasil untuk SVM menghasilkan nilai akurasi 84% sedangkan *Naive Bayes* menghasilkan nilai akurasi 86,4%. Pada penelitian yang dilakukan oleh Meishita Inelza Putri dan Iqbal Kharisudin (2022) dengan aplikasi Tokopedia menggunakan metode *Support Vector Machine (SVM), Naive Bayes*, dan *Logistic Regression* mendapatkan hasil Secara umum *review* pengguna lebih banyak bersentimen positif yang berkaitan dengan kemudahan penggunaan aplikasi tersebut, sedangkan review negatif berkaitan dengan sistem aplikasi *error* dan sering terjadi *lagging/bug*. Penelitian yang dilakukan oleh Lutfi Budi Ilmawan dan Muhmmad Aliyazid Mude (2020) dengan metode *Support Vector Machine* (SVM) dan *Naive Bayes* pada aplikasi ulasan tekstual mendapatkan nilai akurasi untuk *Support Vector Machine* 81,46% dan *Naive Bayes* 75,41%.

Berdasarkan latar belakang tersebut maka rumusan masalah dalam penelitian ini yaitu bagaimana melakukan analisis sentimen terhadap ulasan aplikasi Access by KAI menggunakan metode *Support Vector Machine* dan *Naive Bayes*?, serta bagaimana perbandingan kinerja kedua metode tersebut?.

1.2. Ruang Lingkup

Ruang lingkup Penelitian ini mencakup kinerja metode dalam menghasilkan nilai akurasi. Membandingkan nilai akurasi, *precision, recall,* dan F1-*Score.*

Dataset yang digunakan merupakan ulasan aplikasi Access by KAI yang diambil dari Google Play Store menggunakan *Python Scrapper*. Ulasan yang digunakan adalah ulasan yang diberikan oleh pengguna Access by KAI mulai dari tanggal 10 Agustus 2023 hingga 13 Maret 2024. Ulasan yang diambil merupakan ulasan yang menggunakan Bahasa Indonesia. Pengolahan data akan dilakukan menggunakan bahasa pemrograman *Python* melalui *platform* Google Colaboratory.

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk melakukan analisis sentimen terhadap ulasan Access by KAI menggunakan metode *Support Vector Machine* dan *Naive Bayes*, serta untuk melakukan perbandingan kinerja kedua metode tersebut.

1.4. Sistematika Penulisan

Sistematika penulisan pada penelitian ini secara garis besar terbagi menjadi lima bab, bertujuan untuk mempermudah penulisan dalam menyusun, mengolah dan merangkai data sehingga penulisan ini dapat disajikan dengan baik dan benar. Penelitian dimulai pada bab satu yaitu pendahuluan yang berisikan latar belakang, ruang lingkup, tujuan penelitian, dan sistematika penulisan. Dilanjutkan dengan bab dua yaitu tinjauan pustaka yang memuat uraian mengenai landasan teori dan landasan empiris yang mendukung pendekatan pemecahan masalah. Setelah itu dilanjutkan dengan bab tiga yaitu metode penelitian yang menguraikan tentang cara kerja serta teknik atau proses pengerjaan. Selanjutnya bab empat yaitu hasil dan pembahasan, pada bab ini berisi tentang hasil yang telah diperoleh dan analisis terhadap hasil yang diperoleh untuk menjawab tujuan penelitian. Bagian terakhir dari penulisan ini yaitu bab lima atau penutup yang terdiri dari dua bagian yaitu kesimpulan dan saran.

2. TINJAUAN PUSTAKA

2.1. *Text Mining*

Text mining adalah satu langkah dari analisis teks yang dilakukan secara otomatis oleh komputer untuk menggali informasi yang berkualitas dari suatu rangkaian teks yang terangkum dalam sebuah dokumen yang merupakan variasi dari data *mining* untuk berusaha menemukan pola dari kata tersebut yang menarik dari sekumpulan data tekstual yang berjumlah besar, proses *text mining* ini sendiri merupakan hal berdasarkan kombinasi yang baik serta kata, dan frase yang sesuai, serta melibatkan proses ekstraksi fitur yang menjadi kunci efektif utama dalam tahap pelatihan atau yang sering disebut dengan *training,* selain itu *text mining* atau *text analytics* adalah istilah yang mendeskripsikan sebuah teknologi yang mampu menganalisis data teks semi-terstruktur maupun tidak terstruktur, hal inilah yang membedakannya dengan data *mining* di mana data *mining* mengolah data yang sifatnya terstruktur, *text mining* membuat asosiasikan satu bagian *text* dengan lainnya berdasarkan aturan tertentu. Hasil yang diharapkan adalah kata baru yang tidak terungkap jelas sebelumnya (Purnajaya et al., 2022).

Text mining merupakan suatu penemuan dari pengetahuan di dalam dokumen teks yang dapat mengungkapkan wawasan berharga dari teks-teks yang ada. Menggunakan *text mining* merupakan suatu tantangan dalam mendapatkan pengetahuan yang akurat pada dokumen teks dalam membantu pengguna menemukan manfaat dari teks-teks yang ada tersebut. Penemuan pengetahuan dapat menjadi efektif digunakan dan memperbaharui pola penemuan, lalu menerapkan penemuan tersebut ke dalam *text mining.* Inti dari proses ini adalah menggabungkan informasi yang berhasil diekstraksi dari berbagai sumber untuk mendapatkan informasi yang dapat dimanfaatkan. Proses *text mining* yang khas meliputi kategorisasi teks, *text clustering*, ekstraksi konsep/entitas, produksi taksonomi granular, *sentiment analysis,* penyimpulan dokumen, dan pemodelan relasi entitas, dalam *text mining* salah satu metode umum untuk memvisualisasikan data adalah

dengan menggunakan *wordcloud* yang akan menampilkan kata-kata yang sering muncul dalam suatu teks (Putri Gabriella, 2023).

Pengumpulan data yang terbentuk menjadi sebuah data set dilakukan dengan menggunakan metode Web *Scrapping*, yaitu metode yang mengacu pada operasi teknis ekstraksi informasi secara otomatis dari sumber *online.* Teknik ini digunakan untuk mengumpulkan data terkait penelitian digital dalam bentuk yang terstruktur (Nishfi et al., 2023).

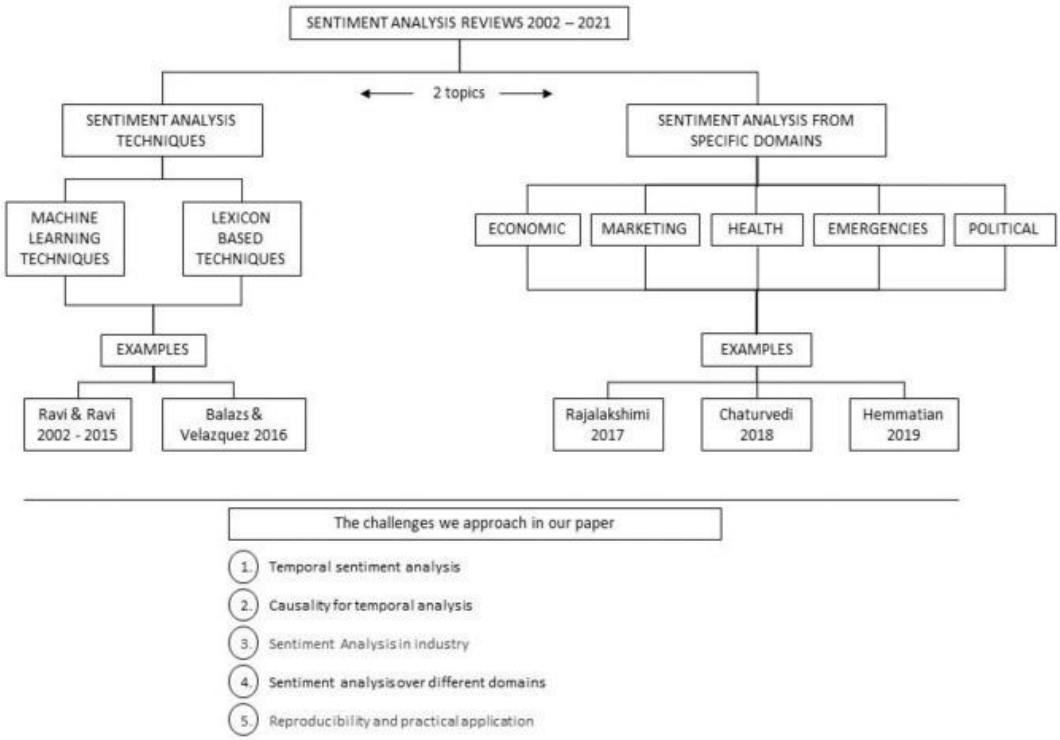
2.2. Analisis Sentimen

Analisis sentimen biasa disebut *opinion mining* adalah salah satu bagian dari *text mining*. *Text mining* melakukan studi tentang pendapat yang muncul dari orang-orang, sentimen, evaluasi, tingkah laku dan emosi terhadap suatu entitas seperti produk, layanan, organisasi, individu, permasalahan, topik, acar. Analisis sentimen biasa diterapkan untuk menganalisis komentar-komentar di sosial media seperti Facebook, Twitter untuk selanjutnya dilakukan proses penerjemahan menjadi sesuatu yang lebih bermakna, salah satunya dalam bentuk *rating*. *Rating* menjadi sangat penting dalam dunia bisnis disebabkan *rating* merupakan salah satu indikator kesuksesan. Di sisi lain, *rating* masih menjadi komoditas monopoli beberapa perusahaan seperti Nielsen, sehingga objektivitasnya menjadi kurang (Agung Pramana and Ramdhani, 2023).

Analisis sentimen dapat didefinisikan sebagai metode pemrosesan dalam memperoleh informasi opini positif atau opini negatif yang berasal dari media sosial, situs jual beli, dan laman unduhan aplikasi. Analisis sentimen merupakan metode yang digunakan untuk memperoleh informasi dari berbagai *platform* media sosial termasuk Google Play Store, analisis sentimen memiliki tujuan untuk menganalisis pendapat, sentimen, evaluasi, sikap, dan emosi seseorang dari bahasa tulisan, dan secara otomatis mendeskripsikan emosi tersebut dan mengelompokkannya menjadi sentimen positif dan sentimen negatif, hal tersebut dapat bermanfaat dan dapat dijadikan sebagai acuan dalam meningkatkan suatu pelayanan, ataupun kualitas tertentu. Analisis sentimen diterapkan dalam berbagai aspek, seperti prediksi harga saham, isu politik, dan kepuasan terhadap suatu produk atau layanan, dengan

analisis sentimen maka pendapat dari pengguna dapat dipahami dan dapat dijadikan sebagai bahan untuk melakukan evaluasi (Miftahusalam et al., 2023).

Analisis sentimen mengalami perkembangan yang begitu pesat, perkembangan yang terjadi tidak hanya pada metode tetapi juga pada objek yang dapat dilakukan analisis sentimen, saat ini berbagai macam objek dapat menggunakan teknik ini untuk mencari sentimen yang ada pada teks-teks tersebut. Besarnya pengaruh dan manfaat dari analisis sentimen menyebabkan penelitian dan aplikasi berbasis analisis sentimen berkembang pesat. Bahkan di Amerika terdapat sekitar 20-30 perusahaan yang memfokuskan pada layanan analisis sentimen (Nurtikasari et al., 2022).

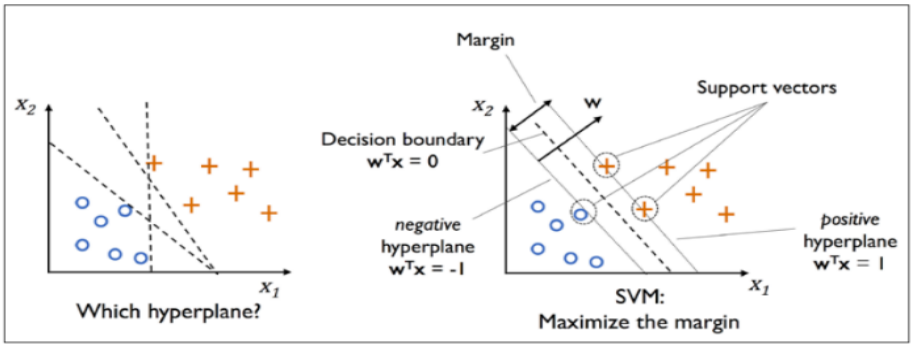


Gambar 2. 1 Jenis Analisis Sentimen

2.3. *Support Vector Machine*

Support Vector Machine adalah algoritma *machine learning* yang menerapkan fungsi *hyperlane* pada data sehingga terbentuk daerah-daerah tiap kelas. *Hyperlane* sendiri merupakan sebuah fungsi yang digunakan sebagai pemisah antar kelas yang ada, *hyperlane* memisahkan kelas dengan label positif dan

kelas dengan label negatif. Dalam memprediksi suatu kelas dari data, SVM akan melabelinya berdasarkan daerah kelas mana yang merupakan tempat dari data tersebut, SVM biasanya digunakan pada *dataset* besar yang diambil dari situs *online* dan menjadi populer karena penerapannya dalam klasifikasi teks, metode SVM sendiri memiliki 3 model pendekatan atau yang lebih sering disebut dengan kernel, yaitu kernel radial, linier, dan sigmoid (Fikri et al., 2020).



Gambar 2. 2 *Hyperlane* yang memisahkan dua kelas

Hyperlane yang ditemukan SVM diilustrasikan seperti pada gambar 2.2 posisinya berada ditengah-tengah antara dua kelas. Artinya jarak antara *hyperplane* dengan objek-objek data berbeda dengan kelas yang berdekatan atau dengan kelas yang terluar yang diberi tanda bulat kosong dan positif. Dalam SVM objek data terluar yang paling dekat dengan *hyperlane* disebut *support vector*. Objek yang disebut *support vector* paling sulit diklasifikasikan dikarenakan posisi yang hampir tertutup *(overlap)* dengan kelas lain. Mengingat sifatnya yang kritis, hanya *support vector* inilah yang diperhitungkan untuk menemukan *hyperlane* yang paling optimal oleh SVM (Resa et al., 2022).

SVM telah menjadi metode klasifikasi dan regresi yang populer untuk masalah linear dan non linear, dan SVM terdiri dari 2 jenis yaitu simple SVM atau biasa disebut dengan SVM biasa dan Kernel SVM yang digunakan untuk memetakan berdasarkan fungsi kernel tertentu. Keistimewaan dari SVM berasal dari kemampuan untuk menerapkan pemisahan linear pada *input* data non linear berdimensi tinggi, dan ini diperoleh dengan menggunakan fungsi kernel yang diperlukan, efektivitas SVM sangat dipengaruhi oleh jenis fungsi kernel yang dipilih dan diterapkan berdasarkan karakteristik data (Indrayuni, 2018).

Kasus klasifikasi yang secara linier dapat dipisahkan menjadi prinsip dasar cara kerja SVM, namun SVM telah mengalami perkembangan agar dapat bekerja pada problem non-linier yaitu dengan memasukkan konsep kernel pada ruang kerja berdimensi tinggi. SVM memiliki prinsip utama yaitu menemukan *hyperlane* terbaik yang berfungsi memisahkan dua buah kelas pada ruang input. *Hyperlane* tersebut dapat berupa garis pada *two-dimension* dan dapat berupa *flat plane* pada *multiplane,* pencarian lokasi *hyperlane* optimal merupakan inti dari metode SVM (Muttaqin and Kharisudin, 2021).

Model persamaan SVM ditunjukkan pada persamaan(Maulana et al., 2024) :

$$f\ (x) = w.x + b \qquad \text{Rumus (2.1)}$$

w = parameter *hyperlane* yang dicari

x = titik data masukan

b = parameter *hyperlane* yang dicari

f = fungsi *hyperlane*

Fungsi klasifikasi dalam klasifikasi SVM diwakili oleh f(x). Fungsi f(x) memprediksi kelas target y berdasarkan fitur *input* x, dan w adalah vektor bobot, x adalah vektor masukan fitur, dan b adalah bias, sehingga persamaan formula perumusan diperoleh:

$$[(WT.xi) + b] \geq 1\ untuk\ yi = 1 \qquad \text{Rumus (2.2)}$$
$$[(WT.xi) + b] \leq 1\ untuk\ yi = -1$$

2.4. *Naive Bayes*

Metode ini berasal dari teorama Bayes, yang ditemukan oleh Thomas Bayes pada abad ke-18. Digunakan untuk memprediksi kemungkinan keanggotaan kelas, klasifikasi *Naive Bayes* adalah teknik statistik. Metode ini menggunakan teori probabilitas, cabang matematika, untuk menemukan kemungkinan tertinggi terhadap suatu klasifikasi dengan melihat frekuensi masing-masing klasifikasi pada data pelatihan. *Naive Bayes* bekerja dengan cara mencari nilai probabilitas besyarat

terbesar dari masing-masing kelas, *classifier* ini diklasifikasikan ke kelas tertentu sesuai teori probabilitas (Kusnia et al., 2022).

Naive Bayes merupakan metode klasifikasi yang dapat memprediksi probabilitas sebuah *class,* sehingga dapat menghasilkan keputusan berdasarkan data pembelajaran. Dari kelompok pendekatan numeris, *Naive Bayes* memiliki kelebihan antara lain, sederhana, cepat, dan berakurasi tinggi, serta efisien karena mampu mempersingkat waktu analisis sentimen teori *Bayesian* juga dapat digunakan sebagai alat pengambilan keputusan salah satunya tentang analisis sentimen, *Naive Bayes* yang sederhana, cepat dan berakurasi tinggi diharapkan dapat dengan cepat mengeksekusi kata dari banyaknya data pada *library* yang selanjutnya dapat menganalisis sentimen dari kalimat berbahasa Indonesia yang memiliki kosa kata yang sangat banyak dan kompleks, *Naive Bayes* memiliki sifat yaitu efek dari nilai atribut pada kelas yang diberikan tidak dipengaruhi oleh atribut atau kelas lain yang didasari oleh teori *Bayes* (Hibban and Susila., 2021).

Dalam teori *Naive Bayes* suatu probabilitas bersyarat dinyatakan dengan persamaan sebagai berikut:

$$P\ (H|X) = \frac{P(X|H).P(H)}{P(X)} \qquad \text{Rumus (2.3)}$$

Di mana:

X = Data dengan kelas yang belum diketahui

H = Hipotesis data merupakan suatu kelas spesifik

P (H | X) = Probabilitas hipotesis H berdasarkan kondisi X

P (H) = Probabilitas Hipotesis H

Dalam *Machine Learning* X pada rumus tersebut adalah sebuah *tuple* atau objek data, H adalah hipotesis dan atau dugaan bahwa *tuple* X adalah kelas C (Pamungkas and Kharisudin, 2021).

Metode ini direpresentasikan dengan pasangan atribut $X_1, X_2, X_3, X_4, X_5, X_n,$ Di mana X_1 adalah kata pertama, X_2 adalah kata kedua, $X_3,$ adalah kata ketiga dan seterusnya. Sedangkan V adalah himpunan kategori data teks sehingga pada saat klasifikasi algoritma akan mencari probabilitas tertinggi dari semua kategori

kalimat yang diuji $V_{NB.}$ Persamaannya dapat dituliskan dengan menggunakan rumus (Ilmawan and Mude, 2020b) :

$$V_{NB} = \frac{\arg max}{V_j EV} \; P\,(vi) \prod \prod_{i=1}^{n} (Xi|Vj) \qquad \text{Rumus (2.4)}$$

2.5. *Text Preprocessing*

Text Preprocessing atau pra-proses teks adalah salah satu bagian pada analisis sentimen yang berguna untuk melakukan seleksi data yang akan diproses pada setiap dokumennya di mana data yang digunakan tidak selamanya dalam kondisi ideal untuk diproses, di mana sudah ada pengurangan kosa kata, menghilangkan *noise*, membuat data lebih terstruktur sehingga diharapkan dapat mempermudah dan mempercepat proses selanjutnya yaitu melakukan pemodelan data. Proses mengolah teks yang difungsikan untuk pengubahan bentuk dokumen menjadi data yang terstruktur sesuai dengan kepentingannya supaya dapat diolah lebih lanjut. Langkah *text preprocessing* dalam klasifikasi berfungsi untuk memaksimalkan akurasi klasifikasi data (Ridwansyah, 2022).

2.6. *Lexicon Based*

Lexicon based adalah suatu proses pemulihan kata penting pada dokumen berdasarkan suatu kamus/leksikon yang sudah ada. Penentuan dilakukan pada data teks berupa kalimat yang memiliki kata pada kamus *lexicon* yang terdiri dari kata negatif dan positif. Kata yang teridentifikasi dalam kamus *lexicon* akan dihitung skornya sesuai dengan jumlah kata pada setiap teks atau kalimat.

Metode analisis dari metode *lexicon based* adalah *VADER* (*Valance Dictionary and Sentimen Reasoner*). *Vader* digunakan untuk menganalisis data berdasarkan *lexicon* (kamus). Hasil dari *Vader* berupa kelas prioritas positif, dan negatif dengan tambahan *compound score* atau skor total. *Vader* sentimen *lexicon* memiliki 7.500 kata (Saron Tandiapa and Caren Rorimpandey, 2023).

Lexicon Based merupakan salah satu cara dalam klasifikasi sentimen berdasarkan kamus kata. Pelabelan dilakukan untuk data *training* yang di mana data tersebut berfungsi dalam pembentukan model dari setiap algoritma. Pelabelan ini

dilakukan dengan memanfaatkan korpus berisi kata positif dan negatif yang telah digunakan pada penelitian sebelumnya. Kamus kata-kata dengan sentimen diperlukan untuk proses ini, dikenal sebagai *sentiment dictionaries.* Kajian terhadap *lexicon* mencakup kata abstrak *leksem*, strukturisasi kosakata, penggunaan dan penyimpangan kata serta evolusi kata. Setiap kata yang terdapat pada kamus positif akan diberi nilai 1, jika terdapat pada kamus negatif akan diberi nilai -1. Kemudian setiap ada kata negasi setelah maupun sebelum kata akan dikali dengan -1. Setelah semua kata dari sebuah kalimat telah diberi label, maka semua nilai akan di akumulasi. Apabila jumlahnya bernilai positif maka akan dikategorikan positif, bila negatif akan diklasifikasikan negatif (Simbolon et al., 2021).

2.7. Pembobotan TF – IDF

Pembobotan TF-IDF (*term frequency-Inverse Document Frequency)* dapat didefinisikan sebagai metode untuk menentukan nilai frekuensi sebuah kata di dalam sebuah dokumen atau artikel. Pada tahap ini terdapat dua bagian proses yaitu TF (*term frequency)* dan IDF (*Inverse Document Frequency),* TF adalah jumlah kemunculan kata dalam sebuah dokumen, semakin banyak kata yang muncul pada setiap dokumen maka semakin besar pula nilai TF-nya. IDF adalah jumlah nilai dokumen untuk setiap kata yang berbanding terbalik, yaitu jika sebuah kata jarang muncul dalam sebuah dokumen, maka nilai IDF lebih besar dari kata yang sering muncul, sehingga nilai TF dan IDF merupakan nilai yang dibandingkan untuk mencari apakah suatu kata sering muncul atau jarang muncul. Secara sistematis, TF-IDF dapat dijabarkan sebagai berikut (Alfyando, 2024):

$$TF - IDF\ (d,t) * IDF\ (t) \qquad \text{Rumus (2.5)}$$

$$TF\ (d,t) = \frac{Jumlah\ kata\ t\ pada\ dokumen\ d}{total\ kata\ pada\ dokumen\ d}$$

$$IDF\ (t) = log\frac{D}{df}$$

Di mana:

t = kata d = dokumen

df = jumlah dokumen yang mengandung kata t D = jumlah dokumen

Term Frequency (TF) adalah faktor yang memberikan bobot *term* pada sebuah dokumen yang didasarkan pada seberapa sering *term* tersebut muncul dalam suatu dokumen. *Inverse Document Frequency* (IDF) mengacu pada penghapusan kata-kata yang mendominasi atau sering ada dalam dokummen. *Term Frequency Inverse Document Frequency (*TF-IDF) adalah hasil perkalian *Term Frequency* dengan *Inverse Document Frequency* (Septiani and Isabela, 2022).

2.8. *Confusion Matrix*

Confusion Matrix adalah pengukuran masalah untuk performa *machine learning* di mana keluarannya merupakan dua kelas atau lebih dari itu. *Confusion Matrix* juga merupakan kombinasi dari 4 tabel dari nilai prediksi dan nilai aktual yang berbeda-beda yang memberikan informasi perbandingan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang sebenarnya. Ada 4 hasil yang merupakan istilah proses representasi yaitu TP (*True Positive),* FP (*False Positive),* FN (*False Negative),* TN (*True Negative). Confusion Matrix* merupakan alat untuk evaluasi visual yang digunakan dalam pembelajaran mesin. *Confusion Matrix* memberikan keputusan yang diperoleh dalam *training* dan testing. Prediksi kolom *Confusion Matrix* adalah mewakili hasil prediksi kelas dan merupakan alat untuk evaluasi visual yang digunakan dalam pembelajaran mesin dan termasuk analisis sentimen. Contoh dari bentuk *Confusion Matrix* 2x2 terdapat secara berurutan pada tabel 2.1 (Oktafiandi and Raziq Olajuwon, 2023).

Tabel 2. 1 *Confusion Matrix* 2x2

		Nilai Aktual	
		Positif	Negatif
Nilai Prediksi	Positif	TP	FP
	Negatif	FN	TN

Di mana:

a. TP adalah *True Positive,* yaitu jumlah data positif yang terklasifikasikan dengan benar oleh sistem.

b. TN adalah *True Negative,* yaitu jumlah data negatif yang terklasifikasikan dengan benar oleh sistem.
c. FN adalah *False Negative,* yaitu jumlah data negatif namun terklasifikasikan salah oleh sistem.
d. FP adalah *False Positive,* yaitu jumlah data positif namun terklasifikasin salah oleh sistem.

Nilai *Confusion Matrix* berfungsi untuk menghitung tingkat akurasi dari prediksi klasifikasi yang dihasilkan. Terdapat beberapa metode yang dapat digunakan dalam menilai performa sistem klasifikasi seperti Akurasi*,* presisi, *recall,* dan F-1 *score.* Hasil dari *Confusion Matrix* digunakan untuk menentukan akurasi dari *classifier* dengan *performance matrix accuracy.*

2.9. Perangkat dan Alat Bantu

2.9.1. Bahasa Pemrograman *Python*

Python adalah bahasa pemrograman yang menggunakan interpreter untuk menjalankan kode programnya. Interpreter tersebut dapat menerjemahkan kode secara langsung, dan *Python* dapat dijalankan di berbagai *platform* seperti Wndows, dan Linux. *Python* mengadopsi paradigma pemrograman dari beberapa bahasa lain, termasuk paradigma pemrograman prosedural seperti bahasa C, pemrograman berorientasi objek seperti Jawa, dan bahasa fungsionalitas seperti LISP. Kombinasi paradigma ini memudahkan para *programmer* dalam mengembangkan berbagai proyek menggunakan *Python*

Python dapat digunakan untuk berbagai keperluan, seperti pengembangan aplikasi web, aplikasi desktop, dan IoT. *Python* juga memiliki integrasi dengan sistem *database* dan mampu membaca serta mengubah file, sehingga sering digunakan untuk *prototyping* atau pengembangan perangkat lunak dengan cepat dan reliabel. Selain itu, *Python* juga digunakan secara luas karena kemampuannya dalam menangani data besar termasuk menangani analisis sentimen yang perlu menggunakan data yang besar karena proses yang dilakukan dengan menggunakan *python* akan lebih cepat sehingga pada data yang besar proses akan dilakukan dengan lebih cepat (Sayuti Rahman et al., 2023).

Berbagai bahasa pemrograman pasti memiliki kelebihan dan kekurangan. Hal tersebut tidak terlepas dari bahasa pemrograman *python*. Berikut ini merupakan kelebihan dari bahasa pemrograman *python* (Harani and Nugraha, 2020).

a. *Python* cukup mudah untuk digunakan dan dipelajari dibandingkan dengan bahasa pemrograman lainnya. Memiliki *sintaks* yang sederhana, mudah diingat serta dipahami karena hal tersebut diperoleh dari filosofi *pyhton* itu sendiri yaitu menekankan pada aspek kemudahan untuk dibaca (*readibility).* Dikarenakan *source code* yang dimiliki oleh *python* memiliki kemudahan dalam penulisan dan pembacaan, sehingga dapat lebih mempermudah melakukan perbaikan apabila terjadi kesalahan dalam proses pembuatan aplikasi dan memudahkan dalam pemeliharaan.
b. *Pyhton* merupakan bahasa pemrograman yang memiliki berbagai fungsi (multifungsi). Oleh karena itu, bahasa pemrograman *python* dapat digunakan untuk membuat berbagai macam produk aplikasi baik itu aplikasi *website, game,* robotika, *data mining,* sampai aplikasi berbasis kecerdasan buatan. Selain itu aplikasi *dekstop* dan *mobile* pun bias dibuat dengan menggunakan *python*
c. *Python* memiliki dukungan *library* (pustaka) standar yang banyak. Kemudian *packages* yang disediakan untuk mendukung kebutuhan pembuatan program pun tersedia sangat banyak baik itu dibuat oleh pengembangan *official* dari *python* maupun pengembang dari pihak ketiga.
d. *Python* merupakan bahasa pemrograman yang berorientasi secara otomatis seperti halnya pada java.

Selain kelebihan *python* juga memiliki kekurangan seperti bahasa pemrograman lainnya, berikut kekurangan dari bahasa pemrograman *python:*

a. Terdapat beberapa tugas yang tidak dapat dilakukan dan berada di luar jangkauan kemampuan dari *python.* Walaupun dikatakan *python* merupakan bahasa pemrograman yang dinamis, namun *python* tidak secepat atau efisien sebagai statis.
b. *Python* dapat digunakan untuk pembuatan aplikasi *mobile*. Namun untuk pengembangan kedepan, penggunaan *python* cukup buruk untuk aplikasi *mobile*

c. *Python* bertindak sebagai interpereter, bukan sebagai alat bantu terbaik untuk pengantar komponen kinerja kritis. Sehingga *python* bukan merupakan pilihan yang baik untuk melakukan tugas-tugas intensif memori.
d. Terbatasnya akses terhadap basis data

2.9.2. Google Colab

Google Colab adalah sebuah IDE untuk pemrograman *Python* di mana pemrosesan akan dilakukan oleh server Google yang memiliki perangkat keras dengan performa yang tinggi. Dari sisi perangkat lunak, Google Colab telah menyediakan hampir sebagian besar pustaka (*library)* yang dibutuhkan termasuk pustaka (*library)* yang dibutuhkan dalam pemrosesan analisis sentimen.

Google telah menciptakan Colaboratory, layanan *cloud* untuk menyebarkan pendidikan dan penelitian *machine learning. Runtime* yang disediakan oleh layanan *cloud* ini sepenuhnya dikonfigurasikan dengan pustaka kecerdasan buatan terkemuka dan juga menawarkan GPU yang kuat. Sehingga memungkinkan pengguna untuk dapat mengeksekusi kode *Python* secara gratis dilingkungan *cloud* menggunakan sumber daya komputasi Google, sepeti CPU, GPU, dan TPU, layanan ini memungkinkan pengguna untuk membuat, menjalankan, dan berbagi *notebook* Jupyter tanpa perlu melakukan pemasangan atau melakukan konfigurasi lingkungan pengembangan di komputer lokal. Google Colab dapat diakses melalui browser web dan menyediakan integrasi dengan Google Drive, memungkinkan pengguna menyimpan dan mengelola *notebook* yang mereka gunakan secara *online*. Google Colab dapat mendukung kebutuhan kolaborasi tim, karena *notebook* yang telah dibuat dapat diedit secara bersamaan oleh anggota tim lain Ini adalah alat yang sangat berguna untuk eksplorasi dan eksperimen dalam pemrosesan data dan pembelajaran mesin tanpa harus khawatir tentang konfigurasi lingkungan lokal. Fitur Utama dari Google Collab meliputi (Gelar Guntara, 2023):

a. *Notebook* Interaktif
b. Penggunaan sumber daya komputasi
c. Penyimpanan dan pembagian

2.10. Penelitian Terdahulu

Penelitian terdahulu mengenai analisis sentimen telah banyak dilakukan, dengan menggunakan berbagai macam metode. Dengan mengumpulkan penelitian terdahulu maka dapat digunakan untuk memperoleh informasi dan data yang berhubungan dengan judul penelitian salah satunya metode yang digunakan yaitu *Support Vector Machine* dan *Naive Bayes*.

Tabel 2. 2 Penelitian Terdahulu

<table>
<tr><td>No.</td><td colspan="2">1</td></tr>
<tr><td></td><td></td><td>Hasil Penelitian</td></tr>
<tr><td>Nama Peneliti</td><td>Lutfi Budi Ilmawana, dan Muhammad Aliyazid Mudeb</td><td rowspan="3">Hasil akurasi dari Support Vector Machine Classifier memili nilai yang lebih tinggi jika dibandingkan dengan akurasi dari Naive Bayes Classifier untuk mengklasifikasikan ulasan tekstual berbahasa Indonesia pada Google Play Store, yaitu SVM Classifier mendapatkan akurasi sebesar 81,46% dan Naive Bayes Classifier sebesar 75,41%.</td></tr>
<tr><td>Judul Penelitian</td><td>Perbandingan Metode Klasifikasi Support Vector Machine dan Naive Bayes Untuk analisis Sentimen pada Ulasan Tekstual di Google Play Store</td></tr>
<tr><td>Metode dan Dataset</td><td>Metode : Support Vector Machine dan Naive Bayes

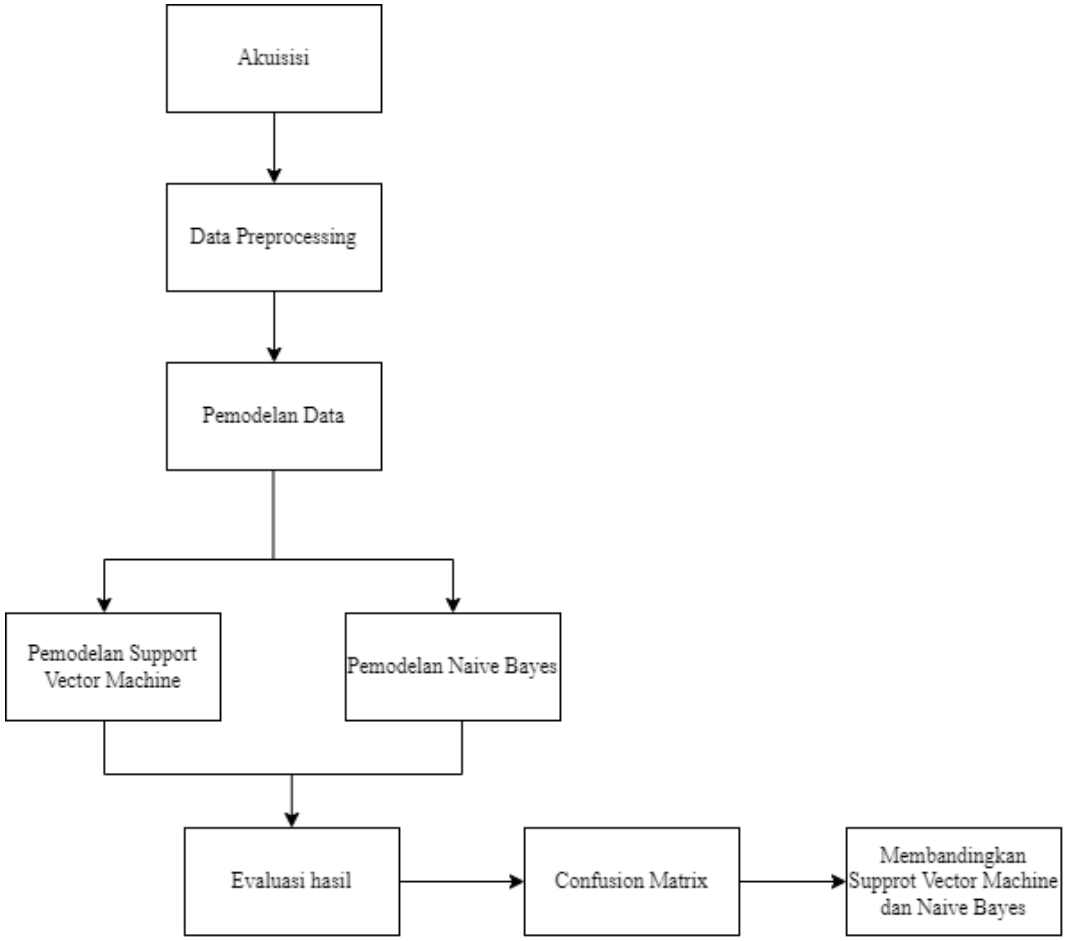
Dataset : 1818</td></tr>
<tr><td>No.</td><td colspan="2">2</td></tr>
<tr><td></td><td></td><td>Hasil Penelitian</td></tr>
<tr><td>Nama Peneliti</td><td>Meishita Inelza Putria, dan Iqbal Kharisudin</td><td rowspan="2">Kesimpulan dari penelitian diperoleh bahwa hasil tersebut menunjukkan bahwa partisi data menggunakan 10-fold cross validation menghasilkan performa klasifikasi lebih baik dibandingkan split data 80:20 karena konsep k-fold cross validation membagi dataset menjadi</td></tr>
<tr><td>Judul Penelitian</td><td>Analisis Sentimen Pengguna Aplikasi Marketplace Tokopedia pada Situs</td></tr>
</table>

	Google Play Menggunakan Metode *Support Vector Machine* (SVM), *Naive Bayes,* dan *Logistic Regression*	beberapa bagian (*k-subse*t) dengan ukuran yang sama dan melakukan pelatihan dan pengujian pada model sebanyak k yang ditentukan. Penggunaan SMOTE terbukti dapat meningkatkan akurasi model pada data yang tidak seimbang (*imbalance)* dilihat dari nilai AUC pada model dengan SMOTE lebih tinggi dari pada nilai AUC yang dihasilkan oleh model non-SMOTE sehingga penerapan SMOTE lebih efektif dalam meningkatkan ketepatan akurasi klasifikasi. Metode klasifikasi yang tepat digunakan untuk klasifikasi *review* pengguna Tokopedia adalah *k-fold cross validation* dengan SMOTE, yaitu metode klasifikasi *Support Vector Machine* dengan nilai AUC sebesar 1,000, sehingga metode tersebut tergolong memiliki kinerja klasifikasi sangat baik. Secara umum *review* pengguna lebih banyak bersentimen positif yang berkaitan dengan kemudahan penggunaan aplikasi tersebut, sedangkan *review* negatif berkaitan dengan sistem aplikasi *error* dan sering terjadi *lagging/bug*.
Metode dan Dataset	Metode: *Support Vector Machine (SVM), Naive Bayes,* dan *Logistic Regression* Dataset: 3125	
No.	**3**	
		Hasil Penelitian
Nama Peneliti	Anggi Puji Astuti, Syariful Alam, dan Irsan Jaelani	Hasil penelitian analisis sentimen mengenai ulasan pengguna aplikasi *mobile banking* BRImo pada platform *google playstore* dengan dua algoritma klasifikasi yaitu *Support vector machine* dan *Naive*

Judul Penelitian	Kompirasi Algoritma *Support Vector Machine* dengan *Naive Bayes* untuk Analisis Sentimen pada Aplikasi BRImo	*Bayes* menggunakan pembobotan kata TF-IDF untuk mengetahui nilai akurasi dari perbandingan dua algoritma yang digunakan dengan jumlah 5000 data ulasan hasil *scrapping*. Data diklasifikasikan dengan 5x percobaan menggunakan metode *Support vector mechine* yang menghasilkan nilai akurasi 97,56%, sedangkan metode *Naive Bayes* menghasilkan nilai akurasi 96,52% serta evaluasi data menggunakan. Sehingga kesimpulan dari penelitian adalah algoritma *Support vector machine* memiliki nilai yang lebih baik untuk klasifikasi data ulasan aplikasi *mobile banking* BRImo dibandingkan algoritma *Naive Bayes.*
Metode dan Dataset	Metode: *Ssupport Vector Machine* dan *Naive Bayes* Dataset : 5000	
No.	**4**	
		Hasil Penelitian
Nama Peneliti	Ulfa Kusnia, dan Fachrul Kurniawan	Berdasarkan hasil prosedur pengujian yang dilakukan terhadap *Review* Aplikasi Media Berita Online Pada *Google Play* Mengunakan Algoritma *Support Vector Machine* (SVM) dan Algoritma *Naive Bayes*. Hasil pengklasifikasian data opini kelas sentimen positif menjadi 5160 dan negatif menjadi 455. Hasil *Experimen machine learning* dengan perbandingan data *traning* 80% dan data testing 20%, menghasilkan nilai akurasi untuk algoritma *Support Vector Machine* (SVM) 88% dan algoritma *Naive Bayes* 87%. Untuk data sentimen *Riview* Aplikasi Media Berita Online membutuhkan upaya yang tidak mudah saat tahap *prossesing* di awal yaitu bagian pelabelan. Faktor penyebab sulitnya pelabelan yaitu penanganan sentimen negasi kata atau kalimat yang membalik makna pesan dalam bahasa indonesia, sehingga belum dapat ditentukan polaritasnya dengan optimal. Sebagai perbaikan untuk peneliti berikutnya perlu menggunakan metode *machine learning* lain.
Judul Penelitian	Analisis Sentimen *Review* Aplikasi Media Berita Online Pada *Google Play* menggunakan Metode Algoritma *Support Vector Machines (SVM)* dan *Naive Bayes*	
Metode dan Dataset	Metode: *Support Vector Machine* (SVM) dan *Naive Bayes* Dataset: 5615	

3. METODE PENELITIAN

Metodologi yang digunakan untuk penelitian ini mencakup beberapa tahapan yang dapat dilihat pada gambar 3.1 di bawah. Penelitian ini dilakukan dengan pengambilan data dari ulasan pengguna di Google Play Store, dengan 2 tahapan yaitu pengumpulan data dan metode analisis data.



Gambar 3. 1 Tahapan Penelitian

Penelitian ini diawali dengan melakukan akuisisi data berupa ulasan yang berasal dari Google Play Store. Tahap berikutnya adalah melakukan pemrosesan terhadap data yang baru saja diakuisisi, tahapan ini bernama *text preprocessing,* pada tahapan ini data yang ada dilakukan pembersihan melalui 6 tahapan yaitu *case folding, cleaning, normalization, stopwords removal, stemming, tokenizing*, tahapan ini dimaksudkan untuk menghasilkan data yang terstruktur. Tahapan selanjutnya

yaitu melakukan pemodelan data, tahapan pemodelan data diawali dengan melakukan pembobotan kata menggunakan TF – IDF, selanjutnya melakukan pelabelan data dengan menggunakan kamus *lexicon based* yang digunakan untuk penentuan kelas positif atau negatif, tahapan terakhir pada proses pemodelan data adalah melakukan pembagian data latih dan data uji, setelah tahapan tersebut selesai dilakukan selanjutnya adalah dengan melakukan klasifikasi menggunakan 2 metode yaitu *Support Vector Machine* dan *Naive Bayes*. Tahapan selanjutnya dari penelitian ini adalah melakukan model evaluasi dengan menggunakan *confusion matrix* untuk menentukan 4 macam hasil performa yaitu akurasi, presisi, *recall,* dan *f-measure.* Serta membandingkan 4 nilai performa antara 2 metode tersebut untuk mengetahui keunggulan dari masing-masing metode.

3.1. Akuisisi Data

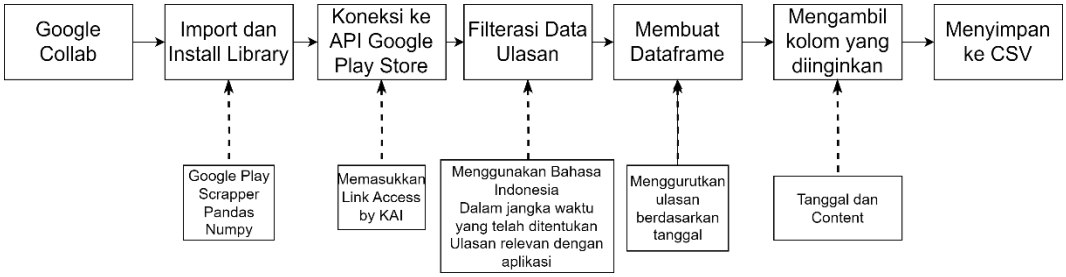
Tahapan pertama yang dilakukan dalam penelitian ini yaitu melakukan akuisisi data dari ulasan aplikasi Access by KAI di Google Play Store.

Gambar 3. 2 Tahapan Akuisisi Data

Proses ini bertujuan untuk melakukan pengumpulan data ulasan yang terdapat pada Google Play Store untuk ulasan aplikasi Access by KAI dengan kriteria data yang diakuisisi yaitu ulasan yang diakuisisi merupakan ulasan dalam rentang waktu tertentu, yaitu mulai dari tanggal 10 Agustus 2023 hingga 13 Maret 2024, rentang waktu ini dipilih karena pada tanggal 10 Agustus 2023 Access by KAI meluncurkan versi terbaru. Ulasan yang diakuisisi merupakan ulasan yang ditulis dalam Bahasa Indonesia dan memiliki relevansi dengan aplikasi tersebut untuk memastikan kualitas data yang digunakan. Data ulasan tersebut akan dikumpulkan menjadi satu untuk membuat sebuah *dataset* yang berisi ulasan dari aplikasi Access by KAI. *Dataset* tersebut akan diubah ke dalam sebuah *datafrane, field* yang terdapat dalam *dataframe* tersebut yaitu, tanggal ulasan tersebut diberikan, serta *content* yang merupakan isi ulasan dari pengguna aplikasi Access by KAI. *Dataframe* tersebut selanjutnya akan disimpan ke dalam format CSV,

format CSV dipilih karena kesederhanaannya dalam penyimpanan data tabular dan kompatibilitasnya dengan berbagai *tools* yang digunakan untuk melakukan analisis sentimen. Data ulasan ini penting karena akan menjadi dasar untuk dapat melalukan analisis sentimen. Tahapan akuisisi data dapat dilihat pada gambar 3.2 di atas.

Akuisisi data dilakukan menggunakan bahasa pemrograman Python dengan *library* google_play_scrapper, *library* tersebut merupakan *Application Programming Interface* (API) untuk mengekstraksi data informasi aplikasi dan ulasan aplikasi dari Google Play Store. Cara untuk mendapatkan ulasan Access by KAI dapat dilihat pada gambar 3.3 di bawah.



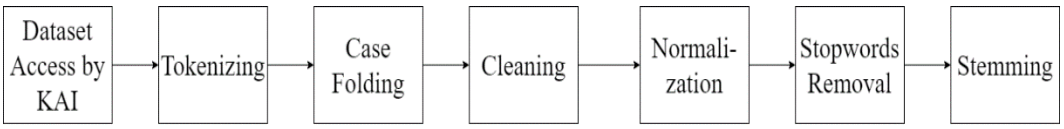
Gambar 3. 3 Alur *Filterisasi* Ulasan Access by KAI

Library yang dibutuhkan selain google_play_scrapper adalah *library pandas* dan *numpy*, setelah keseluruhan *library* terpasang maka proses akuisisi data dapat dilanjutkan yaitu, memasukkan kriteria data yang akan diakuisisi. *Dataset* yang dihasilkan oleh proses tersebut akan sesuai dengan kriteria data yang ingin diakuisisi, setelah itu mengubah *dataset* tersebut ke dalam bentuk *dataframe* dengan 2 *field* yaitu *content* dan *at, content* merupakan ulasan sedangkan *at* adalah tanggal ulasan, setelah itu tahapan terakhir dari program tersebut adalah menyimpan *dataframe* tersebut dalam format *CSV* untuk selanjutnya akan dilakukan tahapan *preprocessing*.

3.2. Data *Preprocessing*

Tahapan yang dilakukan terhadap dokumen setelah proses akuisisi selesai dilakukan yaitu Pemrosesan data atau data *preprocessing*. Proses *Preprocessing* ini melibatkan filtrasi *dataset* atau data asli yang dihasilkan dari akuisisi data. Tahapan ini bertujuan untuk melakukan perbaikan atau perubahan pada data hasil akuisisi sehingga menjadi sebuah data ulasan yang terstruktur dan sesuai dengan kebutuhan

pada proses analisis sentimen. Tahapan ini penting untuk mempersiapkan data agar siap digunakan dalam model analisis sentimen yang akan diproses pada tahapan selanjutnya. tahapan data *preprocessing* dapat dilihat pada gambar 3.4 di bawah.

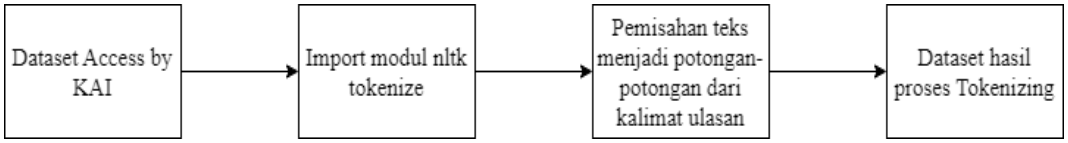


Gambar 3. 4 Tahapan *Preprocessing*

Berdasarkan gambar 3.4 di atas terdapat 6 tahapan *preprocessing* yang akan dilakukan terhadap *dataset* Access by KAI, pada tahapan pertama yaitu melakukan *tokenizing* tahapan ini berguna untuk memecah teks menjadi unit-unit yang lebih kecil, *case folding* tahapan ini berguna untuk menghilangkan huruf kapital, *cleaning* tahapan ini berguna untuk menghilangkan tanda baca, karakter khusus ataupun tautan, *normalization* tahapan ini berguna untuk mengubah kata yang tidak baku menjadi kata baku sesuai dengan kamus besar bahasa Indonesia (KBBI), *stopwords removal* tahapan ini berguna untuk menghilangkan kata yang tidak memiliki arti atau tidak dibutuhkan pada proses analisis sentimen, *stemming* tahapan ini berguna untuk menghilangkan kata yang diawali dengan imbuhan sehingga kata tersebut menjadi kata dasar. Tahapan *preprocessing* dilakukan dengan menggunakan bahasa pemrograman Python dengan bantuan *tools* Google Colab. Tahapan *preprocessing* dilakukan secara efektif dan efisien untuk mempersiapkan data sebelum dilakukan analisis sentimen lebih lanjut.

3.2.1. *Tokenizing*

Setelah tahapan akuisisi data selesai dilakukan, maka tahapan *preprocessing* yang akan pertama kali dilakukan adalah *tokenizing.*



Gambar 3. 5 Tahapan *Tokenizing*

Tokenizing bertujuan untuk memisahkan kata-kata dalam kalimat atau teks menjadi *token-token* individual. *Token* merupakan unit terkecil dari teks yang dianalisis, *token* dapat berupa kata tunggal, frasa, atau karakter individual. Dengan

memecah teks menjadi *token*, model dapat dengan mudah menganalisis teks secara lebih terperinci serta memahami struktur kalimat dan konteks dari teks yang akan diproses lebih lanjut.

Tokenizing diproses dengan menggunakan *Natural Language Toolking* (NLTK), NLTK yang digunakan merupakan *word_tokenize_wrapper,* fungsi ini adalah fungsi pembungkus (*wrapper)* sederhana untuk fungsi *word_tokenize* dari *library* NLTK. Sedangkan *word_tokenize* adalah fungsi yang menerima teks sebagai *input* dan mengembalikannya dalam bentuk *token*, dengan *token* yang berisikan kata atau tanda baca. Fungsi ini bekerja dengan *word_tokenize_wrapper* yang akan memproses setiap teks *tweet* dengan cara memanggil *word_tokenize* untuk mengubah teks menjadi daftar *token,. Word_Tokenize* akan memecah teks menjadi kata-kata dengan memanfaatkan pola spasi dan tanda baca sebagai pemisah, tanda baca seperti titik, koma, tanda seru akan dipisahkan dari kata-kata dan diperlakukan sebagai *token* individu setelah itu fungsi tersebut akan mengembalikan daftar *token* yang berisi kata-kata dan tanda baca yang telah dipisahkan.

Berikut merupakan contoh tahapan *tokenize* pada sebuah ulasan, "Aplikasi sangat bagus, keren sekali.". Proses akan dimulai dengan *Word_Tokenize_Wrapper* akan memanggil *Word_Tokenize* untuk menerima *input teks,* setelah itu dilakukan pemisahan kata, sehingga akan didapatkan hasil pada langkah 1, yaitu:

a. 'Aplikasi', 'sangat', 'bagus,', 'keren', 'sekali.',

Setelah itu akan dilakukan pemisahan terhadap tanda baca sehingga akan didapatkan hasil pada langkah 2, yaitu:

a. 'bagus', ','
b. 'sekali' '.'

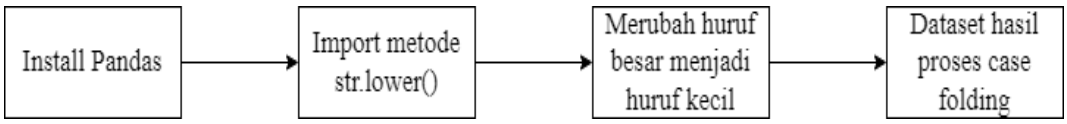
Proses *tokenizing* telah selesai dilakukan setelah itu *Word_Tokenize* akan mengembalikan kata dalam bentuk *token,* sehingga akan didapatkan hasil pada langkah 3, yaitu:

a. ['Aplikasi', 'sangat', 'bagus', ',' ;keren', 'sekali', '.']

Setelah ulasan berbentuk *token* maka dapat dilanjutkan ke *preprocesssing* selanjutnya. Tahapan proses *tokenizing* dapat dilihat pada gambar 3.5 di atas.

3.2.2. *Case Folding*

Tahapan selanjutnya setelah *dataset* melalui tahap *tokenizing* adalah *case folding.*



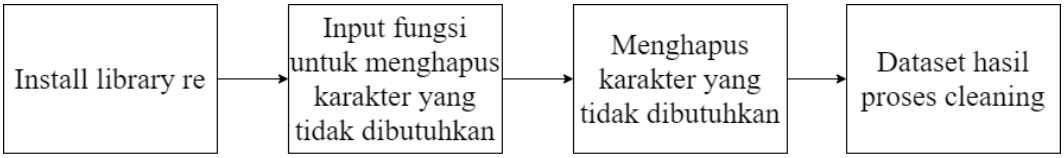
Gambar 3. 5 Tahapan *Case Folding*

Case folding bertujuan untuk mengubah seluruh huruf besar yang terdapat dalam *dataset* menjadi huruf kecil, proses ini untuk memastikan keseragaman dalam analisis teks tanpa memperhatikan apakah suatu kata ditulis dengan huruf besar atau huruf kecil dengan begitu maka dapat mengurangi variasi yang tidak diperlukan dalam data teks, sehingga dapat dipastikan bahwa semua teks berada dalam format yang konsisten, karakter-karakter lain seperti spasi dan tanda baca dianggap sebagai *delimeter* atau pemisah kata. Jika *case folding* tidak dilakukan maka kata "Aplikasi" dan "aplikasi" akan dianggap sebagai dua entitas yang berbeda.

Library yang digunakan dalam tahapan *case folding* adalah *pandas,* dengan memanfaatkan salah satu metode yang terdapat di *pandas* yaitu metode *Series.str.lower()*. Metode tersebut yang dimanfaatkan untuk mengubah huruf besar menjadi huruf kecil pada *dataset* yang digunakan. Metode tersebut akan mendeteksi ulasan yang terdapat huruf besar di dalam ulasan dan akan mengubahnya menjadi huruf kecil. Tahapan *case folding* dapat dilihat pada gambar 3.5 di atas.

3.2.3. *Cleaning*

Setelah tahapan *case folding,* langkah selanjutnya dalam proses *preprocessing* data adalah melakukan *cleaning.*



Gambar 3. 6 Tahapan *Cleaning*

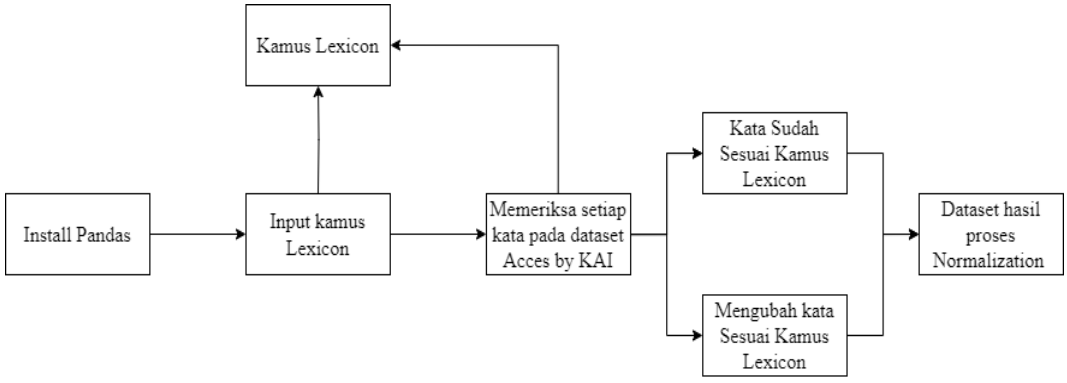
Cleaning bertujuan untuk menghilangkan berbagai elemen karakter khusus yang umumnya digunakan oleh pengguna dalam menulis ulasan aplikasi seperti

hashtag, mention ussername, URL website, Tag HTML, emoticon, koma, titik, dan spasi berlebih. Tanda baca seperti koma dan titik tidak memberikan nilai dalam analisis sentimen dan bisa menyebabkan model salah dalam menginterpretasikan data. Setelah berbagai elemen yang tidak dibutuhkan tersebut dihilangkan maka akan mengurangi *noise* atau gangguan acak dalam data yang dapat mempengaruhi hasil analisis sentimen. Selain itu pada proses *cleaning* juga dilakukan penghilang terhadap ulasan yang *duplicated* atau ganda dan juga pada kolom yang berisi nilai *null* atau tidak terdapat ulasan pada kolom tersebut, dengan menghilangkan semua gangguan tersebut maka *dataset* akan layak digunakan. Proses ini memastikan bahwa *dataset* yang akan dianalisis memiliki kualitas yang baik dan relevan.

Proses *cleaning* dilakukan dengan menggunakan *library* 're' yang merupakan singkatan dari *reguler expression* yang menyediakan serangkaian alat yang berguna untuk menangani teks, *library* tersebut akan memanggil sebuah pola yang digunakan untuk mencocokkan rangkaian karakter dalam teks dengan pola tersebut dapat menentukan karakter apa yang akan dihilangkan, setiap karakter di dalam *dataset* akan dicocokkan dengan *libary 're',* setiap karakter yang terdapat pada *library 're'* akan dihilangkan. Tahapan *cleaning* dapat dilihat pada gambar 3.6 di atas.

3.2.4. *Normalization*

Setelah tahapan *cleaning*, langkah selanjutnya dalam *preprocessing* adalah *normalization.*



Gambar 3. 7 Tahapan *Normalization*

Normalization bertujuan untuk mengubah kata-kata yang tidak baku yang terdapat dalam ulasan menjadi kata-kata baku yang sesuai dengan Kamus Besar Bahasa Indonesia (KBBI). Untuk melakukan ini, digunakan sebuah kamus yang berisi padanan kata dalam bahasa Indonesia yang sesuai dengan KBBI, yang disimpan dalam kamus *Lexicon.* Kamus *Lexicon* ini berfungsi sebagai referensi untuk memeriksa setiap kata dalam ulasan. Ketika kata dalam ulasan sudah sesuai dengan yang ada dalam kamus *Lexicon,* maka kata tersebut dianggap baku. Namun, jika kata dalam ulasan tidak ditemukan dalam kamus *Lexicon*, maka kata tersebut dianggap tidak baku dan akan diubah menjadi kata baku yang sesuai dengan KBBI. Setelah semua ulasan dinormalisasi, ulasan-ulasan tersebut di satukan kembali menjadi teks yang sudah melalui proses *normalization.* Contoh kamus *Lexicon* dapat dilihat dalam tabel 4.5 di bawah.

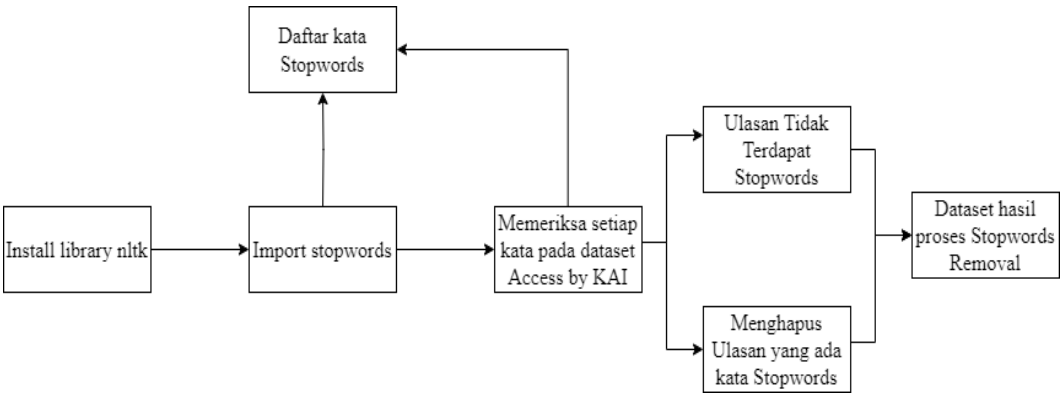
Tabel 3. 1 Contoh Kamus *Lexicon*

Slang	**Formal**
dapet	dapat
maap	maaf
gk	enggak
diluangin	diluangkan
hargs	harga
perhatiin	perhatikan
abal2	abal-abal
kasian	kasihan

Berdasarkan tabel 3. 1 diatas dapat dilihat bahwa kata-kata akan mengalami perubahan menjadi kata baku yang sesuai dengan KBBI. Proses *normalization* ini penting untuk memastikan konsistensi sehingga setiap kata yang memiliki makna yang sama ditulis dalam bentuk yang sama dan kualitas data yang digunakan dalam analisis sentimen sehingga akan menghasilkan analisis yang lebih akurat karena *dataset* akan dalam bentuk standar yang dapat dikenali oleh metode. Tahapan *normalization* dapat dilihat pada gambar 3.7 di atas.

3.2.5. *Stopwords Removal*

Setelah tahapan *normalization,* langkah selanjutnya dalam proses *preprocessing* adalah *stopwords removal.*



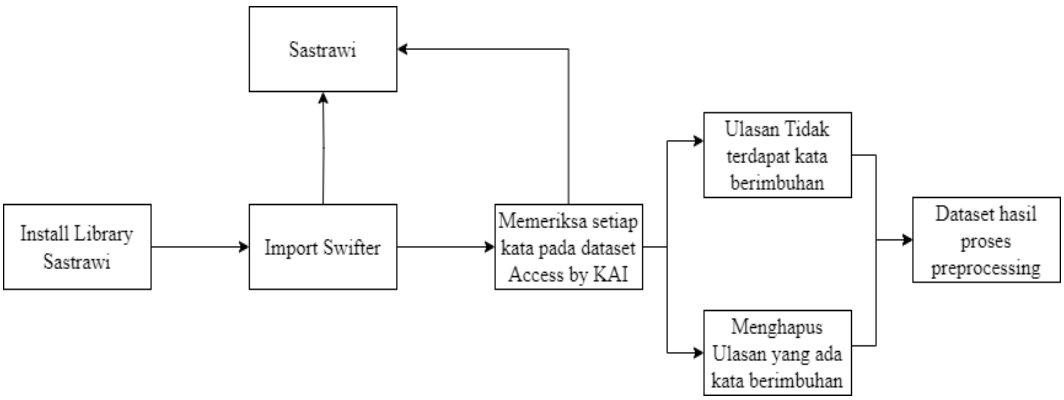
Gambar 3. 8 Tahapan *Stopwords Removal*

Stopwords merupakan kata-kata yang umumnya tidak memiliki makna penting dalam sebuah kalimat, seperti kata penghubung dan kata depan. Kata-kata ini cenderung sering muncul dalam teks namun tidak memberikan kontribusi signifikan dalam analisis sentimen. Ulasan akan dicocokkan dengan kamus *stopwords*, proses penghapusan *stopwords* ini bertujuan untuk mengurangi jumlah kata yang tidak relevan dalam proses analisis sentimen, sehingga meningkatkan kualitas analisis dan pada saat proses analisis dilakukan kata-kata yang ada dalam ulasan merupakan kata-kata yang memiliki arti. Dengan menghapus kata-kata yang tidak memiliki arti di dalam *dataset* maka data menjadi lebih bersih dan fokus pada kata-kata yang membawa makna penting, proses analisis sentimen menjadi lebih cepat dan efisien karena model tidak perlu memproses kata-kata yang tidak memberikan informasi tambahan.

Dalam tahapan ini *library* yang digunakan untuk melakukan pemrosesan adalah *nltk*, pada *llibrary* tersebut disediakan daftar *stopwords* yang dibutuhkan untuk melakukan penghapusan *stopwords* serta ditambahkan dengan menggunakan kamus *stopwrods* yang sudah disesuaikan dengan ulasan. Setiap kata yang ada pada *dataset* akan melalui tahap pencocokan dengan kata yang terdapat pada *library* nltk dan kamus *stopwords* yang telah disesuaikan dengan ulasan, sehingga setelah proses ini dapat dipastikan bahwa sudah tidak terdapat *stopwrods* pada *dataset.* Tahapan proses *stopwords removal* dapat dilihat pada gambar 3.8 di atas.

3.2.6. *Stemming*

Setelah tahapan *stopwords removal,* langkah terakhir dalam proses *preprocessing* adalah *stemming.*



Gambar 3. 9 Tahapan *Stemming*

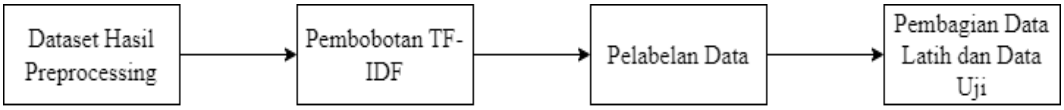
Stemming bertujuan untuk menghilangkan imbuhan pada kata-kata yang terdapat dalam ulasan untuk mengembalikan kata-kata tersebut ke bentuk dasarnya, proses *stemming* membantu dalam mengurangi variasi kata yang sebenarnya memiliki makna yang sama, namun ditulis dalam bentuk yang berbeda karena adanya imbuhan seperti awalan, sisipan, akhiran, dan kombinasi lainnya. Misalnya, “membantu”, “dibantu”, semuanya akan direduksi menjadi “bantu”. Dengan demikian, kata-kata yang diekstraksi dari ulasan akan lebih seragam dan mudah dipahami oleh model analisis sentimen sehingga dapat menghasilkan nilai akurasi yang tinggi pada setiap metodenya. Dengan proses *stemming* maka dapat mengurangi variasi kata, yaitu kata-kata yang memiliki makna yang sama akan diwakili oleh bentuk dasar yang sama. Hal ini mengurangi variasi kata dan memudahkan proses analisis, selain itu bentuk dasar kata-kata membuat data lebih konsisten, yang memungkinkan model analisis sentimen untuk lebih mudah mengenali pola dan hubungan dalam data.

Proses *stemming* dilakukan dengan menggunakan *library StemmerFactory* yang merupakan sebuah kelas yang terdapat dalam modul Sastrawi yang digunakan untuk membuat objek *stemmer*, modul sastrawi merupakan *library* yang digunakan untuk melakukan pemrosesan teks dalam bahasa Indonesia. Setiap kata yang terdapat dalam *dataset* akan dicocokkan dengan modul Sastrawi, jika ada ulasan yang memiliki imbuhan maka akan dilakukan proses *stemming*. Objek *stemmer*

akan menghilangkan imbuhan dari setiap kata untuk mengembalikannya ke bentuk kata dasar. Proses *stemming* merupakan tahapan terakhir dalam *preprocessing.* Tahapan proses *stemming* dapat dilihat pada gambar 3.9 di atas.

3.3. Pemodelan Data

Setelah tahapan *preprocessing* selesai dan *dataset* telah terstruktur, langkah berikutnya adalah pemodelan data.

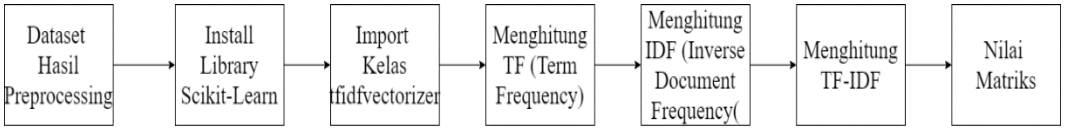


Gambar 3. 10 Tahapan Pemodelan Data

Tahap awal dalam pemodelan data adalah TF-IDF, TF-IDF merupakan metode yang menggabungkan dua konsep penghitungan bobot, yaitu frekuensi kata dalam dokumen tertentu dan frekuensi *invers* dari dokumen yang mengandung kata tersebut. Kemudian, dilakukan pelabelan dengan menggunakan kamus *Lexicon.* Metode ini mengklasifikasikan ulasan berdasarkan kata positif dan negatif dalam data, dengan menghasilkan *dataset* yang di masing-masing ulasannya akan terdapat kategori termasuk kelas positif atau kelas negatif ulasan tersebut. Tahapan terakhir dalam proses pemodelan data adalah, dilakukan pemisahan antara data latih dan data uji dengan menggunakan tiga rasio perbandingan data latih dan data uji yang berbeda. Tahapan proses pemodelan data dapat dilihat pada gambar 3.10 di atas.

3.3.1. Pembobotan TF – IDF

Tahapan pertama yang dilakukan pada fase pemodelan data adalah pembobotan TF-IDF.

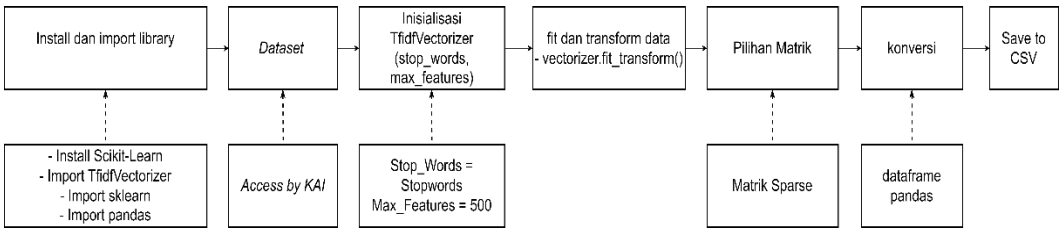


Gambar 3. 11 Tahapan Pembobotan TF-IDF

Tujuan dari proses ini adalah untuk mengubah data yang telah dilakukan *preprocessing* menjadi representasi data dalam bentuk matriks dengan nilai bobot *Term Frequency – Inverse Document Frequeny* atau yang sering disebut TF IDF.

TF merupakan perhitungan untuk menghitung frekuensi kemunculan kata dalam *dataset* tujuan dari nilai TF adalah untuk menyoroti kata-kata kunci yang penting dalam sebuah dokumen, sedangkan IDF adalah perhitungan untuk menentukan keunikan kata dalam seluruh *dataset,* tujuan dari IDF adalah untuk menyoroti kata-kata yang lebih langka dan unik dalam *dataset.* Dengan menggabungkan TF dan IDF maka akan membantu mengidentifikasi kata-kata yang memiliki arti penting dalam teks. Tahapan Pembobotan TF – IDF dapat dilihat pada gambar 3. 11 di atas.

Pembobotan TF-IDF menggunakan *Library Scikit-Learn* dengan memanfaatkan *tfidfvectorizer,* serta *fit_transform* untuk mengubah teks menjadi vektor TF-IDF. Pengaturan TF-IDF dapat dilihat pada gambar 3. 12 di bawah.



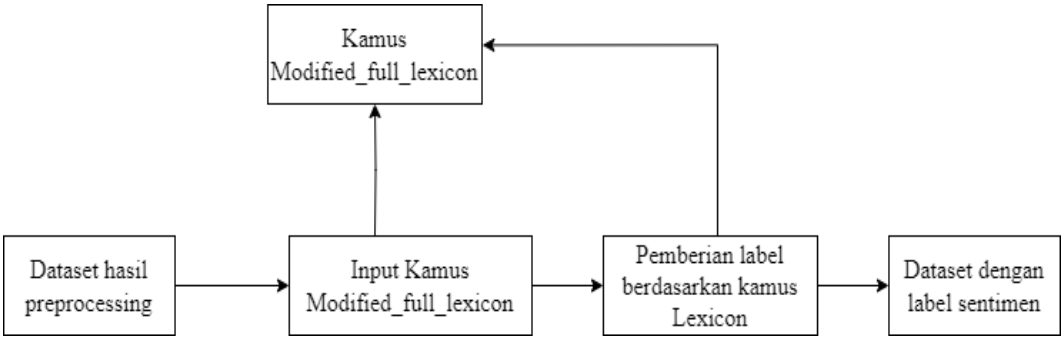
Gambar 3. 12 Penyesuaian TF-IDF Dengan Ulasan

Setelah *Library* selesai di*install* dan di*import,* langkah selanjutnya adalah menginisialisasikan objek *TfidfVectorizer* dari *scikit-learn,* objek ini berfungsi untuk menghapus kata-kata umum yang tidak memberikan informasi penting, yang disebut sebagai *stopwords.* Dalam tahap ini, terdapat beberapa parameter penting yaitu *stop_words* dan *max_features*, yang dapat diatur sesuai kebutuhan analisis untuk mengoptimalkan representasi teks. Pada *dataset* ulasan Access by KAI, *stop_words* menggunakan kamus *stopwords* khusus yang sesuai dengan ulasan, dan *Max_Features* diatur pada 1000, ini berarti bahwa *TfidfVectorizer* akan memilih 1000 kata-kata teratas berdasarkan frekuensi dan kepentingan dalam teks ulasan. Selanjutnya, menggunakan metode *fit_transform* dari *TfidfVectorizer* untuk mengubah teks ulasan menjadi vektor TF-IDF. Metode *fit_transform* ini menggabungkan dua langkah penting yaitu fit yang digunakan untuk mempelajari fitur dari teks yang ada, dan *transform*, yang mengubah teks menjadi representasi vektor berdasarkan fitur yang telah dipelajari. Hasil dari proses *fit_tansform* ini adalah matriks sparse yang berisi nilai TF-IDF untuk setiap kata dalam dokumen. Terakhir, hasil pembobotan TF-IDF disimpan dalam format yang dapat digunakan

oleh model yaitu *dataframe* pandas, yang memungkinkan integrasi yang lebih mudah dengan model analisis data. dan selanjutnya disimpan dalam format file CSV untuk dapat digunakan pada tahapan selanjutnya yaitu, pelabelan data dengan menggunakan *Lexicon Based*. Proses perhitungan dengan menggunakan *TF-IDF* dapat dilihat pada rumus (2.5)

3.3.2. Pelabelan Data Dengan Lexicon Based

Tahapan kedua dalam fase pemodelan data adalah melakukan pelabelan data dengan menggunakan kamus *Lexicon Based.*



Gambar 3. 13 Tahapan Pelabelan Data

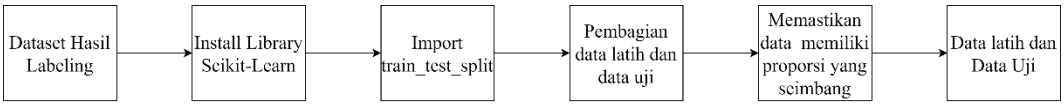
Proses ini bertujuan untuk menentukan apakah suatu ulasan memiliki makna positif atau negatif. Dalam analisis sentimen, pelabelan data adalah langkah krusial yang memungkinkan algoritma untuk memahami dan mengkategorikan teks berdasarkan sentimen yang terkandung di dalamnya. Untuk melakukan pelabelan data, digunakan kamus *Lexicon* Indonesia yang berisikan daftar kata-kata beserta nilai skor polaritasnya. Skor polaritas ini mengindikasikan tingkat positif atau negatifnya suatu kata, dengan nilai yang lebih tinggi menunjukkan sentimen yang kuat, baik positif maupun negatif.

Proses pelabelan data dilakukan dengan mencocokkan setiap kata dalam ulasan dengan kamus *Lexicon* Indonesia. Kamus ini menyediakan skor polaritas untuk berbagai kata, sehingga memungkinkan penilaian yang akurat terhadap sentimen keseluruhan dari ulasan. Jika kata tersebut ditemukan dalam kamus dan memiliki skor polaritas yang positif, maka ulasan akan diberi label sentimen positif. Sebaliknya, jika kata tersebut memiliki skor polaritas negatif, ulasan akan diberi label sentimen negatif.

Untuk menentukan label keseluruhan dari suatu ulasan, skor polaritas dari semua kata dalam ulasan di jumlahkan. Suatu ulasan masuk ke dalam kategori positif apabila skor polaritasnya adalah lebih dari sama dengan 0, jika skor polaritas di bawah 0 maka ulasan tersebut termasuk kategori ulasan negatif. Proses pelabelan data akan menghasilkan *dataset* ulasan yang sudah memiliki label positif ataupun negatif pada setiap ulasannya dan akan disimpan dengan format CSV. Proses ini tidak hanya meningkatkan kualitas data yang digunakan untuk analisis tetapi juga memastikan bahwa model yang dilatih memiliki data yang terstruktur dan berlabel dengan baik. Tahapan proses pelabelan data dapat dilihat pada gambar 3. 13 di atas.

3.3.3. Pembagian Data Latih dan Data Uji

Tahapan terakhir pada fase pemodelan data adalah pembagian data latih dan data uji atau *splitting* data



Gambar 3. 14 Pembagian Data Latih dan Data Uji

Proses ini bertujuan untuk memisahkan data yang akan digunakan sebagai data latih dan data yang akan digunakan sebagai data uji. Data latih digunakan untuk melatih model agar dapat mempelajari pola-pola yang ada dalam data. Sedangkan, data uji digunakan untuk menguji kinerja model yang telah dilatih, dengan menggunakan data uji yang tidak digunakan dalam pelatihan maka dapat mengukur kinerja model dalam melakukan analisis. *Dataset* akan dibagi menjadi tiga rasio yang berbeda Pembagian data dilakukan dengan menggunakan parameter ‘stratify’ memastikan bahwa distribusi kelas di data uji mirip dengan distribusi kelas di seluruh *dataset.* ‘stratify’ merupakan parameter yang digunakan untuk memastikan distribusi kelas di data latih dan data uji akan serupa dengan distribusi kelas di seluruh *dataset.* Misalkan jika distribusi kelas di seluruh *dataset* adalah 70% positif dan 30% negatif maka menggunakan parameter ‘stratify’ akan memastikan bahwa data latih dan data uji yang dibagi pada perbandingan 60 : 40 tetap akan memiliki 70% positif dan 30% negatif. Penjelasan terkait pembagian *dataset* menjadi tiga rasio berbeda dapat dilihat pada tabel 3. 2 di bawah.

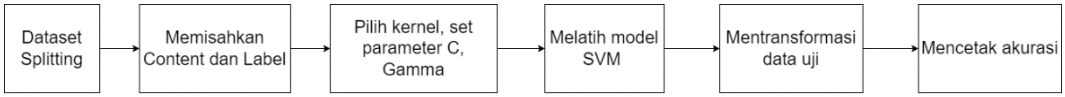
Tabel 3. 2 Pembagian data latih dan data uji

Perbandingan Data Latih dan Data Uji (%)	**Alasan**
60 : 40	Mengetahui kinerja metode dalam melakukan analisis dengan data latih yang minimal dan banyak data untuk diuji sehingga dapat melihat kemampuan metode dalam bekerja pada data yang belum pernah dilihat sebelumnya.
70 : 30	Merupakan rasio yang memberikan keseimbangan, dengan memberikan metode cukup data untuk belajar dan tetap memiliki cukup data untuk evaluasi.
80 : 20	Memberikan model lebih banyak data untuk belajar, untuk mengetahui kinerja metode ketika sudah diberikan data latih yang besar.

Proses pembagian data dilakukan dengan menggunakan *library 'scikit-learn'* pada *library* tersebut terdapat sebuah fungsi yang bernama *'train_test_split'* yang digunakan untuk membagai *dataset* menjadi dua *subset* yang saling eksklusif, yaitu data pelatihan dan data uji. Tahapan proses pembagian data latih dan data uji dapat dilihat pada gambar 3.14 di atas.

3.4. Klasifikasi *Support Vector Machine*

Pada tahap ini, dilakukan proses pengklasifikasian data ulasan yang telah melalui tahap *preprocessing* dan pemodelan data. Proses pengklasifikasian dilakukan dengan menggunakan metode *SVM.*



Gambar 3. 15 Tahapan *Support Vector Machine*

Tahapan dimulai dengan pemilihan model *Support Vector Machine,* yang melibatkan pemilihan *kernel* dan penyesuaian parameter yang sesuai dengan data.

Selanjutnya, model *Support Vector Machine* dilatih menggunakan data latih dan vektor fitur yang telah dibuat sebelumnya dan setelah itu metode akan menghasilkan nilai akurasi, atau gambaran tentang seberapa baik model dalam memprediksi data. Nilai akurasi didapatkan dengan menggunakan rumus :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Rumus (3.1)}$$

Berdasarkan rumus (3.1) dapat dilihat bahwa terdapat empat data yang mempengaruhi perhitungan yaitu *True Positive* (TP), *True Negative* (TN), *False Positve* (FP), dan *False Negative* (FN). *True Positive* (TP) adalah ulasan yang sebenarnya positif dan berhasil diprediksi sebagai positif, *True Negative* (TN) adalah ulasan yang sebenarnya negatif dan berhasil diprediksi sebagai negatif, sedangkan *False Negative* (FN) adalah ulasan yang seharusnya positif tetapi diprediksi sebagai negatif, dan *False Positive* (FP) adalah ulasan yang sebenarnya negatif tetapi salah diprediksi sebagai positif.

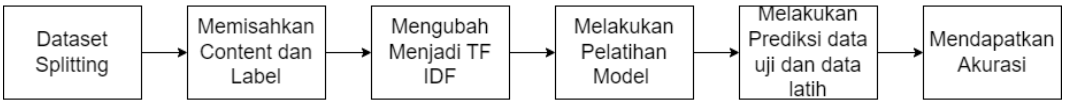
Support Vector Machine akan dilatih dengan menggunakan data latih, data akan digunakan untuk menentukan pola dan menentukan batas pemisah yang memaksimalkan margin antara kelas yang berbeda, yaitu kelas positif dan negatif. Pemaksimalan *margin* untuk memastikan bahwa *hyperlane* berada pada posisi optimal yang memisahkan kedua kelas dengan jelas. Proses persamaan *SVM* dapat dilihat pada rumus (2.1), sedangkan proses klasifikasi dapat dilihat pada rumus (2.2)

Support Vector Machine memiliki kemampuan yang sangat baik dalam mengklasifikasikan data teks, keunggulan *SVM* terletak pada kemampuannya menangani data dalam ruang berdimensi tinggi dan memberikan margin pemisahan yang maksimal antara kelas-kelas yang berbeda. *SVM* dalam menjalankan prosesnya membutuhkan *kernel* untuk mengubah data menjadi bentuk yang lebih tinggi sehingga menjadi lebih mudah untuk dipisahkan, pada pemrosesan ini *kernel Radial Basis Function* (RBF) dipilih karena memiliki fleksibilitas yang tinggi dan seringkali memberikan hasil yang lebih baik. Kernel RBF dapat menangkap kompleksitas dalam data dengan mengukur jarak antara titik-titik data menggunakan *Gaussian Function* setelah melakukan pemilihan kernel selanjutnya akan dilakukan penyesuaian parameter agar mengoptimalkan performa model *SVM*.

Terdapat dua parameter utama dalam *SVM* yaitu, C (*Regularization Parameter)* parameter tersebut untuk mengontrol *trade-off* antara memaksimalkan margin pemisahan dan mengurangi kesalahan klasifikasi, parameter kedua yaitu, Gamma (*Coefficient for RBF)* merupakan parameter yang digunakan untuk menentukan seberapa jauh pengaruh satu titik data. Nilai Gamma yang lebih tinggi berarti radius pengaruh setiap titik data lebih kecil, yang dapat menangkap detail lebih halus. Tahapan *Support Vector Machine* dapat dilihat pada gambar 3. 15 diatas.

3.5. Klasifikasi *Naive Bayes*

Pada tahap ini, dilakukan proses pengklasifikasian data ulasan yang telah melalui tahap *preprocessing* dan pemodelan data.



Gambar 3. 16 Tahapan *Naive Bayes*

Algoritma *Naive Bayes* melalui dua tahap klasifikasi teks, yaitu tahap pelatihan dan tahap klasifikasi. Pada tahap pelatihan, dilakukan analisis terhadap data latih untuk pemilihan kosakata, di mana kata-kata yang sering muncul dalam koleksi data latih dipilih sebagai representasi dokumen. Selanjutnya, probabilitas prioritas untuk setiap kategori ditentukan berdasarkan data latih. Pada tahap penelitian, nilai kategori dari suatu ulasan ditentukan berdasarkan *term* yang muncul dalam data uji yang akan diklasifikasikan dan setelah itu metode akan menghasilkan nilai akurasi. Nilai akurasi terhadap metode *Naive Bayes* didapatkan dengan menggunakan rumus :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Rumus (3.2)}$$

Berdasarkan rumus (3.2) dapat dilihat bahwa terdapat empat data yang mempengaruhi perhitungan yaitu *True Positive* (TP), *True Negative* (TN), *False Positve* (FP), dan *False Negative* (FN). *True Positive* (TP) adalah ulasan yang sebenarnya positif dan berhasil diprediksi sebagai positif, *True Negative* (TN) adalah ulasan yang sebenarnya negatif dan berhasil diprediksi sebagai negatif,

sedangkan *False Negative* (FN) adalah ulasan yang seharusnya positif tetapidiprediksi sebagai negatif, dan *False Positive* (FP) adalah ulasan yang sebenarnya negatif tetapi salah diprediksi sebagai positif.

Naive Bayes merupakan model yang memiliki keunggulan utama terletak pada kesederhanaannya, efisiensi komputasi, dan kemampuannya dalam menangani data teks yang besar dengan baik. *Naive Bayes* mengasumsikan bahwa kata-kata dalam teks bersifat independen satu sama lain dalam konteks kelas yang diberikan. Ini dikenal sebagai asumsi "naif: karena dalam kenyataannya, fitur-fitur mungkin saling bergantung, metode ini menghitung probabilitas posterior dari setiap kelas berdasarkan probabilitas *prior* dan probabilitas *likelihood* dari fitur-fitur dalam kelas tersebut. Probabilitas *prior* dari setiap kelas dihitung berdasarkan proporsi dokumen dalam setiap kelas pada data latih. Sebagai contoh, jika 60% ulasan dalam data latih adalah kelas positif, maka probabilitas *prior* kelas positif adalah 0.6. Sedangkan probabilitas *likelihood* dari setiap kata dalam setiap kelas dihitung berdasarkan frekuensi kemunculannya dalam kelas tersebut. Proses implementasi dari Teorema Bayes untuk menghitung probabilitas posterior yang dilakukan *Naive Bayes* dilakukan dengan menggunakan rumus (2.3), sedangkan proses untuk menentukan kelas yang memiliki probabilitas tertinggi dalam *dataset* dilakukan dengan menggunakan rumus (2.4). Untuk menghindari probabilitas nol ketika fitur tidak muncul dalam data latih suatu kelas, digunakan teknik *smoothing* dengan menambahkan nilai kecil *alpha*. Tahapan klasifikasi *Naive Bayes* dapat dilihat pada gambar 3. 16 di atas.

3.6. Evaluasi Hasil Dengan *Confusion Matrix Model SVM*

Setelah tahapan klasifikasi dengan *SVM* selesai dikerjakan, maka dilakukan evaluasi untuk mengetahui hasil kinerja metode tersebut. Evaluasi dilakukan dengan menggunakan tiga metrik utama yaitu *accuracy score, confusion matrix,* dan *classificatoin report*. Pertama, *accuracy score* dihitung menggunakan fungsi *accuracy_score(y_test, predictions),* yang memberikan persentase prediksi yang benar dari total prediksi. Akurasi ini memberikan gambaran umum tentang seberapa baik model memprediksi data uji secara keseluruhan.

confusion matrix dibuat dengan fungsi *confusion_matrix(y_test, predictions*), yang menghasilkan matriks 2x2 untuk masalah klasifikasi biner. Matriks ini terdiri dari empat komponen: *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Matriks ini membantu dalam memahami distribusi kesalahan prediksi yang dibuat oleh model, dengan menunjukkan berapa banyak contoh yang diklasifikasikan dengan benar atau salah dalam setiap kategori. Misalnya, jika terdapat banyak *false positive*, maka menunjukkan bahwa model terlalu sering mengklasifikasikan *instance* negatif sebagai positif.

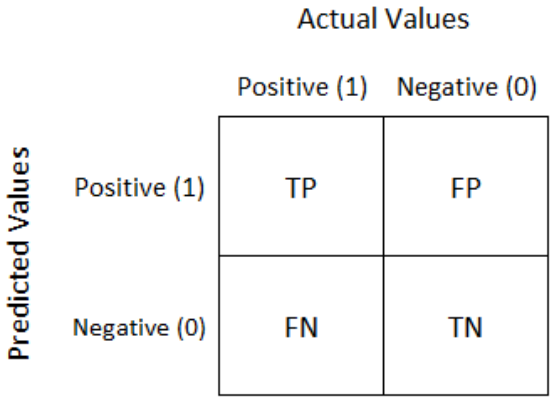
classification report diperoleh menggunakan fungsi *classification_report(y_test, predictions),* yang menyediakan metrik evaluasi yang lebih rinci termasuk *precision, recall*, dan F1-score untuk setiap kelas. *Precision* adalah rasio prediksi yang benar untuk kelas tertentu terhadap semua prediksi yang dibuat untuk kelas tersebut, sedangkan *recall* adalah rasio prediksi yang benar terhadap semua *instance* aktual dari kelas tersebut.

F1-score adalah rata-rata harmonis dari *precision* dan *recall*, memberikan gambaran keseimbangan antara keduanya. *Classification report* juga mencakup *support*, yang merupakan jumlah *instance* aktual di setiap kelas. *Precision* dan *recall* sering berlawanan, Jika *precision* meningkat maka akan mengurangi *recall* begitu juga sebaliknya. *F1-Score* menggabungkan kedua metrik ini menjadi satu nilai, untuk memberikan keseimbangan antara kemampuan model untuk mengidentifikasikan *instance* positif dengan benar atau *recall* dan meminimalkan kesalahan positif atau *precision. F1-Score* merupakan matrik evaluasi yang sangat berguna dalam memberikan gambaran yang lebih akurat yang berkaitan dengan kinerja model. Dengan menganalisis ketiga metrik ini secara komprehensif, maka dapat dilakukan evaluasi kinerja model *SVM* dengan lebih akurat, dan dapat memahami kekuatan dan kelemahan model.

Perbandingan yang dilakukan terhadap nilai akurasi, *precision, recall,* dan *F1-Score* pada metode *SVM* dapat menunjukkan keunggulan metode dan mengetahui kelemahan dari metode tersebut dalam melakukan analisis terhadap *dataset* ulasan Access by KAI,

3.6.1. Evaluasi Hasil *SVM* Dengan *Confusion Matrix* Rasio 60 : 40

Evaluasi yang dilakukan pada model *SVM* yang pertama adalah evaluasi pada pengujian yang dilakukan pada perbandingan data latih dan data uji 60 : 40. *Confusion Matrix* menghasilkan gambar yang terdiri dari empat persegi yang masing-masing persegi terdapat nilai. Nilai-nilai tersebut terdiri dari *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Contoh *Confusion Matrix* dapat dilihat pada gambar 3. 17 dibawah.



Gambar 3. 17 Contoh *Confusion Matrix*

Dapat dilihat pada gambar 3. 17 di atas, merupakan struktur dari *confusion matrix,* masing-masing bagian menunjukkan angka-angka mempresentasikan data yang benar diprediksi dan data yang tidak benar diprediksi oleh model. Pada bagian kiri bawah merupakan *True Positive* atau jumlah ulasan yang sebenarnya positif dan berhasil diprediksi positif oleh model, pada bagian kiri atas merupakan *True Negative* atau jumlah ulasan yang sebenarnya negatif dan berhasil diprediksi negatif oleh model, pada bagian kanan atas merupakan *False Positive* atau ulasan yang seharusnya negatif tetapi diprediksi sebagai positif dan pada bagian kanan bawah merupakan *False Negatif* atau ulasan yang seharusnya positif tetapi salah diprediksi sebagai negatif oleh model. Berdasarkan nilai *confusion matrix* tersebut maka dapat dilakukan perhitungan akurasi yang didapatkan oleh model, perhitungan terhadap akurasi dilakukan dengan menggunakan rumus :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Rumus (3.3)}$$

Nilai akurasi didapatkan dengan melakukan penjumlahan terhadap nilai yang berhasil diprediksi dengan benar dan dibagi dengan seluruh nilai yang ada.

Berdasarkan nilai akurasi tersebut makan dapat dilakukan pengukuran kinerja dari metode, perhitungan yang dapat dilakukan yaitu, melakukan perhitungan terhadap nilai *precision* yang mengukur akurasi dari prediksi positif yang sebenarnya benar. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus :

$$Precision = \frac{TP}{TP + FP} \qquad \text{Rumus (3.4)}$$

Precision dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua prediksi positif yaitu T*rue Positive* dan *False Negative.*

recall berguna untuk mengukur kemampuan model untuk menemukan semua kasus positif. Perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus :

$$Recall = \frac{TP}{TP + FN} \qquad \text{Rumus (3.5)}$$

Recall dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua data aktual positif.

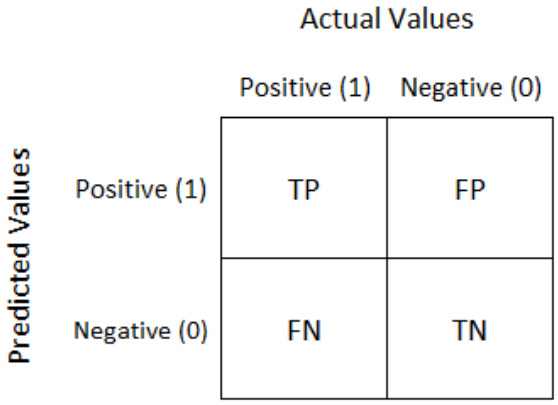
F1-Score yang merupakan ukuran gabungan yang menggabungkan *precision* dan *recall* menjadi satu nilai dengan cara harmonis. *F1-Score* memberikan gambaran yang seimbang antara *precision* dan *recal,* yang sangat penting dalam konteks klasifikasi, terutama ketika terdapat ketidakseimbangan kelas dalam *dateset*. Perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus :

$$F1 - Score = 2\,X\,\frac{Precision\ x\ Recall}{Precision + Recall} \qquad \text{Rumus (3.6)}$$

Ketiga nilai tersebut merupakan ukuran evaluasi kinerja terhadap metode *SVM* dalam melakukan analisis sentimen terhadap ulasan Access by KAI pada perbandingan data latih dan data uji 60 : 40. Dengan ketiga nilai tersebut maka dapat diketahui kualitas model *SVM* dalam melakukan analisis terhadap *dataset.*

3.6.2. Evaluasi Hasil *SVM* Dengan *Confusion Matrix* Rasio 70 : 30

Evaluasi yang kedua adalah evaluasi pada perbandingan data latih dan data uji 70 : 30. *Confusion Matrix* menghasilkan gambar yang terdiri dari empat persegi yang masing-masing persegi terdapat nilai. Nilai-nilai tersebut terdiri dari *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Contoh *Confusion Matrix* dapat dilihat pada gambar 3. 18 di bawah.



Gambar 3. 18 Contoh *Confusion Matrix*

Dapat dilihat pada gambar 3. 18 di atas, merupakan struktur dari *confusion matrix,* masing-masing bagian menunjukkan angka-angka mempresentasikan data yang benar diprediksi dan data yang tidak benar diprediksi oleh model. Pada bagian kiri bawah merupakan *True Positive* atau jumlah ulasan yang sebenarnya positif dan berhasil diprediksi positif oleh model, pada bagian kiri atas merupakan *True Negative* atau jumlah ulasan yang sebenarnya negatif dan berhasil diprediksi negatif oleh model, pada bagian kanan atas merupakan *False Positive* atau ulasan yang seharusnya negatif tetapi diprediksi sebagai positif dan pada bagian kanan bawah merupakan *False Negatif* atau ulasan yang seharusnya positif tetapi salah diprediksi sebagai negatif oleh model. Berdasarkan nilai *confusion matrix* tersebut maka dapat dilakukan perhitungan akurasi yang didapatkan oleh model, perhitungan terhadap akurasi dilakukan dengan menggunakan rumus :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Rumus (3.7)}$$

Nilai akurasi didapatkan dengan melakukan penjumlahan terhadap nilai yang berhasil diprediksi dengan benar dan dibagi dengan seluruh nilai yang ada.

Berdasarkan nilai akurasi tersebut makan dapat dilakukan pengukuran kinerja dari metode, perhitungan yang dapat dilakukan yaitu, melakukan perhitungan terhadap nilai *precision* yang mengukur akurasi dari prediksi positif yang sebenarnya benar. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus :

$$Precision = \frac{TP}{TP + FP} \qquad \text{Rumus (3.8)}$$

Precision dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua prediksi positif yaitu T*rue Positive* dan *False Negative.*

recall berguna untuk mengukur kemampuan model untuk menemukan semua kasus positif. Perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus :

$$Recall = \frac{TP}{TP + FN} \qquad \text{Rumus (3.9)}$$

Recall dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua data aktual positif.

F1-Score yang merupakan ukuran gabungan yang menggabungkan *precision* dan *recall* menjadi satu nilai dengan cara harmonis. *F1-Score* memberikan gambaran yang seimbang antara *precision* dan *recall,* ukuran ini menjadi sangat penting ketika terdapat ketidakseimbangan kelas dalam *dataset*. Perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus :

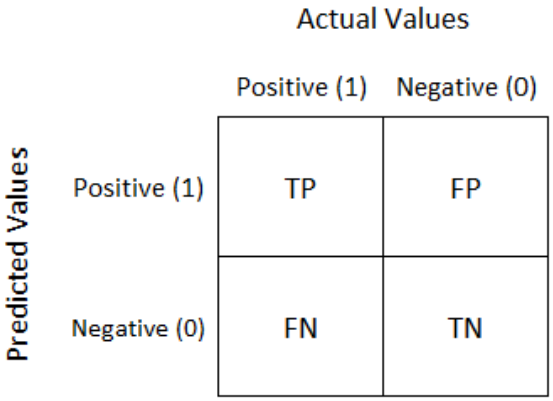
$$F1 - Score = 2\,X\frac{Precision\ x\ Recall}{Precision + Recall} \qquad \text{Rumus (3.10)}$$

Ketiga nilai tersebut merupakan ukuran evaluasi kinerja terhadap metode *SVM* dalam melakukan analisis sentimen terhadap ulasan Access by KAI pada perbandingan data latih dan data uji 70 : 30%.

3.6.3. Evaluasi Hasil *SVM* Dengan *Confusion Matrix* Rasio 80 : 20

Evaluasi yang ketiga adalah evaluasi pada pengujian dengan perbandingan data latih dan data uji 80 : 20. Pengujian ini dilakukan untuk mengetahui kinerja

model ketika menggunakan data latih dalam jumlah yang sangat besar dan data uji yang lebih sedikit. *Confusion Matrix* menghasilkan gambar yang terdiri dari empat persegi yang masing-masing persegi terdapat nilai. Nilai-nilai tersebut terdiri dari *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Contoh *Confusion Matrix* dapat dilihat pada gambar 3. 19 di bawah.



Gambar 3. 19 Contoh *Confusion Matrix*

Dapat dilihat pada gambar 3. 19 di atas, merupakan struktur dari *confusion matrix,* masing-masing bagian menunjukkan angka-angka mempresentasikan data yang benar diprediksi dan data yang tidak benar diprediksi oleh model. Pada bagian kiri bawah merupakan *True Positive* atau jumlah ulasan yang sebenarnya positif dan berhasil diprediksi positif oleh model, pada bagian kiri atas merupakan *True Negative* atau jumlah ulasan yang sebenarnya negatif dan berhasil diprediksi negatif oleh model, pada bagian kanan atas merupakan *False Positive* atau ulasan yang seharusnya negatif tetapi diprediksi sebagai positif dan pada bagian kanan bawah merupakan *False Negatif* atau ulasan yang seharusnya positif tetapi salah diprediksi sebagai negatif oleh model. Berdasarkan nilai *confusion matrix* tersebut maka dapat dilakukan perhitungan akurasi yang didapatkan oleh model, perhitungan terhadap akurasi dilakukan dengan menggunakan rumus :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Rumus (3.11)}$$

Nilai akurasi didapatkan dengan melakukan penjumlahan terhadap seluruh nilai yang berhasil diprediksi dengan benar dan dibagi dengan seluruh nilai yang ada.

Berdasarkan nilai akurasi tersebut makan dapat dilakukan pengukuran kinerja dari metode, perhitungan yang dapat dilakukan yaitu, melakukan perhitungan terhadap nilai *precision* yang mengukur akurasi dari prediksi positif yang sebenarnya benar. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus :

$$Precision = \frac{TP}{TP + FP} \qquad \text{Rumus (3.12)}$$

Precision dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua prediksi positif yaitu T*rue Positive* dan *False Negative.*

recall berguna untuk mengukur kemampuan model untuk menemukan semua kasus positif. Perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus :

$$Recall = \frac{TP}{TP + FN} \qquad \text{Rumus (3.13)}$$

Recall dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua data aktual positif.

F1-Score yang merupakan ukuran gabungan yang menggabungkan *precision* dan *recall* menjadi satu nilai dengan cara harmonis. *F1-Score* memberikan gambaran yang seimbang antara *precision* dan *recall,* ukuran ini memberikan gambaran yang seimbang antara *precision* dan *recall,* yang sangat penting dalam konteks klasifikasi, terutama ketika terdapat ketidakseimbangan kelas data dalam *dataset*. Perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus :

$$F1 - Score = 2\,X\frac{Precision\ x\ Recall}{Precision + Recall} \qquad \text{Rumus (3.14)}$$

Ketiga nilai tersebut merupakan ukuran evaluasi kinerja terhadap metode *SVM* dalam melakukan analisis sentimen terhadap ulasan Access by KAI pada perbandingan data latih dan data uji 80 : 20. Dengan menggunakan ketiga nilai evaluasi tersebut maka dapat diketahui kinerja model dalam melakukan analisis terhadap *dataset* ulasan Access by KAI.

3.7. Evaluasi Hasil Dengan *Confusion Matrix* Model *Naive Bayes*

Setelah tahapan klasifikasi dengan *Naive Bayes* selesai dikerjakan, dilakukan evaluasi untuk mengetahui kinerja metode tersebut. Evaluasi dilakukan dengan menggunakan tiga metrik utama *yaitu accuracy score, confusion matrix,* dan *classification report.*

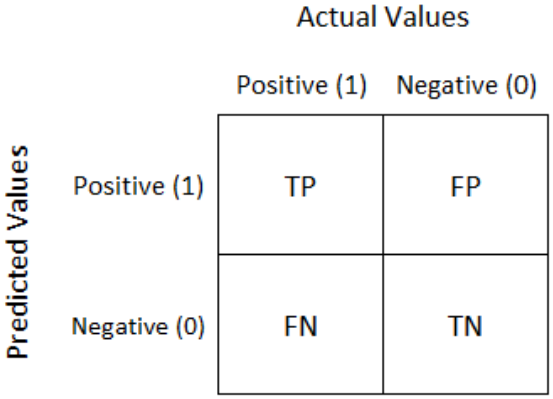
Pertama, *accuracy score* dihitung menggunakan fungsi *accuracy_score(y_test, predictions*), yang memberikan persentase prediksi yang benar dari total prediksi. Akurasi skor adalah metrik yang paling umum digunakan untuk emngukur kinerja model klasifikasi. Metrik ini memberikan persentase prediksi yang benar dari total prediksi yang dibuat oleh model. Dengan kata lain, akurasi skor menunjukkan seberapa banyak *instance* yang berhasil diprediksi dengan benar oleh model dari seluruh *instance* yang ada.

Kedua, *confusion matrix* dibuat dengan fungsi *confusion_matrix(y_test, predictions)*, yang menghasilkan matriks 2x2 untuk masalah klasifikasi biner. Matriks ini terdiri dari empat komponen: *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Matriks ini membantu dalam memahami distribusi kesalahan prediksi yang dibuat oleh model, dengan menunjukkan berapa banyak contoh yang diklasifikasikan dengan benar atau salah dalam setiap kategori. *Consuion Matrix* sangat berguna untuk mengidentifikasi dan memahami kesalahan spesifik yang dibuat oleh model.

Terakhir, *classification report* diperoleh menggunakan fungsi *classification_report(y_test, predictions),* yang menyediakan metrik evaluasi yang lebih rinci termasuk precision, recall, dan *F1-score* untuk setiap kelas. *Precision* adalah rasio prediksi yang benar untuk kelas tertentu terhadap semua prediksi yang dibuat untuk kelas tersebut, sedangkan *recall* adalah rasio prediksi yang benar terhadap semua *instance* aktual dari kelas tersebut. F1-score adalah rata-rata harmonis dari *precision* dan *recall*, memberikan gambaran keseimbangan antara keduanya. *Classification report* juga mencakup *support,* yang merupakan jumlah *instance* aktual di setiap kelas. Evaluasi yang dilakukan dengan *classification report* memberikan pandangan yang lebih mendalam tentang kinerja model *Naive Bayes* dalam menangani klasifikasi sentimen secara menyeluruh.

3.7.1. Evaluasi Hasil *Naive Bayes* Dengan *Confusion Matrix* Rasio 60 : 40

Evaluasi pertama pada model *Naive Bayes* dilakukan dengan perbandingan data latih dan data uji 60 : 40. Pengujian ini bertujuan untuk menilai kinerja model saat data latih yang tersedia relatif sedikit, sementara data uji lebih banyak. Pengujian ini dilakukan untuk mengetahui kinerja model ketika menggunakan data latih dalam jumlah yang sangat besar dan data uji yang lebih sedikit. *Confusion Matrix* menghasilkan gambar yang terdiri dari empat persegi yang masing-masing persegi terdapat nilai. Nilai-nilai tersebut terdiri dari *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Contoh *Confusion Matrix* dapat dilihat pada gambar 3. 20 di bawah.



Gambar 3. 20 Contoh *Confusion Matrix*

Dapat dilihat pada gambar 3. 20 di atas, merupakan struktur dari *confusion matrix,* masing-masing bagian menunjukkan angka-angka mempresentasikan data yang benar diprediksi dan data yang tidak benar diprediksi oleh model. Pada bagian kiri bawah merupakan *True Positive* atau jumlah ulasan yang sebenarnya positif dan berhasil diprediksi positif oleh model, pada bagian kiri atas merupakan *True Negative* atau jumlah ulasan yang sebenarnya negatif dan berhasil diprediksi negatif oleh model, pada bagian kanan atas merupakan *False Positive* atau ulasan yang seharusnya negatif tetapi diprediksi sebagai positif dan pada bagian kanan bawah merupakan *False Negatif* atau ulasan yang seharusnya positif tetapi salah diprediksi sebagai negatif oleh model. Berdasarkan nilai *confusion matrix* tersebut maka dapat dilakukan perhitungan akurasi yang didapatkan oleh model, perhitungan terhadap akurasi dilakukan dengan menggunakan rumus :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Rumus (3.15)}$$

Nilai akurasi didapatkan dengan melakukan penjumlahan terhadap nilai yang berhasil diprediksi dengan benar dan dibagi dengan seluruh nilai yang ada.

Berdasarkan nilai akurasi tersebut makan dapat dilakukan pengukuran kinerja dari metode, perhitungan yang dapat dilakukan yaitu, melakukan perhitungan terhadap nilai *precision* yang mengukur akurasi dari prediksi positif yang sebenarnya benar. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus :

$$Precision = \frac{TP}{TP + FP} \qquad \text{Rumus (3.16)}$$

Precision dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua prediksi positif yaitu T*rue Positive* dan *False Negative.*

recall berguna untuk mengukur kemampuan model untuk menemukan semua kasus positif. Perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus :

$$Recall = \frac{TP}{TP + FN} \qquad \text{Rumus (3.17)}$$

Recall dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua data aktual positif.

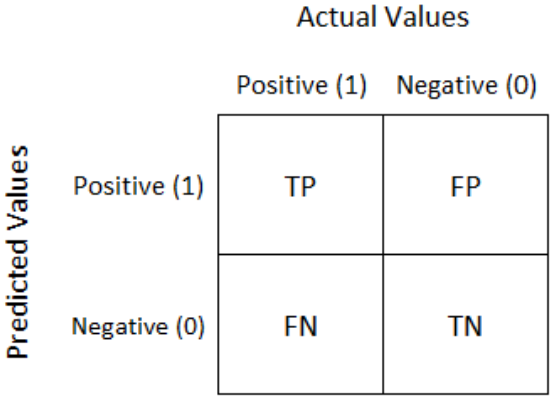
F1-Score memberikan gambaran yang seimbang antara *precision* dan *recall,* ukuran ini memberikan gambaran yang seimbang antara *precision* dan *recall,* yang sangat penting dalam konteks klasifikasi, terutama ketika terdapat ketidakseimbangan kelas data dalam *dataset*. Perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus :

$$F1 - Score = 2\,X\,\frac{Precision\ x\ Recall}{Precision + Recall} \qquad \text{Rumus (3.18)}$$

Ketiga nilai tersebut merupakan ukuran evaluasi kinerja terhadap metode *SVM* dalam melakukan analisis sentimen terhadap ulasan Access by KAI pada perbandingan data latih dan data uji 60 : 40.

3.7.2. Evaluasi Hasil *Naive Bayes* Dengan *Confusion Matrix* Rasio 70 : 30

Evaluasi kedua melibatkan pengujian model dengan perbandingan data latih dan data uji 70 : 30. Dalam skenario ini, jumlah data latih lebih banyak dibandingkan data uji. Tujuannya adalah untuk mengamati apakah peningkatan jumlah data latih dapat memperbaiki performa model dan bagaimana model merespons dengan data uji yang lebih sedikit dibandingkan dengan perbandingan 60 : 40. *Confusion Matrix* menghasilkan gambar yang terdiri dari empat persegi yang masing-masing persegi terdapat nilai. Nilai-nilai tersebut terdiri dari *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Contoh *Confusion Matrix* dapat dilihat pada gambar 3. 21 di bawah.



Gambar 3. 21 Contoh *Confusion Matrix*

Dapat dilihat pada gambar 3. 21 di atas, merupakan struktur dari *confusion matrix,* masing-masing bagian menunjukkan angka-angka mempresentasikan data yang benar diprediksi dan data yang tidak benar diprediksi oleh model. Pada bagian kiri bawah merupakan *True Positive* atau jumlah ulasan yang sebenarnya positif dan berhasil diprediksi positif oleh model, pada bagian kiri atas merupakan *True Negative* atau jumlah ulasan yang sebenarnya negatif dan berhasil diprediksi negatif oleh model, pada bagian kanan atas merupakan *False Positive* atau ulasan yang seharusnya negatif tetapi diprediksi sebagai positif dan pada bagian kanan bawah merupakan *False Negatif* atau ulasan yang seharusnya positif tetapi salah diprediksi sebagai negatif oleh model. Berdasarkan nilai *confusion matrix* tersebut maka dapat dilakukan perhitungan akurasi yang didapatkan oleh model, perhitungan terhadap akurasi dilakukan dengan menggunakan rumus :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Rumus (3.19)}$$

Nilai akurasi didapatkan dengan melakukan penjumlahan terhadap nilai yang berhasil diprediksi dengan benar dan dibagi dengan seluruh nilai yang ada.

Berdasarkan nilai akurasi tersebut makan dapat dilakukan pengukuran kinerja dari metode, perhitungan yang dapat dilakukan yaitu, melakukan perhitungan terhadap nilai *precision* yang mengukur akurasi dari prediksi positif yang sebenarnya benar. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus :

$$Precision = \frac{TP}{TP + FP} \qquad \text{Rumus (3.20)}$$

Precision dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua prediksi positif yaitu T*rue Positive* dan *False Negative.*

recall berguna untuk mengukur kemampuan model untuk menemukan semua kasus positif. Perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus :

$$Recall = \frac{TP}{TP + FN} \qquad \text{Rumus (3.21)}$$

Recall dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua data aktual positif.

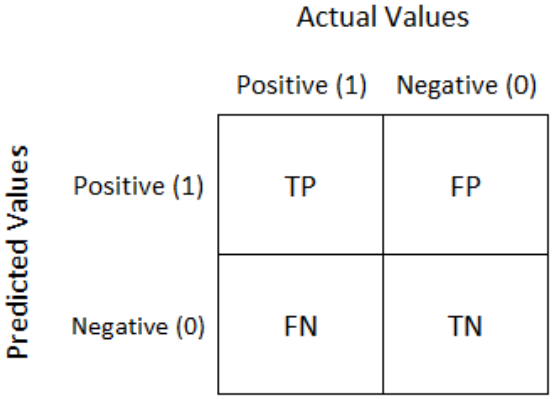
F1-Score memberikan gambaran yang seimbang antara *precision* dan *recall,* ukuran ini memberikan gambaran yang seimbang antara *precision* dan *recall,* yang sangat penting dalam konteks klasifikasi, terutama ketika terdapat ketidakseimbangan kelas data dalam *dataset*. Perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus :

$$F1 - Score = 2\,X\,\frac{Precision\ x\ Recall}{Precision + Recall} \qquad \text{Rumus (3.22)}$$

Ketiga nilai tersebut merupakan ukuran evaluasi kinerja terhadap metode *SVM* dalam melakukan analisis sentimen terhadap ulasan Access by KAI pada perbandingan data latih dan data uji 70 : 30.

3.7.3. Evaluasi Hasil *Naive Bayes* Dengan *Confusion Matrix* rasio 80 : 20

Evaluasi ketiga menggunakan perbandingan data latih dan data uji 80 : 20. Di sini, model dilatih dengan jumlah data yang sangat besar, sementara data uji jauh lebih sedikit. Pengujian ini dilakukan untuk mengevaluasi apakah peningkatan yang signifikan dalam jumlah data latih dapat secara substansial meningkatkan kinerja model dan apakah model tetap akurat dengan data uji yang terbatas. *Confusion Matrix* menghasilkan gambar yang terdiri dari empat persegi yang masing-masing persegi terdapat nilai. Nilai-nilai tersebut terdiri dari *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Contoh *Confusion Matrix* dapat dilihat pada gambar 3. 22 di bawah.



Gambar 3. 22 Contoh *Confusion Matrix*

Dapat dilihat pada gambar 3. 22 di atas, merupakan struktur dari *confusion matrix,* masing-masing bagian menunjukkan angka-angka mempresentasikan data yang benar diprediksi dan data yang tidak benar diprediksi oleh model. Pada bagian kiri bawah merupakan *True Positive* atau jumlah ulasan yang sebenarnya positif dan berhasil diprediksi positif oleh model, pada bagian kiri atas merupakan *True Negative* atau jumlah ulasan yang sebenarnya negatif dan berhasil diprediksi negatif oleh model, pada bagian kanan atas merupakan *False Positive* atau ulasan yang seharusnya negatif tetapi diprediksi sebagai positif dan pada bagian kanan bawah merupakan *False Negatif* atau ulasan yang seharusnya positif tetapi salah diprediksi sebagai negatif oleh model. Berdasarkan nilai *confusion matrix* tersebut maka dapat dilakukan perhitungan akurasi yang didapatkan oleh model, perhitungan terhadap akurasi dilakukan dengan menggunakan rumus :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Rumus (3.23)}$$

Nilai akurasi didapatkan dengan melakukan penjumlahan terhadap nilai yang berhasil diprediksi dengan benar dan dibagi dengan seluruh nilai yang ada.

Berdasarkan nilai akurasi tersebut makan dapat dilakukan pengukuran kinerja dari metode, perhitungan yang dapat dilakukan yaitu, melakukan perhitungan terhadap nilai *precision* yang mengukur akurasi dari prediksi positif yang sebenarnya benar. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus :

$$Precision = \frac{TP}{TP + FP} \qquad \text{Rumus (3.24)}$$

Precision dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua prediksi positif yaitu T*rue Positive* dan *False Negative.*

recall berguna untuk mengukur kemampuan model untuk menemukan semua kasus positif. Perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus :

$$Recall = \frac{TP}{TP + FN} \qquad \text{Rumus (3.25)}$$

Recall dihitung sebagai perbandingan antara *True Positive* (TP) dengan jumlah semua data aktual positif.

F1-Score memberikan gambaran yang seimbang antara *precision* dan *recall,* ukuran ini memberikan gambaran yang seimbang antara *precision* dan *recall,* yang sangat penting dalam konteks klasifikasi, terutama ketika terdapat ketidakseimbangan kelas data dalam *dataset*. Perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus :

$$F1 - Score = 2\,X\,\frac{Precision\ x\ Recall}{Precision + Recall} \qquad \text{Rumus (3.26)}$$

Ketiga nilai tersebut merupakan ukuran evaluasi kinerja terhadap metode *SVM* dalam melakukan analisis sentimen terhadap ulasan Access by KAI pada perbandingan data latih dan data uji 80 : 20.

3.8. Membandingkan *Support Vector Machine* dan *Naive Bayes*

Tahapan terakhir dari penelitian ini adalah melakukan perbandingan metode *Support Vector Machine* dan *Naive Bayes.* Perbandingan dilakukan berdasarkan nilai akurasi dari masing-masing metode dengan tiga rasio perbandingan data latih dan data uji yang berbeda, nilai *precision,* nilai *recall,* dan nilai F1-*Score,* serta kelas sentimen berdasarkan *true positive* dan *true negative.*

4. HASIL DAN PEMBAHASAN

4.1. *Dataset*

Hasil dari akuisisi data pada ulasan aplikasi Access by KAI di Google Play Store telah berhasil mengumpulkan total 3555 data ulasan yang keseluruhan datanya menggunakan bahasa Indonesia. Contoh hasil akuisisi data dapat dilihat pada tabel 4. 1 di bawah.

Tabel 4. 1 Hasil Akuisisi Data

Tanggal	***Content***
1/14/2024 7:46:53	Tetiba sering logout sendiri, bukanya lama, akses login sering error. Lebih simple aplikasi yang dulu.
2/23/2024 12:53:40	Aplikasinya berat, mending buat kya yg versi lama aja sat set ringan
11/30/2023 11:32:11	Sering suruh update tapi sering juga errornya :(
3/7/2024 6:29:26	Aplikasi lemot kalo lagi pesen kereta
2/28/2024 7:06:04	Mau kasih bintang 100/10 deh untuk access by KAI nih ,sangat membantu sekali untuk semua keluhan dan saran , customer servis nya juga sangat ramah dan baik dalam menjelaskan cara pembatalan keberangkatan dan ubah jadwal keberangkatan kai bagus banget deh

Contoh rinci dari hasil akuisisi data ini dapat dilihat pada tabel 4.1 di atas. Dalam tabel tersebut, akan terdapat tanggal yang merupakan informasi kapan ulasan tersebut diberikan, terdapat pula informasi mengenai jam ulasan tersebut diberikan, serta *content* yang merupakan ulasan dari pengguna Google Play Store mengenai pengalaman mereka dalam menggunakan aplikasi Access by KAI.

4.2. Hasil Data *Preprocessing*

Hasil dari *preprocessing* adalah data yang telah terstruktur dan siap untuk analisis sentimen. *Preprocessing* menghasilkan *tokens, lower case text, cleaning, normalization, stopword removal, stemming.* Inti dari keseluruhan hasil proses *preprocessing* adalah memastikan bahwa data siap digunakan untuk analisis lebih lanjut dengan kualitas yang baik dan *noise* yang minimal.

4.2.1. *Tokens*

Berdasarkan contoh yang telah diberikan, berikut merupakan hasil proses *tokenizing* yang telah mengubah ulasan pada *dataset* menjadi berbentuk *tokens*. Contoh hasil *tokenizing* dapat dilihat pada tabel 4. 2 di bawah.

Tabel 4. 2 *Tokenizing*

Sebelum *tokenizing*	**Setelah *tokenizing***
Tetiba sering logout sendiri, bukanya lama, akses login sering error. Lebih simple aplikasi yang dulu.	['Tetiba', 'sering', 'logout', 'sendiri,', 'bukanya', 'lama', ',', 'akses', 'login', 'sering', 'error', '.', 'lebih', 'simple', 'aplikasi', 'yang', 'dulu.']
Aplikasinya berat, mending buat kya yg versi lama aja sat set ringan	['Aplikasinya', 'berat,', 'mending', 'buat', 'kya', 'yg', 'versi', 'lama', 'aja', 'sat', 'set', 'ringan']
Sering suruh update tapi sering juga errornya :(	['Sering', 'suruh', 'update', 'tapi', 'sering', 'juga', 'errornya', ':', '(']
Aplikasi lemot kalo lagi pesen kereta	['Aplikasi', 'lemot', 'kalo', 'lagi', 'pesen', 'kereta']
Mau kasih bintang 100/10 deh untuk access by KAI nih ,sangat membantu sekali untuk semua keluhan dan saran , customer servis nya juga sangat ramah dan baik dalam menjelaskan cara	['Mau', 'kasih', 'bintang', '100/10', 'deh', 'untuk', 'access', 'by', 'KAI', 'nih', ',', 'sangat', 'membantu', 'sekali', 'untuk', 'semua', 'keluhan', 'dan', 'saran', ',', 'customer', 'servis', 'nya', 'juga', 'sangat',

Sebelum *tokenizing*	**Setelah *tokenizing***
pembatalan keberangkatan dan ubah jadwal keberangkatan KAI bagus banget deh	'ramah', 'dan', 'baik', 'dalam', 'menjelaskan', 'cara', 'pembatalan', 'keberangkatan', 'dan', 'ubah', 'jadwal', 'keberangkatan', 'KAI', 'bagus', 'banget', 'deh']

Dalam tabel 4.2 di atas terlihat bahwa setelah melewati proses *tokenizing,* ulasan-ulasannya dipisahkan menjadi kata-kata individu. Setiap kata dalam ulasan tersebut diapit oleh tanda kutip untuk menandai batasan kata, sementara di antara satu kata dengan kata lainnya dipisahkan oleh koma.

4.2.2. *Lowercased Text*

Hasil dari *case folding* menghasilkan seluruh ulasan yang terdapat dalam *dataset* telah diubah menjadi *lowercased*. Berikut merupakan hasil dari *case folding* yang telah memanfaatkan *library pandas* untuk menyeragamkan seluruh ulasan dalam *lowercased.* Contoh hasil *tokenizing* dapat dilihat pada tabel 4. 3 di bawah.

Tabel 4. 3 *Case Folding*

Sebelum *Case Folding*	**Setelah *Case Folding***
['Tetiba', 'sering', 'logout', 'sendiri', ‘,’, 'bukanya', 'lama', ‘,’, 'akses', 'login', 'sering', 'error', ‘.’, 'lebih', 'simple', 'aplikasi', 'yang', 'dulu.']	['tetiba', 'sering', 'logout', 'sendiri', ‘,’, 'bukanya', 'lama', ‘,’, 'akses', 'login', 'sering', 'error', ‘.’, 'lebih', 'simple', 'aplikasi', 'yang', 'dulu.']
['Aplikasinya', 'berat,', 'mending', 'buat', 'kya', 'yg', 'versi', 'lama', 'aja', 'sat', 'set', 'ringan']	['aplikasinya', 'berat,', 'mending', 'buat', 'kya', 'yg', 'versi', 'lama', 'aja', 'sat', 'set', 'ringan']
['Sering', 'suruh', 'update', 'tapi', 'sering', 'juga', 'errornya', ':’, ‘(']	['sering', 'suruh', 'update', 'tapi', 'sering', 'juga', 'errornya', ':’, ‘(']

Sebelum *Case Folding*	**Setelah *Case Folding***
['Aplikasi', 'lemot', 'kalo', 'lagi', 'pesen', 'kereta']	['aplikasi', 'lemot', 'kalo', 'lagi', 'pesen', 'kereta']
['Mau', 'kasih', 'bintang', '100/10', 'deh', 'untuk', 'access', 'by', 'KAI', 'nih', ',sangat', 'membantu', 'sekali', 'untuk', 'semua', 'keluhan', 'dan', 'saran', ',customer', 'servis', 'nya', 'juga', 'sangat', 'ramah', 'dan', 'baik', 'dalam', 'menjelaskan', 'cara', 'pembatalan', 'keberangkatan', 'dan', 'ubah', 'jadwal', 'keberangkatan', 'KAI', 'bagus', 'banget', 'deh']	['mau', 'kasih', 'bintang', '100/10', 'deh', 'untuk', 'access', 'by', 'kai', 'nih', ',sangat', 'membantu', 'sekali', 'untuk', 'semua', 'keluhan', 'dan', ',saran', 'customer', 'servis', 'nya', 'juga', 'sangat', 'ramah', 'dan', 'baik', 'dalam', 'menjelaskan', 'cara', 'pembatalan', 'keberangkatan', 'dan', 'ubah', 'jadwal', 'keberangkatan', 'kai', 'bagus', 'banget', 'deh']

Dalam tabel 4.3 di atas terlihat bahwa setelah melewati tahap *case folding,* seluruh huruf yang ada pada ulasan telah berubah menjadi huruf kecil, dan ulasan tetap berada dalam individu-individu atau *token*. Proses *case folding* dilakukan setelah melakukan *tokenizing,* sehingga struktur ulasan yang sudah dipecah menjadi kata-kata individu tetap dipertahankan.

4.2.3. *Cleaned Text*

Dataset yang telah berbentuk *tokens* dan sudah tidak terdapat huruf besar akan dilanjutkan dengan melakukan *cleaning,* hasil dari *cleaning* adalah *cleaned text*. Berikut merupakan hasil dari pemanfaatan *library "re"* untuk melakukan *cleaning.* Contoh hasil *cleaning* dapat dilihat pada tabel 4. 4 di bawah.

Tabel 4. 4 *Cleaning*

Sebelum *Cleaning*	**Setelah *Cleaning***
['tetiba', 'sering', 'logout', 'sendiri,', 'bukanya', 'lama,', 'akses', 'login',	['tetiba', 'sering', 'logout', 'sendiri', 'bukanya', 'lama', 'akses', 'login',

Sebelum *Cleaning*	Setelah *Cleaning*
'sering', 'error.', 'lebih', 'simple', 'aplikasi', 'yang', 'dulu.']	'sering', 'error', 'lebih', 'simple', 'aplikasi', 'yang', 'dulu']
['aplikasinya', 'berat,', 'mending', 'buat', 'kya', 'yg', 'versi', 'lama', 'aja', 'sat', 'set', 'ringan']	['aplikasinya', 'berat', 'mending', 'buat', 'kya', 'yg', 'versi', 'lama', 'aja', 'sat', 'set', 'ringan']
['sering', 'suruh', 'update', 'tapi', 'sering', 'juga', 'errornya', ':(']	['sering', 'suruh', 'update', 'tapi', 'sering', 'juga', 'errornya']
['aplikasi', 'lemot', 'kalo', 'lagi', 'pesen', 'kereta']	['aplikasi', 'lemot', 'kalo', 'lagi', 'pesen', 'kereta']
['mau', 'kasih', 'bintang', '100/10', 'deh', 'untuk', 'access', 'by', 'kai', 'nih', ',sangat', 'membantu', 'sekali', 'untuk', 'semua', 'keluhan', 'dan', ',saran', 'customer', 'servis', 'nya', 'juga', 'sangat', 'ramah', 'dan', 'baik', 'dalam', 'menjelaskan', 'cara', 'pembatalan', 'keberangkatan', 'dan', 'ubah', 'jadwal', 'keberangkatan', 'kai', 'bagus', 'banget', 'deh']	['mau', 'kasih', 'bintang', 'deh', 'untuk', 'access', 'by', 'kai', 'nih', 'sangat', 'membantu', 'sekali', 'untuk', 'semua', 'keluhan', 'dan', 'saran', 'customer', 'servis', 'nya', 'juga', 'sangat', 'ramah', 'dan', 'baik', 'dalam', 'menjelaskan', 'cara', 'pembatalan', 'keberangkatan', 'dan', 'ubah', 'jadwal', 'keberangkatan', 'kai', 'bagus', 'banget', 'deh']

Dalam tabel 4.4 di atas terlihat bahwa setelah melewati *cleaning,* ulasan pada *dataset* kini hanya tersisa huruf-huruf, karena seluruh angka, tanda baca, dan karakter non-alfabet lainnya yang terdapat di dalam *dataset* sudah dihilangkan. Selain itu spasi berlebih yang terdapat di ulasan juga sudah dihilangkan.

4.2.4. *Normalized Text*

Berikut merupakan hasil dari pemanfaatan kamus *lexicon* untuk melakukan perubahan kata *slang* menjadi kata yang sesuai KBBI atau bernama n*ormalization* yang menghasilkan *normalized text*. Contoh dari hasil *Normalization* dapat dilihat pada tabel 4.5 di bawah.

Tabel 4. 5 Normalization

Sebelum *Normalizaton*	**Setelah *Normalizaton***
['tetiba', 'sering', 'logout', 'sendiri', 'bukanya', 'lama', 'akses', 'login', 'sering', 'error', 'lebih', 'simple', 'aplikasi', 'yang', 'dulu']	['tiba-tiba', 'sering', 'keluar', 'sendiri', 'bukanya', 'lama', 'akses', 'masuk', 'sering', 'kesalahan', 'lebih', 'mudah', 'aplikasi', 'yang', 'dulu']
['aplikasinya', 'berat', 'mending', 'buat', 'kya', 'yg', 'versi', 'lama', 'aja', 'sat', 'set', 'ringan']	['aplikasinya', 'berat', 'mending', 'buat', 'kaya', 'yang', 'versi', 'saja', 'ringan']
['sering', 'suruh', 'update', 'tapi', 'sering', 'juga', 'errornya']	['sering', 'suruh', 'pembaharuan', 'tapi', 'sering', 'juga', 'kesalahannya']
['aplikasi', 'lemot', 'kalo', 'lagi', 'pesen', 'kereta']	['aplikasi', 'lambat', 'kalau', 'lagi', 'pesan', 'kereta']
['mau', 'kasih', 'binta', 'deh', 'untuk', 'access', 'by', 'kai', 'nih', 'sangat', 'membantu', 'sekali', 'untuk', 'semua', 'keluhan', 'dan', 'saran', 'customer', 'servis', 'nya', 'juga', 'sangat', 'ramah', 'dan', 'baik', 'dalam', 'menjelaskan', 'cara', 'pembatalan', 'keberangkatan', 'dan', 'ubah', 'jadwal', 'keberangkatan', 'kai', 'bagus', 'banget', 'deh']	['mau', 'kasih', 'bintang', 'untuk', 'access', 'by', 'kai', 'sangat', 'membantu', 'untuk', 'semua', 'keluhan', 'dan', 'saran', 'pelanggan', 'pelayanan', 'nya', 'juga', 'sangat', 'ramah', 'dan', 'baik', 'dalam', 'menjelaskan', 'cara', 'pembatalan', 'keberangkatan', 'dan', 'ubah', 'jadwal', 'keberangkatan', 'kai', 'sangat', 'bagus']

Dalam tabel 4.5 di atas terlihat bawa setelah melewati proses *normalization* yang memanfaatkan kamus *lexicon,,* seluruh kata-kata yang terdapat dalam ulasan telah mengalami perubahan yang cukup signifikan yaitu, seluruh kata yang tidak sesuai dengan KBBI keseluruhannya diubah menjadi kata yang sesuai dengan KBBI, karena setiap kata telah dicocokkan dengan kamus *lexicon*. Dengan demikian, dalam ulasan tersebut sudah tidak terdapat lagi kata-kata yang penulisannya disingkat atau tidak baku karena semua kata telah dikembalikan sesuai dengan KBBI.

4.2.5. *Filtered Text*

Seluruh *dataset* yang telah kembali menjadi KBBI akan dilanjutkan dengan melakukan *stopwords removal* yang menghasilkan *filtered text.* Berikut merupakan hasil *stopwords removal* yang memanfaatkan *library NLTK* dan juga ditambahkan dengan *dataset stopwords removal* khusus sebanyak 357 kata. Hasil dari proses *stopwords removal* adalah *dataset* yang hanya memiliki kata-kata yang memiliki makna, dan juga karena proses ini maka terjadi pengurangan dimensi kata dalam *corpus.* Contoh *stopwords removal* dapat dilihat pada tabel 4.6 di bawah.

Tabel 4. 6 Stopwords Removal

Sebelum S*topwords Removal*	**Setelah S*topwords Removal***
['tiba-tiba', 'sering', 'keluar', 'sendiri', 'bukanya', 'lama', 'akses', 'masuk', 'sering', 'kesalahan', 'lebih', 'mudah', 'aplikasi', 'yang', 'dulu']	['keluar', 'buka', 'akses', 'masuk', 'kesalahan', 'lama', 'mudah']
['aplikasinya', 'berat', 'mending', 'buat', 'kaya', 'yang', 'versi', 'saja', 'ringan']	['berat', 'mending', 'kaya', 'versi', 'ringan']
['sering', 'suruh', 'pembaharuan', 'tapi', 'sering', 'juga', 'kesalahannya']	['suruh', 'pembaharuan', 'kesalahannya']
['aplikasi', 'lambat', 'kalau', 'lagi', 'pesan', 'kereta']	['lambat', 'pesan', 'kereta']
['mau', 'kasih', 'bintang', 'untuk', 'access', 'by', 'kai', 'sangat', 'membantu', 'untuk', 'semua', 'keluhan', 'dan', 'saran', 'pelanggan', 'pelayanan', 'nya', 'juga', 'sangat', 'ramah', 'dan', 'baik', 'dalam', 'menjelaskan', 'cara', 'pembatalan', 'keberangkatan', 'dan', 'ubah', 'jadwal', 'keberangkatan', 'kai', 'sangat', 'bagus']	['kasih', 'bintang', 'membantu', 'keluhan', 'saran', 'pelanggan', 'pelayanan', 'ramah', 'menjelaskan', 'pembatalan', 'keberangkatan', 'ubah', 'jadwal', 'keberangkatan', 'bagus']

Dalam tabel 4.6 di atas terlihat bahwa setelah melewati proses *stopwords removal,* banyak kata-kata yang terdapat di dalam ulasan yang dihilangkan sehingga hanya tersisa kata-kata yang memiliki makna saja yang terdapat di ulasan.

4.2.6. *Root Words*

Root words adalah mengembalikan seluruh kata yang terdapat pada ulasan menjadi kata dasar, *root words* merupakan hasil dari *stemming.* Berikut merupakan hasil dari pemanfaatan *library StennerFactori* untuk menghasilkan *dataset* yang sudah tidak terdapat kata imbuhan. Dengan ulasan yang telah dihasilkan pada *dataset* rood words, model dapat lebih mudah menangkap esensi dan pola dalam teks. Contoh *stemming* dapat dilihat pada tabel 4.7 di bawah.

Tabel 4. 7 *Stemming*

Sebelum *stemming*	**Setelah *stemming***
['keluar', 'buka', 'akses', 'masuk', 'kesalahan', 'lama', 'mudah']	['keluar', 'buka', 'akses', 'masuk', 'salah', 'lama', 'mudah']
['berat', 'mending', 'kaya', 'versi', 'ringan']	['berat', 'mending', 'kaya', 'versi', 'ringan']
['suruh', 'pembaharuan', 'kesalahannya']	['suruh', 'baharu', ‘salah']
['lambat', 'pesan', 'kereta']	['lambat', 'pesan', 'kereta']
['kasih', 'bintang', 'membantu', 'keluhan', 'saran', 'pelanggan', 'pelayanan', 'ramah', 'menjelaskan', 'pembatalan', 'keberangkatan', 'ubah', 'jadwal', 'keberangkatan', 'bagus']	['kasih', 'bintang', 'bantu', 'keluh', 'saran', 'langgan', 'layan', 'ramah', 'jelas', 'batal', 'berangkat', 'ubah', 'jadwal', 'berangkat', 'bagus']

Dalam tabel 4.7 di atas terlihat bahwa setelah melewati proses *stemming,* kata-kata yang terdapat di ulasan kembali menjadi kata dasarnya dengan menghilangkan seluruh imbuhan yang terdapat pada awalan atau akhiran kata berdasarkan kamus Sastrawi.

4.3. Hasil Pemodelan Data

Pemodelan data menghasilkan tiga jenis yang telah diubah bentuknya dan menghasilkan data yang sudah siap digunakan untuk dilakukan analisis sentimen. Pemodelan data menghasilkan data berbentuk *numeric* pada TF-IDF, data yang telah memiliki label pada pelabelan data dan menghasilkan data yang telah terbagi menjadi dua jenis data yaitu data latih dan data uji pada *splitting* data.

4.3.1. Perhitungan TF-IDF

Hasil dari pembobotan kata menggunakan TF-IDF ini adalah data berbentuk *numeric* dari data yang berupa teks, pembobotan ini dilakukan karena pada proses menganalisis klasifikasi, data harus berbentuk *numeric* atau angka yang mana merupakan proses pemberian nilai terhadap setiap term yang ada pada data ulasan. Hasil dari TF-IDF merupakan representasi numerik yang dapat digunakan oleh model analisis sentimen. Dengan perhitungan TF-IDF memungkinkan pengolahan data teks menjadi lebih terstruktur dan siap untuk diolah oleh model analisis sentimen. Tanpa pembobotan TF-IDF, model analisis sentimen tidak akan mampu memahami pentingnya setiap *term* dalam konteks dokumen, sehingga kinerja analis akan terganggu, proses TF-IDF akan memastikan hasil yang lebih akurat dan bermakna. Contoh hasil TF-IDF dapat dilihat pada tabel 4. 8 di bawah.

Tabel 4. 8 Pembobotan TF-IDF

Term	***Weight***
Hilang	0.7574
Error	0.6239
Susah	0.6131
Bayar	0.545

Dalam tabel 4. 8 di atas terlihat terdapat 2 baris yang berisi *term* dan *weight* pada baris *term* berisi kata yang sering muncul dalam ulasan, sedangkan *weight* merupakan bobot atau nilai TF-IDF dari kata tersebut dalam *dataset.* Nilai tersebut

dihasilkan dari perkalian TF dan IDF, yaitu perkataan yang sering muncul dalam ulasan dikalikan dengan kata yang jarang muncul dalam ulasan.

Pada baris *term* menunjukkan bahwa kata “Hilang” relatif penting dan unik dalam *dataset* ulasan. Ini berarti kata “hilang” sering muncul dalam ulasan, begitu juga dengan kata “Error” yang menunjukkan bahwa kata ini juga penting dan memberikan kontribusi signifikan dalam konteks ulasan pengguna. Kata “susah” dan “bayar” menunjukkan bahwa pengguna sering menyebutkan kesulitan dalam ulasan mereka, dan kata ini relatif penting dalam *dataset,* begitu pula dengan bayar yang menunjukkan bahwa pembayaran adalah salah satu aspek penting yang sering disebutkan oleh pengguna dalam ulasan mereka.

4.3.2. Label Sentimen

Setelah diketahui nilai bobot dari setiap kata hasil dari TF-IDF *dataset* telah dapat dilakukan proses pelabelan. Proses pelabelan dilakukan dengan menggunakan *dataset* yang telah melewati proses *preprocessing, dataset* yang telah melewati proses *preprocessing* akan menghasilkan data yang berisi ulasan yang hanya memiliki makna dan juga kata yang telah kembali ke bentuk dasarnya, karena kamus *lexicon* berisikan kata-kata yang merupakan kata dasar. Hasil dari pelabelan data adalah *dataset* yang di setiap ulasannya sudah diklasifikasikan ke dalam kelas positif ataupun negatif. Kelas positif ataupun negatif didapatkan dari hasil penjumlahan total nilai kata positif pada keseluruhan kalimat ulasan yang dikurangi total nilai kata negatif pada kalimat ulasan. Pada *dataset* tersebut mendapatkan hasil, yaitu 2361 ulasan termasuk dalam label positif sedangkan 1339 ulasan termasuk dalam label negatif. Contoh pelabelan data dengan *Lexicon Based* dapat dilihat pada tabel 4. 9 di bawah.

Tabel 4. 9 Pelabelan Lexicon Based

Content	Sentiment
Keluar buka akses masuk salah lama mudah	Negatif
berat mending kaya versi ringan	Negatif

Content	Sentiment
suruh baharu salah	Negatif
lambat pesan kereta	Negatif
kasih bintang bantu keluh saran langgan layan ramah jelas batal berangkat ubah jadwal berangkat bagus	Positif

Dalam tabel 4. 9 di atas terlihat bahwa ulasan yang terdapat di dalam *dataset* telah terbagi menjadi 2 kategori. Kategori ini ditentukan berdasarkan sentimen yang diungkapkan dalam setiap ulasan dan ditentukan berdasarkan nilai yang ada pada kamus *lexicon* yang digunakan sebagai pembanding terhadap *dataset* yang digunakan. Pada kategori positif menunjukkan kepuasan, penghargaan, atau pujian dari pengguna terhadap aplikasi. Ulasan positif pada hasil pelabelan sentimen tersebut berupa kata-kata yang menunjukkan apresiasi seperti "bagus', "mantap", "cepat", dan frasa-frasa yang menyiratkan pengalaman dengan aplikasi. Sedangkan ulasan dalam kategori negatif menunjukkan ketidakpuasan, keluhan, atau kritik dari pengguna terhadap aplikasi. Ulasan negatif berdasarkan hasil pelabelan tersebut mengandung kata-kata yang menunjukkan frustrasi atau kekecewaan seperti "kecewa", "susah", "aneh", dan frasa yang menyoroti pengalaman negatif atau masalah yang dihadapi saat menggunakan aplikasi.

4.3.3. Hasil Pembagian Data Latih dan Data Uji

Berdasarkan penjelasan pembagian data latih dan data uji dengan memanfaatkan parameter 'staratify' maka telah didapatkan jumlah data latih dan data uji yang berbeda. Hasil pembagian data dapat dilihat pada tabel 4.10 di bawah.

Tabel 4. 10 Pembagian Data Latih dan Data Uji

Persentase Pembagian	Data Latih	Data Uji
80% : 20%	2844	711
70% : 30%	2489	1066
60% : 40%	2133	1422

Dalam tabel 4. 10 di atas menunjukkan berbagai presentasi pembagian data latih dan data uji yang digunakan dalam proses pelatihan model analisis sentimen untuk aplikasi Access by KAI.

4.4. Hasil Akurasi Klasifikasi *Support Vector Machine*

Setelah *dataset* terbagi menjadi data latih dan data uji pengklasifikasian dengan menggunakan metode *Suppor Vector Machine* dapat dilakukan. Klasifikasi ini menghasilkan performa dari kinerja klasifikasi yang dilakukan terhadap tiga rasio perbandingan data latih dan data uji. Pengklasifikasian dilakukan terhadap tiga perbandingan data latih dan data uji yang berbeda. Hasil akurasi nilai *SVM* dapat dilihat pada tabel 4. 11 di bawah.

Tabel 4. 11 Hasil Akurasi *SVM*

Perbandingan Data Latih dan Data Uji (%)	**Hasil Akurasi**
60 : 40	81,4%
70 : 30	84,2%
80 : 20	85,5%

Berdasarkan tabel 4.11 di atas dapat dilihat bahwa pada rasio perbandingan data 60 : 40 mendapatkan akurasi sebesar 81,4%, rasio perbandingan data 70 : 30 mendapatkan akurasi sebesar 84,2%, dan rasio perbandingan 80 : 20 mendapatkan akurasi sebesar 85,5%. Perhitungan nilai akurasi didapatkan dengan menggunakan rumus (3.1).

Berdasarkan nilai akurasi tersebut metode klasifikasi *Support Vector Machine* memiliki akurasi paling tinggi dalam menglasifikasikan data ulasan aplikasi Access by KAI berdasarkan perbandingan data latih dan data uji 80 : 20 dengan hasil tertinggi yaitu sebesar 85,5%. Perbandingan 80 : 20 dapat menghasilkan akurasi yang tertinggi karena dengan 80% data digunakan untuk pelatihan, model memiliki lebih banyak contoh untuk belajar. Ini memungkinkan model untuk mengenali pola lebih efektif dan generalisasi lebih baik terhadap data baru. Dengan lebih banyak data pelatihan, model *SVM* dapat membangun *hiperlane*

yang lebih representatif dari distribusi data yang sebenarnya. Sesuai dengan keunggulan dari *SVM* yaitu cenderung bekerja lebih baik dengan lebih banyak data pelatihan karena mampu menemukan margin optimal yang memaksimalkan jarak antara kelas, mengurangi kemungkinan *overfitting*, dan model yang dilatih dengan lebih banyak data cenderung memiliki kemampuan generalisasi yang lebih baik.

4.5. Hasil Akurasi Klasifikasi *Naive Bayes*

Klasifikasi ini menghasilkan performa dan kinerja klasifikasi yang dilakukan terhadap *dataset* ulasan aplikasi Access by KAI. Dalam setiap percobaan, ukuran data latih dan data uji bervariasi untuk mengevaluasi bagaimana perubahan dalam jumlah data dapat mempengaruhi hasil, sehingga dapat diketahui kinerja dan performa dari metode dalam melakukan analisis sentimen terhadap tiga jenis rasio perbandingan data yang berbeda, dengan . Hasil akurasi nilai *Naive Bayes* dapat dilihat pada tabel 4. 12 di bawah.

Tabel 4. 12 Hasil Akurasi *Naive Bayes*

Perbandingan Data Latih dan Data Uji (%)	Hasil Akurasi
60 : 40	74,8%
70 : 30	77%
80 : 20	78,1%

Berdasarkan tabel 4. 12 di atas dapat dilihat bahwa pada rasio perbandingan data latih dan data uji 60 : 40 mendapatkan akurasi sebesar 74,8%, pada rasio perbandingan data latih dan data uji 70 : 30 mendapatkan akurasi sebesar 77%, dan pada rasio perbandingan data latih dan data uji 80 : 20 mendapat akurasi sebesar 78,1%. Nilai akurasi didapatkan dengan menggunakan rumus (3.2)

Berdasarkan hasil tersebut metode klasifikasi *Naive Bayes* memiliki akurasi paling tinggi dalam mengklasifikasikan ulasan aplikasi Access by KAI berdasarkan perbandingan data latih dan data uji 80 : 20 atau sama dengan metode *SVM,* tetapi pada metode *Naive Bayes* mendapatkan nilai akurasi yang lebih rendah. Secara keseluruhan, hasil pengklasifikasian menunjukkan bahwa dengan penambahan

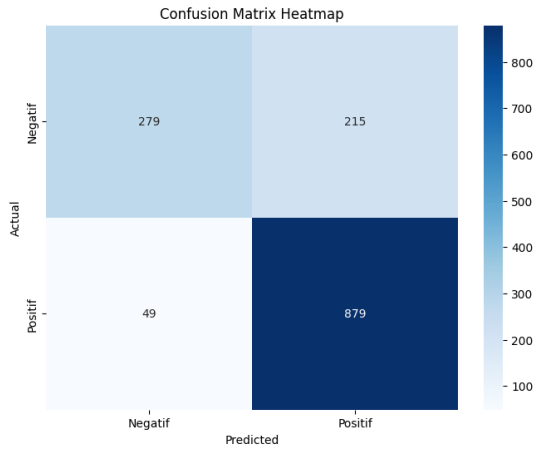
jumlah data latih, akurasi metode *Naive Bayes* meningkat secara signifikan. Hal ini dapat dijelaskan dengan sifat model *Naive Bayes* yang mampu memanfaatkan lebih banyak data untuk menghitung probabilitas lebih akurat, sehingga meningkatkan kemampuan model dalam mengklasifikasikan sampel dengan lebih baik. Namun, tantangan utama yang dihadapi adalah ketidakseimbangan dalam deteksi kelas 'Negatif' dan 'Positif'.

4.6. *Confusion Matrix SVM*

Hasil dari nilai akurasi yang didapatkan pada proses analisis sentimen dengan menggunakan *SVM* untuk mengukur keakuratan dan kinerja model dalam melakukan tugasnya sehingga akan menghasilkan *cunfusion matrix* atau nilai evaluasi. *Confusion matrix* akan dilakukan kepada tiga jenis perbandingan data yang berbeda untuk mengetahui bagaimana kinerja model terhadap tiga bentuk data yang berbeda. Dengan menggunakan *confusin matrix* pada ketiga jenis perbandingan data ini, sehingga dapat dilakukan analisis terhadap metrik penting yaitu, *precision, recall,* dan *F1-Score.*

4.6.1. Hasil *Confusion Matrix SVM* 60 : 40

Pada percobaan pertama pengujian metode *SVM* dilakukan dengan menggunakan perbandingan data latih dan data hasil 60 : 40. Hasil *Confusion Matrix* dengan perbandingan 60 : 40 dapat dilihat pada gambar 4. 1.



Gambar 4. 1 *Confusion Matrix SVM* 60 : 40

Dapat dilihat pada gambar 4. 1 di atas, *Confusion Matrix* memberikan visualisasi yang jelas mengenai kinerja model klasifikasi. Berdasarkan data tersebut maka dapat dipahami kekuatan dan kelemahan dari metode klasifikasi. Warna yang lebih gelap menunjukkan jumlah yang lebih tinggi, membantu dalam cepat mengidentifikasi area dengan prediksi yang benar atau salah. Hasil *Confusion Matrix* dalam bentuk tabel dapat dilihat pada tabel 4. 13 di bawah.

Tabel 4. 13 Hasil *Confusion Matrix SVM* 60 : 40

Aktual	**Prediksi**	
	Negatif	**Positif**
Negatif	279	215
Positif	49	879
Akurasi	81,4%	

Berdasarkan tabel 4. 13 di atas didapatkan hasil yaitu, berhasil diprediksi 279 ulasan sebagai *True Negatif* (TN). terdapat 215 ulasan yang termasuk dalam kategori *False Positive* (FP). Pada bagian prediksi ulasan negatif terdapat 879 ulasan yang berhasil diprediksi dengan benar sehingga termasuk dalam kategori *True Positive* (TP), terdapat 49 ulasan yang seharusnya positif tetapi diprediksi sebagai negatif sehingga termasuk dalam kategori *False Negative* (FN). Sehingga terdapat beberapa kesalahan yang dilakukan oleh model.

Akurasi yang berhasil didapatkan pada pengujian yang dilakukan dengan menggunakan perbandingan data latih dan data uji 60 : 40 adalah 81,4%. Akurasi 81,4% berarti bahwa model memiliki tingkat keberhasilan yang cukup dalam memprediksi kategori ulasan dengan benar. Dari total 1422 ulasan, 1158 ulasan diprediksi dengan benar. Berdasarkan hasil tersebut terdapat kekurangan model *Support Vector Machine* dalam memprediksi ulasan *positive* karena terdapat 264 ulasan yang seharusnya *negative* tetapi di prediksi sebagai ulasan *positve* atau ulasan yang seharusnya *positive* tetapi diprediksi sebagai ulasan negative. Dengan hanya menggunakan 60% data sebagai data untuk pelatihan maka jumlah data latih tidak cukup untuk menangkap semua variasi dalam *dataset* karena *SVM*

membutuhkan data yang cukup untuk membangun model yang *robust* dan *generalizabel,* dan dengan 40% data digunakan untuk pengujian terdapat kemungkinan data uji mencakup variasi yang lebih besar atau pola yang tidak cukup terwakili dalam data latih. Hal ini menyebabkan model kesulitan dalam memprediksi untuk mendapatkan akurasi yang lebih tinggi. Perhitungan terhadap nilai akurasi tersebut dihitung menggunakan rumus (3.3).

Berdasarkan nilai akurasi tersebut maka dapat dilakukan perhitungan terhadap nilai *precision, recall,* dan *F1-Score.* Hasil dari perhitungan tersebut dapat dilihat tabel 4. 14 di bawah.

Tabel 4. 14 Hasil Kinerja Klasifikasi

Sentimen	**Precision**	**Recall**	**F1-Score**	**Support**
Negatif	85%	56%	68%	494
Positif	80%	95%	87%	928
Rata-rata makro	83%	76%	77%	1422
Bobot rata-rata	82%	81%	80%	1422

Berdasarkan tabel 4. 14 di atas didapat hasil dari evaluasi kinerja model dalam melakukan analisis terhadap ulasan yaitu, *precision* pada kelas negatif mendapatkan hasil 85% yang berarti dari semua contoh yang diprediksi sebagai negatif, 85% di antaranya benar-benar negatif, *recall* pada kelas negatif yang berarti dari semua contoh yang sebenarnya negatif, 56% di antaranya berhasil diidentifikasi sebagai negatif oleh model, *F1-Score* atau *Harmonic Mean* dari *precision* dan *recall* untuk kelas negatif mendapatkan hasil 68% memberikan keseimbangan antara keduanya, sedangkan *support* sebesar 494 yang merupakan jumlah contoh yang sebenarnya negatif dalam *dataset.*

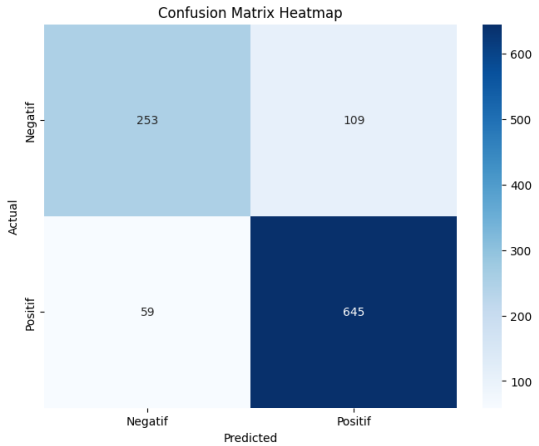
Sedangkan pada kelas positif, *precision* mendapatkan hasil 80% yang berarti dari semua contoh yang diprediksi sebagai positif, 80% di antaranya benar-benar positif, *recall* pada kelas positif yang berarti dari semua contoh yang

sebenarnya positif, 95% di antaranya berhasil diidentifikasi sebagai positif oleh model, *F1-Score* mendapatkan hasil 87%, sedangkan *support* sebesar 928 yang merupakan jumlah contoh yang sebenarnya positif dalam *dataset.* Pada kelas positif *recall* mendapatkan hasil tertinggi. Menunjukkan bahwa kinerja model sangat baik dalam mengidentifikasi pada kelas positif.

Untuk rata-rata makro, mendapatkan nilai *precision* sebesar 83%, *recall* 76%, dan *F1-Score* 77%, merupakan nilai yang memberikan gambaran yang seimbang mengenai kinerja model untuk setiap kelas, tanpa memperhatikan jumlah contoh. Sedangkan bobot rata-rata mendapatkan nilai *precision* sebesar 82%, *recall* 81%, dan *F1-Score* 80%, merupakan nilai yang memberikan gambaran yang lebih realistis mengenai kinerja model pada keseluruhan *dataset,* dengan memperhatikan ketidakseimbangan jumlah contoh di setiap kelas. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus (3.4), sedangkan perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus (3.5), dan perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus (3.6).

4.6.2. Hasil *Confusion Matrix SVM* 70 : 30

Percobaan kedua yang dilakukan untuk melakukan pengujian kinerja terhadap metode *Support Vector Machine* dengan menggunakan perbandingan data latih dan data uji 70 : 30. Hasil *Confusion Matrix* dengan perbandingan 70 : 30% dapat dilihat pada gambar 4. 3.



Gambar 4. 2 *Confusion Matrix SVM* 70 : 30

Pada gambar 4.3, *Confusion Matrix* memberikan visualisasi yang jelas mengenai kinerja model klasifikasi. Melalui data ini, kekuatan dan kelemahan metode klasifikasi dapat dipahami. Warna yang lebih gelap pada matriks menunjukkan jumlah prediksi yang lebih tinggi, memudahkan identifikasi area dengan prediksi yang benar atau salah. Dapat dilihat bahwa pada bagian *true positive* memiliki warna yang paling gelap, karena pada bagian tersebut mendapatkan hasil yang paling besar. Hasil *Confusion Matrix* dalam bentuk tabel dapat dilihat pada tabel 4. 15 di bawah.

Tabel 4. 15 Hasil *Confusion Matrix SVM* 70 : 30

Aktual	**Prediksi**	
	Negatif	**Positif**
Negatif	253	109
Positif	59	645
Akurasi	84,2%	

Berdasarkan tabel 4. 15 di atas didapatkan hasil yaitu, berhasil diprediksi 253 ulasan sebagai *True Negatif* (TN). terdapat 109 ulasan yang seharusnya negatif tetapi diprediksi sebagai positif atau termasuk dalam kategori *False Positive* (FP). Pada bagian prediksi ulasan negatif terdapat 645 ulasan yang berhasil diprediksi dengan benar sehingga termasuk dalam kategori *True Positive* (TP), sedangkan terdapat 59 ulasan yang seharusnya positif tetapi diprediksi sebagai negatif sehingga termasuk dalam kategori *False Negative.* Berdasarkan hasil tersebut maka dapat dilihat bahwa model masih melakukan beberapa kesalahan prediksi.

Akurasi yang berhasil didapatkan pada pengujian yang dilakukan dengan menggunakan perbandingan data latih dan data uji 70 : 30 adalah 84,2%. Akurasi 84,2% berarti bahwa model memiliki tingkat keberhasilan yang cukup tinggi dalam memprediksi kategori ulasan dengan benar. Dari total 1086 ulasan, 896 ulasan diprediksi dengan benar. Dengan menggunakan perbandingan 70 : 30, model dilatih dengan 70% dari keseluruhan *dataset,* memberikan lebih banyak contoh untuk belajar dari berbagai pola dan variasi dalam data, sehingga dapat membantu model

dalam membangun pemahaman yang lebih baik tentang hubungan antara fitur dan label, sehingga meningkatkan kemampuan generalisasi saat memprediksi data uji. Jumlah data latih yang lebih besar memungkinkan model untuk menangkap pola yang lebih kompleks dan mengurangi risiko *overfitting*. Dengan lebih banyak data, model dapat lebih efektif membedakan, yang meningkatkan akurasi prediksi pada data. Salah satu ciri dari model *SVM* yaitu bekerja lebih baik dengan *dataset* yang lebih besar untuk pelatih, dengan lebih banyak data pelatihan, proses pelatihan menjadi lebih stabil dan kurang terpengaruh oleh *outlier* atau variasi acak dalam *dataet* ini memungkinkan model untuk menghasilkan *hyperlane* yang lebih stabil untuk klasifikasi. Perhitungan terhadap nilai akurasi dilakukan dengan menggunakan rumus (3.7).

Berdasarkan hasil tersebut kekurangan model *Support Vector Machine* dibandingkan dengan perbandingan *dataset* 60 : 40 sudah menunjukkan peningkatan akurasi dalam memprediksi ulasan *positive* karena pada perbandingan data latih dan data uji 70 : 30 hanya terdapat 190 kesalahan dalam memprediksi data, menunjukkan metode mengalami peningkatan performa.

Berdasarkan nilai akurasi tersebut maka dapat dilakukan perhitungan terhadap nilai *precision, recall,* dan *F1-Score.* Hasil dari perhitungan tersebut dapat dilihat pada tabel 4. 16 di bawah.

Tabel 4. 16 Hasil Kinerja Klasifikasi

Sentimen	**Precision**	**Recall**	**F1-Score**	**Support**
Negatif	81%	70%	75%	362
Positif	86%	92%	88%	704
Rata-rata makro	83%	81%	82%	1066
Bobot rata-rata	84%	84%	84%	1066

Berdasarkan tabel 4. 16 di atas didapat hasil dari evaluasi kinerja model dalam melakukan analisis terhadap ulasan yaitu, *precision* sebesar 81% untuk kelas

negatif menunjukkan bahwa 81% dari prediksi negatif yang dibuat oleh model adalah benar. *Recall* sebesar 70% menunjukkan bahwa model berhasil mengidentifikasi 70% dari semua contoh negatif yang sebenarnya ada. *F1-Score* sebesar 75% merupakan keseimbangan antara *precision* dan *recall,* menandakan kinerja yang cukup baik dalam mengenali kelas negatif.

Sedangkan pada kelas positif, *precision* sebesar 86% untuk kelas positif menunjukkan bahwa 86% dari prediksi positif yang dibuat oleh model adalah benar. *Recall* sebesar 92% menunjukkan bahwa model berhasil mengidentifikasi 92% dari semua contoh positif yang sebenarnya ada. *F1-Score* sebesar 88% menunjukkan keseimbangan yang sangat baik antara *precision* dan *recall,* menandakan kinerja yang sangat kuat dalam mengenali kelas positif.

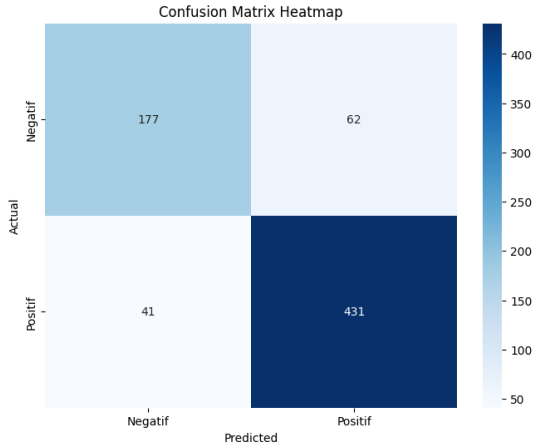
Untuk rata-rata makro memberikan gambaran yang seimbang mengenai kinerja model untuk setiap kelas tanpa memperhatikan jumlah contoh. *Precision* sebesar 83% dan *recall* sebesar 81% menunjukkan bahwa model secara konsisten baik dalam mengenali kedua kelas. *F1-Sxore* sebesar 82% mengindikasikan keseimbangan yang baik antara *precision* dan *recall,* menunjukkan bahwa model tidak hanya akurat tetapi juga efektif alam mengidentifikasi kelas yang benar.

Bobot rata-rata memberikan gambaran yang lebih realistis mengenai kinerja model pada keseluruhan *dataset* dengan mempertimbangkan ketidakseimbangan jumlah contoh setiap kelas. Nilai *precision* dan *recall* sebesar 84% menunjukkan bahwa model memiliki kinerja yang kuat dalam membuat prediksi yang benar secara keseluruhan. *F1-Score* sebesar 84% menunjukkan bahwa model mampu mempertahan keseimbangan yang baik antara *precision* dan *recall* di seluruh *dataset.* Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus (3.8), sedangkan perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus (3.9), dan perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus (3.10).

Berdasarkan hasil tersebut secara keseluruhan hasil evaluasi menunjukkan bahwa model memiliki kinerja yang cukup baik dalam melakukan tugasnya. Model menunjukkan kemampuan yang kuat dalam mengidentifikasi kelas positif dan negatif, dengan *precision* dan *recall* yang tinggi.

4.6.3. Hasil *Confusion Matrix SVM* 80 : 20

Percobaan ketiga yang dilakukan untuk melakukan pengujian kinerja terhadap metode *Support Vector Machine* dengan menggunakan perbandingan data latih dan data uji 80 : 20. Hasil *Confusion Matrix* dengan perbandingan 80 : 20 dapat dilihat pada gambar 4. 5.



Gambar 4. 3 *Confusion Matrix SVM* 80 : 20

Pada gambar 4.5, *Confusion Matrix* memberikan visualisasi yang jelas mengenai kinerja model klasifikasi. Tingginya nilai pada TP dan TN menunjukkan bahwa model mampu melakukan prediksi yang benar dengan baik, adanya nilai FP dan FN menunjukkan bahwa masih ada kesalahan prediksi yang dilakukan oleh model. Hasil *Confusion Matrix* dalam bentuk tabel dapat dilihat pada tabel 4. 17 di bawah.

Tabel 4. 17 Hasil *Confusion Matrix SVM* 80 : 20

Aktual	**Prediksi**	
	Negatif	**Positif**
Negatif	177	62
Positif	41	431
Akurasi	85,5%	

Berdasarkan tabel 4. 17 di atas didapatkan hasil yaitu, berhasil diprediksi 177 ulasan sebagai *True Negatif* (TN) sedangkan terdapat 62 ulasan yang seharusnya negatif tetapi diprediksi sebagai positif atau termasuk dalam kategori

False Positive (FP). Pada bagian prediksi ulasan negatif terdapat 431 ulasan yang berhasil diprediksi dengan benar sehingga termasuk dalam kategori *True Positive* (TP), sedangkan terdapat 41 ulasan yang seharusnya positif tetapi diprediksi sebagai negatif sehingga termasuk dalam kategori *False Negative* (FN). Berdasarkan *confusion matrix* tersebut dapat terlihat bahwa model sudah dapat memprediksi dengan akurasi yang tinggi, kesalahan prediksi yang dilakukan sudah sangat berkurang sekali dibandingkan dengan 2 perbandingan data sebelumnya.

Akurasi yang berhasil didapatkan pada pengujian yang dilakukan dengan menggunakan perbandingan data latih dan data uji 80 : 20 adalah 85,5%. Akurasi 85,5% berarti bahwa model memiliki tingkat keberhasilan yang tinggi dalam memprediksi kategori ulasan dengan benar. Dari total 711 ulasan, 608 ulasan diprediksi dengan benar. Dengan perbandingan 80 : 20, 80% dari total data digunakan untuk melatih model. Ini berarti model memiliki lebih banyak data untuk mempelajari pola dan karakteristik dari setiap kelas, lebih banyak data latih memungkinkan model untuk mengidentifikasi dan memahami pola yang lebih kompleks dan variatif dalam data, yang mengarah pada generalisasi yang lebih baik ketika diaplikasikan pada data uji. Dibandingkan pada perbandingan data latih dan uji 60 : 40, ada risiko yang lebih tinggi bahwa model mungkin *overfit* pada data latih karena proporsi data latih yang lebih kecil tidak cukup representatif dari seluruh distribusi data, dengan porsi data latih yang lebih besar risiko untuk terjadinya *overfitting* akan lebih rendah, dan model dapat memprediksi dengan lebih baik pada data uji. Dengan menggunakan 80% data untuk pelatihan, ada kemungkinan lebih besar bahwa model melihat hampir semua variasi yang ada dalam data. Ini membantu model untuk belajar lebih baik dan memprediksi dengan lebih akurat pada data uji yang belum pernah dilihat sebelumnya, pada perbandingan 60 : 40, dengan hanya 60% data untuk pelatihan, variasi data yang tersedia untuk model belajar tidak cukup untuk mencakup semua skenario yang ada dalam data uji. Pada perbandingan 70 : 30, meskipun lebih baik dari 60 : 40, masih kurang dibandingkan dengan 80 : 20. Dengan distribusi data yang lebih representatif model melihat hampir semua variasi yang ada dalam data, yang meningkatkan kemampuan prediksi pada data uji. Berdasarkan hasil tersebut

metode *Support Vector Machine* telah berhasil meraih nilai akurasi yang tinggi pada perbandingan data latih dan data uji 80 : 20, dengan hanya terdapat 103 kesalahan prediksi. Perhitungan terhadap nilai akurasi tersebut dihitung dengan menggunakan rumus (3.11)

Berdasarkan nilai akurasi tersebut maka dapat dilakukan perhitungan terhadap nilai *precision, recall,* dan *F1-Score.* Hasil dari perhitungan tersebut dapat dilihat pada tabel 4. 18 di bawah.

Tabel 4. 18 Hasil Kinerja Klasifikasi

Sentimen	**Precision**	**Recall**	**F1-Score**	**Support**
Negatif	81%	74%	77%	239
Positif	87%	91%	89%	472
Rata-rata makro	84%	83%	83%	711
Bobot rata-rata	85%	86%	85%	711

Berdasarkan tabel 4. 18 di atas didapat hasil dari evaluasi kinerja model dalam melakukan analisis terhadap ulasan yaitu, *precision* sebesar 81% untuk kelas negatif menunjukkan bahwa 81% dari prediksi negatif yang dibuat oleh model adalah benar. *Recall* sebesar 74% menunjukkan bahwa model berhasil mengidentifikasi 74% dari semua contoh negatif yang sebenarnya ada. *F1-Score* sebesar 77% merupakan keseimbangan antara *precision* dan *recall,* menandakan kinerja yang sangat baik dalam mengenali kelas negatif

Precision sebesar 87% untuk kelas positif menunjukkan bahwa 87% dari prediksi positif yang dibuat oleh model adalah benar. *Recall* sebesar 91% menunjukkan bahwa model berhasil mengidentifikasi 91% dari semua contih positif yang sebenarnya ada. *F1-Score* sebesar 89% menunjukkan keseimbangan yang sangat baik antara *precision* dan *recall,* menandakan kinerja yang sangat kuat dalam mengenali kelas positif.

Rata – rata makro mendapatkan nilai *Precision* sebesar 84% dan *recall* sebesar 83% menunjukkan bahwa model secara konsisten baik dalam mengenali kedua kelas. *F1-Score* sebesar 83% mengindikasikan keseimbangan yang sangat baik antara *precision* dan *recall,* menunjukkan bahwa model tidak hanya akurat tetapi juga efektif dalam mengidentifikasi kelas yang benar.

Bobot rata-rata yang menghasilkan nilai *precision* sebesar 85% dan *recall* sebesar 86% menunjukkan bahwa model memiliki kinerja yang sangat kuat dalam membuat prediksi yang benar secara keseluruhan. *F1-Score* sebesar 85% menunjukkan bahwa model mampu mempertahankan keseimbangan yang sangat baik antara *precision* dan *recall* di seluruh *dataset.* Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus (3.12), sedangkan perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus (3.13), dan perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus (3.14).

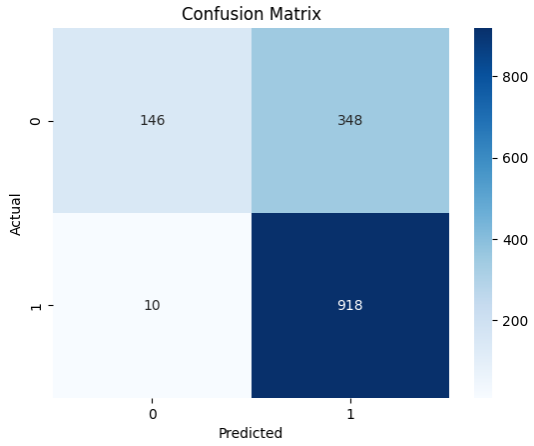
Secara keseluruhan, model menunjukkan kemampuan yang sangat kuat dalam mengidentifikasi kelas positif dan negatif dengan *precision* dan *recall* yang tinggi. Rata-rata makro dan bobot rata-rata yang tinggi menunjukkan bahwa model memiliki keseimbangan yang sangat baik dalam performanya di seluruh *dataset.* Pengujian yang dilakukan pada perbandingan data latih dan data uji 80 : 20% memberikan prediksi yang akurat dan menunjukkan bahwa model *SVM* sangat baik dalam bekerja dengan data latih yang besar karena metode akan lebih memiliki banyak data untuk belajar dan dengan data uji dengan jumlah yang lebih kecil membuat metode *SVM* akan lebih mudah mengenali data yang pernah di lihat pada data latih sebelumnya.

4.7. *Confusion Matrix Naive Bayes*

Hasil dari nilai akurasi yang didapatkan pada proses analisis sentimen dengan menggunakan *Naive Bayes* digunakan untuk mengukur keakuratan dan kinerja model dalam melakukan tugasnya. Evaluasi kinerja model ini menghasilkan *confusion matrix,* yang memberikan nilai-nilai evaluasi penting. *Confusion matrix* akan digunakan pada tiga jenis perbandingan data yang berbeda untuk mengetahui bagaimana kinerja model terhadap tiga bentuk data yang berbeda.

4.7.1. Hasil *Confusion Matrix Naive Bayes 60 : 40*

Percobaan pertama yang dilakukan untuk melakukan pengujian kinerja terhadap metode *Naive Bayes* dengan menggunakan perbandingan 60 : 40. Hasil *Confusion Matrix* dengan perbandingan 60 : 40 dapat dilihat pada gambar 4. 7.



Gambar 4. 4 *Confusion Matrix Naive Bayes* 60 : 40

Pada gambar 4.7, Tingginya nilai pada TP dan TN menunjukkan bahwa model mampu melakukan prediksi yang benar dengan baik, adanya nilai FP dan FN menunjukkan bahwa masih ada kesalahan prediksi yang dilakukan oleh model. Hasil *Confusion Matrix* dalam bentuk tabel dapat dilihat pada tabel 4. 19 di bawah.

Tabel 4. 19 Hasil *Confusion Matrix Naive Bayes* 60 : 40

Aktual	**Prediksi**	
	Negatif	**Positif**
Negatif	146	348
Positif	10	918
Akurasi	74,8%	

Berdasarkan tabel 4. 19 di atas, model berhasil mencapai nilai akurasi sebesar 74,8%. Model ini mampu mengidentifikasi 146 ulasan yang sebenarnya negatif dan berhasil diprediksi sebagai negatif atau *True Negative* (TN). Namun, terdapat 348 ulasan yang sebenarnya negatif tetapi salah diprediksi sebagai positif atau *False Positive* (FP). Di sisi lain, model berhasil mengidentifikasi dengan benar

918 ulasan yang sebenarnya positif dan diprediksi sebagai positif atau *True Positive* (TP). Meskipun demikian, ada 10 ulasan yang seharusnya positif tetapi diprediksi sebagai negatif oleh model atau *False Negative* (FN). Dari hasil ini, terlihat bahwa model mengalami kesulitan dalam mengidentifikasi ulasan positif. Perhitungan terhadap nilai akurasi menggunakan rumus (3.15)

Model mengalami kesulitan dalam mengidentifikasi ulasan positif, yang mengakibatkan kesalahan dalam memprediksi kelas negatif sebagai positif. Hal ini disebabkan oleh sifat *Naive Bayes* yang mengasumsikan bahwa semua fitur (kata-kata) bersifat independen, padahal dalam data teks sering kali terdapat korelasi tinggi antara kata-kata. Asumsi *independensi* ini tidak terpenuhi, sehingga mengurangi performa model dalam mengenali pola kompleks. Selain itu, *Naive Bayes* tidak memodelkan interaksi atau relasi antara fitur-fitur dalam teks, yang menyebabkan model kehilangan informasi penting mengenai bagaimana kata-kata saling mempengaruhi dalam konteks yang lebih luas. Akurasi sebesar 74,8% berarti bahwa model memiliki tingkat keberhasilan yang tinggi dalam memprediksi kategori ulasan dengan benar. Dari total 1422 ulasan, 1064 ulasan diprediksi dengan benar, sehingga terdapat 358 kesalahan prediksi terhadap ulasan. Ini berarti sekitar 25% dari total ulasan yang diuji mengalami kesalahan

Berdasarkan nilai akurasi tersebut maka dapat dilakukan perhitungan terhadap nilai *precision, recall,* dan *F1-Score.* Hasil dari perhitungan tersebut dapat dilihat pada tabel 4. 20 di bawah.

Tabel 4. 20 Hasil Kinerja Klasifikasi *Naive Bayes*

Sentimen	**Precision**	**Recall**	**F1-Score**	**Support**
Negatif	94%	30%	45%	494
Positif	73%	99%	84%	928
Rata-rata makro	83%	64%	64%	1422
Bobot rata-rata	80%	75%	70%	1422

Berdasarkan tabel 4. 20 di atas didapat hasil dari evaluasi kinerja model dalam melakukan analisis terhadap ulasan yaitu, *precision* sebesar 94% untuk kelas negatif menunjukkan bahwa 94% dari prediksi negatif yang dibuat oleh model adalah benar. *Recall* sebesar 30% menunjukkan bahwa model berhasil mengidentifikasi 30% dari semua contoh negatif yang sebenarnya ada. *F1-Score* sebesar 45% merupakan keseimbangan antara *precision* dan *recall,* menandakan kinerja yang kurang baik dalam mengenali kelas negatif.

Precision sebesar 73% untuk kelas positif menunjukkan bahwa 37% dari prediksi positif yang dibuat oleh model adalah benar. *Recall* sebesar 99% menunjukkan bahwa model berhasil mengidentifikasi 99% dari semua contih positif yang sebenarnya ada. *F1-Score* sebesar 84% menunjukkan keseimbangan yang sangat baik antara *precision* dan *recall,* menandakan kinerja yang sangat kuat dalam mengenali kelas positif.

Pada rata-rata makro mendapatkan nilai *precision* sebesar 83% dan *recall* sebesar 64% menunjukkan bahwa model secara konsisten cukup baik dalam mengenali kedua kelas. *F1-Score* sebesar 64% mengindikasikan keseimbangan yang baik antara *precision* dan *recall,* menunjukkan bahwa model sudah cukup akurat dan juga efektif dalam mengidentifikasi kelas yang benar.

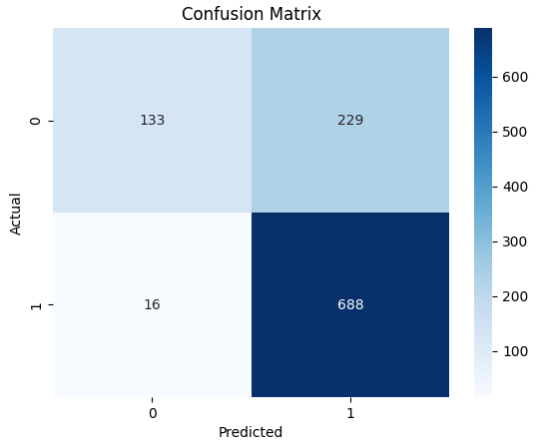
Pada bobot rata-rata mendapatkan nilai *precision* sebesar 80% dan *recall* sebesar 75% menunjukkan bahwa model memiliki kinerja yang kuat dalam membuat prediksi yang benar secara keseluruhan. *F1-Score* sebesar 70% menunjukkan bahwa model mampu mempertahankan keseimbangan yang baik antara *precision* dan *recall* di seluruh *dataset.* Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus (3.16), sedangkan perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus (3.17), dan perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus (3.18).

Secara keseluruhan, hasil evaluasi menunjukkan bahwa model *Naive Bayes* memiliki kinerja yang bagus. Beberapa temuan yang didapatkan berdasarkan hasil *confusion matrix* tersebut yaitu, *Precision* yang rendah pada sentimen negatif menunjukkan bahwa dari semua ulasan yang diprediksi sebagai negatif oleh model, hanya sebagian kecil yang benar-benar negatif. Hal ini dapat terjadi karena *Naive*

Bayes cenderung menjadi kurang tepat dalam memprediksi kelas minoritas (dalam hal ini, sentimen negatif) jika distribusi kelas tidak seimbang dalam data latih. *Naive Bayes* dapat lebih cenderung memprediksi kelas mayoritas (dalam hal ini, sentimen positif) karena frekuensi kemunculannya yang lebih tinggi dalam data. *Recall* yang rendah pada sentimen negatif menunjukkan bahwa model gagal mengidentifikasi sebagian besar ulasan yang sebenarnya negatif. F1-score yang rendah pada sentimen negatif mencerminkan ketidakseimbangan antara *precision* dan *recall*. Ini menunjukkan bahwa model memiliki kesulitan dalam mencapai keseimbangan yang baik antara memprediksi dengan tepat ulasan yang benar-benar negatif (*precision*) dan mengidentifikasi sebagian besar ulasan yang sebenarnya negatif (*recall*). Bobot rata-rata yang rendah menunjukkan bahwa model tidak secara konsisten baik dalam mengklasifikasikan kelas positif maupun negatif. Performa model bervariasi antara kelas-kelas yang, ada dan model lebih baik dalam memprediksi satu kelas dibandingkan yang lain. Bobot rata-rata yang rendah pada ulasan negatif menandakan bahwa model lebih efektif pada ulasan positif.

4.7.2. Hasil *Confusion Matrix Naive Bayes* 70 : 30

Percobaan kedua yang dilakukan untuk melakukan pengujian kinerja terhadap metode *Naive Bayes* dengan menggunakan perbandingan 70 : 30. Percobaan tersebut menghasilkan *confusion matrix* berdasarkan nilai akurasi. Hasil *Confusion Matrix* dengan perbandingan 70 : 30 dapat dilihat pada gambar 4. 9.



Gambar 4. 5 *Confusion Matrix Naive Bayes* 70 : 30

Gambar 4.9 menampilkan *Confusion Matrix* yang menunjukkan kinerja model klasifikasi. Matriks ini terbagi menjadi empat bagian: *True Positive* (TP) di kiri bawah, *True Negative* (TN) di kiri atas, *False Positive* (FP) di kanan atas, dan *False Negative* (FN) di kanan bawah. Hasil *Confusion Matrix* dapat digunakan untuk melakukan penyeimbangan data atau penyesuaian parameter. Hasil *Confusion Matrix* dalam bentuk tabel dapat dilihat pada tabel 4. 21 di bawah.

Tabel 4. 21 Hasil *Confusion Matrix Naive Bayes* 70 : 30

Aktual	**Prediksi**	
	Negatif	**Positif**
Negatif	133	229
Positif	16	688
Akurasi	77%	

Berdasarkan tabel 4. 21 di atas didapatkan hasil yaitu, model berhasil mendapatkan nilai akurasi sebesar 77%. Pada tabel tersebut berhasil diprediksi 133 ulasan sebagai *True Negatif* (TN) atau ulasan yang benar-benar negatif dan diprediksi negatif. Terdapat 229 ulasan yang seharusnya negatif tetapi diprediksi sebagai positif atau *False Positive* (FP). Pada prediksi ulasan negatif terdapat 688 ulasan yang berhasil diprediksi dengan atau *True Positive* (TP), sedangkan terdapat 16 ulasan yang seharusnya positif tetapi diprediksi sebagai negatif atau *False Negative* (FN). Nilai akurasi dihasilkan dengan menggunakan rumus (3.19)

Model memiliki permasalahan dalam melakukan identifikasi ulasan positif, sehingga model melakukan kesalahan dalam melakukan prediksi terhadap kelas negatif yang diprediksi menjadi kelas positif, hal tersebut disebabkan karena salah satu sifat *Naive Bayes* yang juga dapat menyebabkan akurasi yang rendah adalah asumsi bahwa distribusi fitur dalam setiap kelas adalah independen terutama dalam konteks kalimat atau frasa tertentu. Misalnya, dalam kalimat "produk ini tidak bagus", kata-kata "tidak" dan "bagus" memiliki hubungan yang erat, di mana kemunculan "tidak" mengubah makna kata "bagus" menjadi negatif. Namun, *Naive Bayes* tidak mampu menangkap dependensi semacam ini antara kata-kata. Karena

Naive Bayes tidak memodelkan hubungan antar kata-kata atau urutan kata-kata, hal ini dapat menyebabkan model kehilangan informasi penting dalam data teks, yang pada gilirannya dapat mengurangi akurasi prediksinya.

Akurasi 77% berarti bahwa model memiliki tingkat keberhasilan yang cukup tinggi dalam memprediksi kategori ulasan dengan benar. Dari total 1066 ulasan, 621 ulasan diprediksi dengan benar. Berdasarkan hasil tersebut, metode *Naive Bayes* telah berhasil meraih nilai akurasi yang cukup baik pada perbandingan data latih dan data uji 70 : 30 dibandingkan pada perbandingan data lati dan data uji 60 : 40, dengan terdapat 445 kesalahan prediksi terhadap ulasan. Model menunjukkan kemampuan yang lebih andal dalam mempelajari pola dari data latih yang lebih besar.

Berdasarkan nilai akurasi tersebut maka dapat dilakukan perhitungan terhadap nilai *precision, recall,* dan *F1-Score.* Hasil dari perhitungan tersebut dapat dilihat pada tabel 4. 22 di bawah.

Tabel 4. 22 Hasil Kinerja Klasifikasi *Naive Bayes*

Sentimen	**Precision**	**Recall**	**F1-Score**	**Support**
Negatif	89%	37%	52%	362
Positif	75%	98%	85%	704
Rata-rata makro	82%	67%	68%	1066
Bobot rata-rata	80%	77%	74%	1066

Berdasarkan tabel 4. 22 di atas didapatkan hasil dari evaluasi kinerja model dalam menganalisis sentimen, dengan memperhitungkan metrik *precision, recall*, dan *F1-Score,* serta dukungan (*support)* untuk setiap kelas sentimen, yaitu negatif dan positif. *Precision* mengukur seberapa banyak dari semua prediksi positif yang sebenarnya benar positif. Untuk kelas negatif, *precision* sebesar 89% menunjukkan bahwa dari semua ulasan yang diprediksi sebagai negatif, 89% di antaranya benar-benar negatif, hal tersebut menunjukkan bahwa metode cukup baik dalam

menghindari *false postive*. Sedangkan untuk kelas positif, *precision* sebesar 75% menandakan bahwa dari semua ulasan yang diprediksi sebagai positif, 75%nya benar positif.

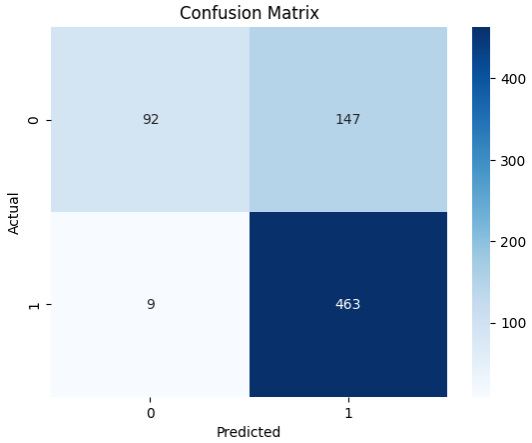
Untuk kelas negatif, *recall* sebesar 37% menunjukkan bahwa dari semua ulasan yang sebenarnya negatif, model hanya berhasil mengidentifikasi 37% di antaranya, hal ini dapat disebabkan karena ulasan negatif menggunakan bahasa yang lebih beragam atau kurang konsisten dibandingkan dengan ulasan positif. Kata-kata yang menunjukkan sentimen negatif bisa lebih banyak varian dan tidak mudah diprediksi. Sementara untuk kelas positif, *recall* sebesar 98% menandakan bahwa model sangat baik dalam mengidentifikasi ulasan yang sebenarnya positif, dengan berhasil mengidentifikasi 98% dari semua ulasan positif yang ada, menunjukkan bahwa metode dapat memahami kata-kata secara independen sehingga metode mamu melakukan prediksi ulasan dengan sangat baik pada kelas positif. F1-Score adalah *harmonic mean* dari *precision* dan *recall*. Untuk kelas negatif, F1-Score sebesar 52% mencerminkan keseimbangan antara *precision* dan *recall,* sedangkan untuk kelas positif, F1-Score sebesar 85% menunjukkan kinerja yang baik dengan mendapatkan hasil yang tinggi dalam menggabungkan *precision* dan *recall.*

Dari rata-rata makro, dapat dilihat bahwa *precision* rata-rata untuk kedua kelas adalah 82%, *recall* rata-rata adalah 67%, dan F1-Score rata-rata adalah 68%. Ini menunjukkan kinerja keseluruhan model dalam mengklasifikasikan ulasan sentimen, dengan mempertimbangkan kedua kelas secara seimbang.

Dengan bobot rata-rata, *precision* mencapai 80%, *recall* mencapai 77%, dan F1-Score mencapai 74%. Hal ini menggambarkan kinerja metode mendapatkan hasil yang sangat baik pada klasifikasi ulasan positif tetapi mengalami penurunan pada pengklasifikasian ulasan negatif yang menyebabkan nilai akurasi mengalami penurunan dan mendapatkan bobot rata-rata yang rendah karena terdapat kekurangan dalam prediksi kelas negatif. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus (3.20)*,* sedangkan perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus (3.21)*,* dan perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus (3.22).

4.7.3. Hasil *Confusion Matrix Naive Bayes* 80 : 20

Percobaan ketiga yang dilakukan untuk melakukan pengujian kinerja terhadap metode *Naive Bayes* dengan menggunakan perbandingan data latih dan data uji 80 : 20. Hasil *Confusion Matrix* dengan perbandingan 80 : 20 dapat dilihat pada gambar 4. 11.



Gambar 4. 6 *Confusion Matrix Naive Bayes* 80 : 20

Pada gambar 4.11 menunjukkan *Confusion Matrix* yang terdiri dari empat bagian persegi yang masing-masing menampilkan angka-angka yang merepresentasikan prediksi yang benar dan yang salah oleh model. Di bagian kiri bawah, terdapat *True Positive* (TP), di bagian kiri atas terdapat *True Negative* (TN), di bagian kanan atas terdapat *False Positive* (FP), dan di bagian kanan bawah terdapat *False Negative* (FN). *Confusion Matrix* membantu untuk dengan jelas melihat di mana model berhasil dan di mana ada kelemahan, terlihat bahwa terdapat banyak kesalahan pada bagian kanan bawah yaitu *false Negatice*. Hasil *Confusion Matrix* dalam bentuk tabel dapat dilihat pada tabel 4. 23 di bawah.

Tabel 4. 23 Hasil *Confusion Matrix Naive Bayes* 70 : 30

Aktual	**Prediksi**	
	Negatif	**Positif**
Negatif	92	147
Positif	9	465
Akurasi	78,1%	

Berdasarkan tabel 4. 23 di atas didapatkan hasil yaitu, model berhasil mendapatkan nilai akurasi sebesar 78.1%. pada tabel tersebut berhasil diprediksi 92 ulasan sebagai *True Negatif* (TN). Terdapat juga *False Positive* (FP) terdapat 147 ulasan yang seharusnya negatif tetapi diprediksi sebagai positif. Pada bagian prediksi ulasan negatif terdapat kategori *True Positive* (TP) terdapat 465 ulasan yang berhasil diprediksi dengan benar sehingga termasuk dalam kategori *True Positive* (TP), terdapat 9 ulasan yang seharusnya positif tetapi diprediksi sebagai negatif sehingga termasuk dalam kategori *False Negative* (FN). Nilai akurasi didapatkan dengan menggunakan rumus (3.23)

Model bisa mendapatkan akurasi yang paling tinggi pada data latih dan data uji 80:20 karena pada pembagian tersebut, model memiliki lebih banyak data untuk melatihnya (80% dari total data). Hal ini memiliki hubungan dengan sifat dari *Naive Bayes* yang mengasumsikan bahwa setiap kata adalah independen satu sama lain. Dengan lebih banyak data latih, model memiliki kesempatan yang lebih besar untuk melihat variasi yang lebih luas dari setiap fitur atau kata-kata dalam teks, bahkan dengan asumsi *independensi* ini.

Lebih banyak data latih memberikan model kesempatan untuk melihat lebih banyak contoh di mana asumsi *Naive Bayes* tentang independensi antara fitur tidak sepenuhnya benar. Dengan kata lain, meskipun model menggunakan asumsi *independensi*, dengan melihat lebih banyak data, model dapat melihat bahwa dalam konteks tertentu, kata-kata dapat memiliki korelasi atau dependensi yang signifikan.

Selain itu, dengan lebih banyak data latih, model juga dapat memperoleh pemahaman yang lebih baik tentang variasi dalam urutan kata-kata dalam teks. Meskipun *Naive Bayes* mengabaikan urutan kata-kata, dengan melihat lebih banyak contoh teks, model dapat secara tidak langsung menangkap beberapa pola dalam urutan kata-kata yang dapat membantu dalam analisis sentimen.

Sehingga, pada pembagian data latih dan data uji 80:20, model memiliki lebih banyak kesempatan untuk memperoleh pemahaman yang lebih baik tentang kompleksitas teks, termasuk korelasi antara fitur dan urutan kata-kata, yang pada akhirnya dapat meningkatkan akurasinya dalam memprediksi sentimen ulasan, sehingga meningkatkan akurasi dari metode tersebut.

Akurasi 78.1% berarti bahwa model memiliki tingkat keberhasilan yang tinggi dalam memprediksi kategori ulasan dengan benar. Dari total 713 ulasan, 557 ulasan diprediksi dengan benar. Berdasarkan hasil tersebut metode *Naive Bayes* telah berhasil meraih nilai akurasi yang cukup baik pada perbandingan data latih dan data uji 80 : 20 dibandingkan pada perbandingan data lati dan data uji 60 : 40, dengan terdapat 445 kesalahan prediksi terhadap ulasan, dan pada perbandingan 70 : 30, dengan terdapat 156 kesalahan.

Berdasarkan nilai akurasi tersebut maka dapat dilakukan perhitungan terhadap nilai *precision, recall,* dan *F1-Score.* Hasil dari perhitungan tersebut dapat dilihat pada tabel 4. 24 di bawah.

Tabel 4. 24 Hasil Kinerja Klasifikasi *Naive Bayes*

Sentimen	**Precision**	**Recall**	**F1-Score**	**Support**
Negatif	91%	38%	54%	239
Positif	76%	98%	86%	472
Rata-rata makro	83%	68%	70%	711
Bobot rata-rata	81%	78%	75%	711

Berdasarkan tabel 4. 24 di atas didapatkan hasil dari evaluasi kinerja model dalam menganalisis sentimen. Pada kelas negatif, *precision* sebesar 91% menunjukkan bahwa dari semua ulasan yang diprediksi sebagai negatif, 91% di antaranya benar-benar negatif. Sedangkan untuk kelas positif, *precision* sebesar 76% menandakan bahwa dari semua ulasan yang diprediksi sebagai positif, 76% di antaranya adalah benar-benar positif. Pada kelas postif terlihat bawa model berhasil mendapatkan kinerja yang cukup bagus.

Kelas negatif, *recall* sebesar 38% menunjukkan bahwa dari semua ulasan yang sebenarnya negatif, model hanya berhasil mengidentifikasi 38% di antaranya. Sementara untuk kelas positif, *recall* sebesar 98% menandakan bahwa model sangat baik dalam mengidentifikasi ulasan yang sebenarnya positif, dengan berhasil

mengidentifikasi 98% dari semua ulasan positif yang ada. Menunjukkan bahwa model dapat melakukan kinerja yang baik pada kelas positif.

F1-Score adalah *harmonic mean* dari *precision* dan *recall*, memberikan gambaran keseluruhan tentang kinerja model. Untuk kelas negatif, F1-Score sebesar 55% mencerminkan keseimbangan antara *precision* dan *recall,* sedangkan untuk kelas positif, F1-Score sebesar 86% menunjukkan kinerja yang baik dalam menggabungkan *precision* dan *recall. Support* mengindikasikan jumlah kemunculan setiap kelas dalam *dataset*. Ada 279 ulasan yang termasuk dalam kelas negatif dan 472 ulasan yang termasuk dalam kelas positif.

Dari rata-rata makro, dapat dilihat bahwa *precision* rata-rata untuk kedua kelas adalah 83%, *recall* rata-rata adalah 68%, dan F1-Score rata-rata adalah 70%. Ini menunjukkan kinerja keseluruhan model dalam mengklasifikasikan ulasan sentimen, dengan mempertimbangkan kedua kelas secara seimbang. Dapat terlihat bahwa *precision* mendapatkan hasil yang lebih tinggi dibandingkan dengan dua kinerja metode lainnya.

Sementara itu, bobot rata-rata memberikan perhatian lebih besar pada kelas yang lebih banyak muncul dalam *dataset*. Dengan bobot rata-rata, *precision* mencapai 81%, *recall* mencapai 78%, dan F1-Score mencapai 75%. Hal ini menggambarkan kinerja model dengan mempertimbangkan distribusi yang tidak seimbang dari kedua kelas. Dengan *precision* yang mencapai 81% ini adalah indikator yang baik bahwa model memiliki tingkat ketepatan yang tinggi dalam prediksi kelas mayoritas. Sedangkan *recall* yang mencapai 78% mengindikasikan bahwa dari semua ulasan yang sebenarnya positif, model mampu mengidentifikasikan 78% di antaranya dengan benar. Ini menunjukkan bawha model cukup baik dalam menangkap sebagian besar *instance* dari kelas positif, meskipun ada beberapa *instance* yang tidak terdeteksi dengan benar. *F1-Score* yang mencapai 75% menunjukkan bahwa model tidak hanya akurat dalam prediksi kelas mayoritas. Perhitungan terhadap nilai *precision* dilakukan dengan menggunakan rumus (3.24)*,* sedangkan perhitungan terhadap nilai *recall* dilakukan dengan menggunakan rumus (3.25)*,* dan perhitungan terhadap nilai *F1-Score* dilakukan dengan menggunakan rumus (3.26).

4.8. Perbandingan *Support Vector Machine* dan *Naive Bayes*

Perbandingan antara kedua metode tersebut akan menunjukkan metode mana yang lebih unggul dalam nilai akurasi, nilai *precission,* nilai *recall,* dan nilai *F1-Score.* Perbandingan dilakukan terhadap tiga jenis data latih dan data uji dengan perbandingan yang berbeda-beda. Dengan melakukan perbandingan terhadap hasil klasifikasi dari kedua metode pada tiga jenis pembagian data yang berbeda, maka dapat terlihat kelebihan dan kekurangan masing-masing metode. Perbandingan dari hasil klasifikasi *SVM* dan *Naive Bayes* dapat dilihat pada tabel 4. 25 di bawah.

Tabel 4. 25 Perbandingan Metode *SVM* dan *Naive Bayes*

Data latih dan Data Uji 60 : 40		
Ukuran Evaluasi	*SVM*	*Naive Bayes*
Akurasi	81,4%	74,8%
Precision	82%	80%
Recall	81%	75%
F1-Score	80%	70%
Data latih dan Data Uji 70 : 30		
Ukuran Evaluasi	*SVM*	*Naive Bayes*
Akurasi	84,2%	77%
Precision	84%	80%
Recall	84%	77%
F1-Score	84%	74%
Data latih dan Data Uji 80 : 20		
Ukuran Evaluasi	*SVM*	*Naive Bayes*
Akurasi	85,5%	78,1%
Precision	85%	81%
Recall	86%	78%
F1-Score	85%	75%

Berdasarkan tabel 4. 25 di atas dapat dilihat merupakan hasil dari proses *Support Vector Machine* dan *Naive Bayes.* Berdasarkan rasio 60 : 40 *SVM* memiliki akurasi lebih tinggi yaitu 81,4% dibandingkan dengan *Naive Bayes* dengan akurasi 74,8%. Terdapat selisih sebesar 6,6% dari nilai akurasi kedua metode tersebut.

Pada *precsison SVM* masih sedikit lebih tinggi dengan nilai 82% dibandingkan dengan *Naive Bayes* dengan nilai 80%. Faktor-faktor yang dapat mempengaruhi *SVM* lebih tinggi dibandingkan dengan *Naive Bayes* yaitu, *SVM* memilih *hyperlane* yang memaksimalkan margin antara kelas-kelas yang berbeda, yang membantu dalam membuat prediksi positif yang lebih akurat dan mengurangi jumlah *false postives,* sedangkan pada *Naive Bayes* yang menghitung probabilitas untuk setiap kelas dan memilih kelas dengan probabilitas tertinggi dapat meningkatkan jumlah *false positives.*

Pada *recall SVM* memiliki keunggulan yang cukup jauh dengan 81% dibandingkan dengan *Naive Bayes* dengan nilai 75%. Faktor yang dapat mempengaruhi *SVM* lebih tinggi dibandingkan dengan *Naive Bayes* yaitu, *SVM* memaksimalkan margin, yang meningkatkan kemampuannya untuk menangkap *instance* positif dengan lebuh baik, sedangkan pada *Naive Bayes* asumsi indepensi *Naive Bayes* dapat menyebabkan hilangnya *instance* positif karena korelasi antar kata yang tidak diperhitungkan.

Pada perbandingan F1-*Score SVM* lebih tinggi dengan nilai 80% dibandingkan dengan *Naive Bayes* dengan 70%. Faktor yang dapat mempengaruhi nilai tersebut yaitu karena keterbatasan *Naive Bayes* dalam korelasi fitur.

Berdasarkan rasio 70 : 30, menunjukkan bahwa seluruh nilai mengalami peningkatan, hal ini dapat disebabkan karena peningkatan data latih membantu kedua metode dalam mempelajari kata-kata dalam *dataset* dengan lebih baik, sehingga meningkatkan performa dalam memprediksi data uji yang jumlahnya berkurang dibandingkan dengan rasio perbandingan sebelumnya.

Pada nilai akurasi *SVM* masih lebih unggul dengan 84,2% dibandingkan dengan *Naive bayes* yaitu 77% memiliki selisih 7.2% mengalami peningkatan dibandingkan pada perbandingan 60:40, dengan meningkatnya jumlah data latih menunjukkan peningkatan kemampuan *SVM* dalam memisahkan kelas sehingga

SVM semakin mampu mengenali pola yang lebih kompleks, sedangkan pada *Naive Bayes* meskipun mengalami peningkatan nilai akurasi tetapi karena asumsi independensi kata masih menjadi kelemahan dalam memahami korelasi antar kata.

Pada nilai *precision SVM* mengalami peningkatan nilai menjadi 84% dibandingkan dengan *Naive Bayes* dengan nilai 80% yang tidak mengalami peningkatan nilai, hal ini menunjukkan bahwa metode *SVM* semakin efektif dalam menghindari *false positive* sedangkan *Naive Bayes* masih melakukan prediksi *false positive* yang lebih tinggi dibandingkan dengan *SVM.*

Pada nilai *recall* SVM semakin efektif dalam menemukan *instance* positif dengan nilai 84% dibandingkan dengan *Naive Bayes* dengan nilai 77% menunjukkan bahwa *Naive Bayes* masih kalah dalam mendeteksi semua *instance* positif, meskipun mengalami peningkatan, menunjukkan bahwa *Naive Bayes* lebih sering dalam melakukan kesalahan deteksi pada kelas positif.

Pada nilai *F1-Score SVM* masih lebih unggul dengan 84% dibandingkan dengan *Naive Bayes* 74%, menunjukkan *SVM* memiliki keseimbangan yang sangat baik antara *precision* dan *recall* yang berarti bahwa *SVM* bekerja dengan sangat baik secara keseluruhan, sedangkan *Naive Bayes* mengalami peningkatan yang cukup baik pada perbandingan dengan data latih yang lebih besar yaitu dengan mendapatkan data latih sebesar 70%. Perbandingan 70 : 30 menunjukkan bahwa metode *SVM* berhasil mendapatkan nilai yang stabil pada seluruh ukuran evaluasi, dengan keseluruhan mendapatkan nilai 84%.

Berdasarkan rasio 80 : 20 menunjukkan seluruh nilai mengalami kenaikan yang cukup besar, dengan semakin banyaknya data latih membuat metode semakin memiliki kemampuan dalam melakukan analisis. *SVM* mendapatkan akurasi 85,5% menunjukkan kemampuan yang sangat tinggi dalam memprediksi dengan benar, memanfaatkan lebih banyak data latih untuk mengenali pola dengan lebih baik, sedangkan *Naive Bayes* mengalami peningkatan yang cukup besar dengan berhasil mendapatkan nilai 78.1% memiliki selisih sebesar 7.4% yang menunjukkan *SVM* masih jauh lebih unggul.

Pada *precision* SVM mendapatkan nilai 85% pada perbandingan data 80 : 20 menunjukkan bahwa *SVM* sangat efektif dalam menghindari *false positive*

dengan data latih yang lebih besar, *precision* pada *SVM* menjadi evaluasi yang mengalami selisih peningkatan terkecil yaitu hanya mengalami kenaikan 2% dari perbandingan 60 : 40 ke perbandingan 70 : 30 dan hanya mengalami kenaikan 1% pada perbandingan 80 : 20, sedangkan *Naive Bayes* juga mengalami peningkatan nilai menjadi 81% pada rasio perbandingan 80 : 20 tetapi karena masih lemah dalam melakukan prediksi terhadap ulasan positif sehingga nilai *precision* masih lebih rendah dibandingkan dengan *SVM*. Pada *Naive Bayes* tidak terdapat peningkatan nilai *precision* dari perbandingan 60 : 40 ke perbandingan 70 : 30 nilai *precision* tetap berada pada angka 80%, hanya berselisih 1% dengan perbandingan 80 : 20. Nilai *precision* menjadi nilai tertinggi yang berhasil diraih oleh *Naive Bayes. Precision* pada *Naive Bayes* relatif stabil di angka 80-81%, menunjukkan sensitivitas yang lebih rendah terhadap perubahan rasio pembagian data.

Pada nilai *recall SVM* juga masih lebih baik dengan mendapatkan nilai 86% pada rasio 80 : 20 menunjukkan bahwa *SVM* sangat baik dalam mendeteksi semua *instance* positif dibandingkan dengan *Naive Bayes* yang mendapatkan nilai 78% yang menunjukkan bahwa *Naive Bayes* masih kurang efektif dalam menemukan semua *instance* positif dibandingkan dengan *SVM*. Nilai *recall* pada *SVM* dan *Naive Bayes* mengalami peningkatan dengan data latih yang lebih besar, peningkatan yang terjadi memiliki selisih sebesar 1% pada setiap perbandingan rasio yang berbeda.

Meningkatnya nilai *precision* dan *recall* juga mempengaruhi nilai *F1-Score* yang mengalami peningkatan, Pada *SVM* mendapatkan nilai 85% pada rasio perbandingan 80 : 20 menunjukkan keseimbangan yang sangat baik pada *precision* dan *recall,* menunjukkan performa yang sangat baik secara keseluruhan, sedangkan *Naive Bayes* dengan nilai 75% menunjukkan adanya peningkatan pada data latih yang lebih besar. Selisih *F1-Score* dengan perbandingan yang berbeda menunjukkan nilai yang stabil yaitu memiliki selisih 10% antara *SVM* dan *Naive Bayes*. *F1-Score* menjadi ukuran evaluasi yang memili nilai terendah pada metode *Naive Bayes* dengan rata-rata nilai hanya sebesar 73%, sedangkan pada *SVM* memiliki rata-rata nilai sebesar 83%, dengan nilai rata-rata sebesar 83% menunjukkan bahwa *SVM* memiliki ketepatan yang baik dalam melakukan identifikasi kelas positif dengan tingkat *false positive* yang rendah.

Berdasarkan perbandingan tersebut dapat terlihat bahwa nilai *precision* selalu stabil dalam setiap rasio perbandingan, dan pada kedua metode. Kestabilan pada *precision* menunjukkan bahwa *dataset* memiliki distribusi yang relatif seimbang antara kelas positif dan negatif sehingga metode lebih mudah dalam membuat prediksi yang tepat, meskipun memiliki akurasi yang lebih rendah dibandingkan dengan *SVM* tetapi nilai *precision* pada *Naive Bayes* mendapatkan hasil yang cukup tinggi, ini dapat terjadi karena *Naive Bayes* mampu menangkap probabilitas dari fitur individu yang berkorelasi dengan target kelas yaitu positif.

Dibandingkan dengan nilai *precision* yang selalu stabil dengan nilai yang tinggi, nilai *recall* pada *Naive Bayes* menunjukkan nilai yang tidak cukup baik pada seluruh rasio perbandingan, faktor yang dapat menjadi penyebab rendahnya nilai *recall* pada *Naive bayes* yaitu, karena metode ini mengasumsikan bahwa semua kata adalah independen satu sama lain, sehingga mengakibatkan *Naive Bayes* tidak dapat menangkap korelasi yang kompleks antara kata, yang mengarah pada prediksi yang kurang akurat untuk *instance* positif, dan *Naive Bayes* sangat bergantung pada distribusi keseimbangan antara kelas data, jika distribusi tersebut tidak seimbang maka akan mempengaruhi kinerja model, dan menyebabkan nilai *recall* menjadi lebih rendah.

Berdasarkan hasil tersebut dapat diketahui bahwa dengan rasio perbandingan 60 : 40 *SVM* mendapatkan hasil akurasi 74.8% sudah jauh lebih unggul di bandingkan dengan *Naive Bayes* yang pada rasio perbandingan 80 : 20 hanya mendapatkan nilai akurasi 78.1%. Perbedaan akurasi ini dapat disebabkan oleh beberapa faktor yaitu, metode *SVM* memili kemampuan untuk memisahkan kelas dengan margin yang maksimal, yang membantu dalam menghasilkan keputusan yang lebih tepat, sedangkan *Naive Bayes* mengasumsikan bahwa kata-adalah independen. Jika kata-kata dalam *dataset* memiliki korelasi *Naive Bayes* kesulitan untuk mengetahui korelasi antar kata, seperti kemunculan kata “bagus” tidak mempengaruhi kemunculan kata “jelek” atau “tidak” pada kenyataannya, kata-kata ini sering berkorelasi. Ketika kata-kata berkorelasi, asumsi independensi *Naive Bayes* tidak lagi valid, sedangkan, metode *SVM* tidak membuat asumsi tentang independensi kata, sehingga dapat menangani korelasi antara kata dengan

lebih baik. Asumsi bahwa setiap kata adalah independen menyebabkan penurunan akurasi pada *Naive* Bayes.

SVM secara konsisten mengungguli *Naive Bayes* di berbagai pembagian data, menunjukkan kekuatannya dalam menangani data linear dan non-linear dengan kemampuan memodelkan pola dan dependensi yang kompleks. *SVM* sangat efektif dalam analisis sentimen karena kemampuannya untuk memisahkan data yang tidak terpisahkan secara linear dengan menggunakan *kernel trick*. Ini memungkinkan *SVM* untuk memodelkan batas keputusan yang kompleks, yang sangat berguna dalam menangani berbagai ekspresi bahasa dalam teks ulasan.

Naive Bayes, meskipun efisien dan cepat, tetapi terbatas oleh asumsi independensi fiturnya dan kesulitan dalam menangkap interaksi antara kata-kata. Asumsi bahwa setiap fitur (kata) bersifat independen satu sama lain sering kali tidak sesuai dengan kenyataan dalam data teks, di mana kata-kata dapat memiliki korelasi yang signifikan. Hal ini menyebabkan *Naive Bayes* kesulitan dalam mengidentifikasi pola yang lebih kompleks, sehingga kinerjanya relatif lebih rendah dibandingkan dengan *SVM*.

Dengan demikian, *SVM* lebih unggul pada analisis sentimen terhadap ulasan aplikasi Access by KAI di tiga perbandingan rasio yang berbeda, menunjukkan kemampuannya yang lebih baik dalam memodelkan kompleksitas data teks dibandingkan dengan *Naive Bayes*.

5. PENUTUP

5.1. Kesimpulan

Proses analisis sentimen yang dilakukan dengan menggunakan metode *Support Vector Machine* dan *Naive Bayes* dengan membagi *dataset* menjadi tiga perbandingan data latih dan data uji yang berbeda yaitu perbandingan 60 : 40, perbandingan 70 : 30, dan perbandingan 80 : 20. Berdasarkan ketiga perbandingan data latih dan data uji yang berbeda tersebut, menghasilkan empat ukuran evaluasi yaitu akurasi, *precision, recall,* dan *F1-Score.*

Berdasarkan hasil pengujian yang telah dilakukan, hasil evaluasi yang didapatkan oleh metode *SVM* pada perbandingan 60 : 40 mendapatkan hasil 81,4% untuk nilai akurasi, 82% untuk nilai *precision,* 81% untuk nilai *recall,* dan 80% untuk nilai *F-1 Score*, pada rasio perbandingan 70 : 30 mendapatkan hasil 84,2% pada nilai akurasi, 84% pada nilai *precision*, 84% pada nilai *recall,* dan 84% pada nilai *F1-Score* dan pada perbandingan 80 : 20 mendapatkan hasil 85,5% pada nilai akurasi, 85% pada nilai *precision* dan *F1-Score,* dan 86% pada nilai *recall*. Pada pengujian yang dilakukan dengan menggunakan metode *Naive Bayes* hasil evaluasi yang didapatkan pada perbandingan 60 : 40 mendapatkan hasil 74,8% pada nilai akurasi, 80% pada nilai *precision,* 75% pada nilai *recall,* dan 70% pada nilai *F1-Score*, pada rasio perbandingan 70 : 30 mendapatkan hasil 77% pada nilai akurasi, 80% pada nilai *precision,* 77% pada nilai *recall,* dan 74% pada nilai *F1-Score*, dan pada perbandingan 80 : 20 mendapatkan hasil 78,1% pada nilai akurasi, 81% pada nilai *precision,* 78% pada nilai *recall,* dan 75% pada nilai *F1-Score*.

Berdasarkan hasil evaluasi yang telah didapatkan menunjukkan bahwa *SVM* secara konsisten lebih unggul dibandingkan dengan *Naive Bayes* pada seluruh nilai evaluasi di seluruh perbandingan data yang berbeda. Bahkan dengan jumlah data latih yang lebih sedikit yaitu pada perbandingan 60 : 40 dapat lebih unggul dibandingkan dengan *Naive Bayes* pada perbandingan 80 : 20 serta *SVM* secara konsisten mengalami peningkatan nilai evaluasi pada perbandingan dengan data latih yang lebih besar. Berdasarkan hasil tersebut maka didapatkan kesimpulan

bahwa metode *Support Vector Machine* memiliki nilai evaluasi yang lebih unggul pada klasifikasi data ulasan aplikasi Access by KAI dibandingkan dengan metode *Naive Bayes.*

5.2. Saran

Dalam penelitian yang telah dilakukan, ulasan pengguna yang diberikan terhadap aplikasi Access by KAI dibagi menjadi dua kelas yaitu ulasan positif dan ulasan negatif, dapat dikembangkan pada penelitian selanjutnya menjadi kelas lain seperti ulasan positif kuat, dan ulasan negatif kuat. Perbaikan juga dapat dilakukan dengan menambahkan jumlah data ulasan dengan mengambil ulasan yang diberikan setelah tanggal 13 Maret 2024 sehingga jumlah ulasan yang dapat dianalisis menjadi lebih banyak dan diharapkan hasil sentimen yang didapatkan menjadi lebih baik. Menambahkan fitur untuk memahami penggunaan *emoticon* yang terdapat di ulasan untuk meningkatkan pemahaman terhadap sentimen ulasan.

DAFTAR PUSTAKA

Agung Pramana, T., Ramdhani, Y. (2023). Sentiment Analysis Tanggapan Masyarakat Tentang Hacker Bjorka Menggunakan Metode SVM. *Jurnal Nasional Komputasi Dan Teknologi Informasi*, *6*(1), 49–62. https://doi.org/https://ojs.serambimekkah.ac.id/jnkti/article/view/5583

Alfyando, M. (2024). Perbandingan Algoritma Random Forest dan Logistic Regression Untuk Analisis Sentimen Ulasan Aplikasi Tumbuh Kembang Anak Di Play Store. *Jurnal Sistem Informasi Dan Ilmu Komputer*, *2*(1), 77–86. https://doi.org/10.59581/jusiik-widyakarya.v2i1.2262

Fikri, M. I., Sabrila, T. S., Azhar, Y. (2020). Perbandingan Metode Naïve Bayes dan Support Vector Machine pada Analisis Sentimen Twitter. *SMATIKA Jurnal*, *10*(2),71–76. https://doi.org/https://journal.stekom.ac.id/index.php/elkom/article/view/918

Gelar Guntara, R. (2023). Pemanfaatan Google Colab Untuk Aplikasi Pendeteksian Masker Wajah Menggunakan Algoritma Deep Learning YOLOv7. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *5*(1), 55–60. https://doi.org/10.47233/jteksis.v5i1.750

Harani, N., Nugraha, F. (2020). *Segmentasi Pelanggan Menggunakan Python* (F. Nugraha, Ed.; 1st ed., Vol. 1). Kreatif Industri Nusantara.

Hibban, M. I., Susila, A. (2024). Analisis Sentimen Terhadap PSBB Menggunakan Algoritma Naïve Bayes. *Jurnal Ilmu Komputer Dan Pendidikan*,*2*(2),376–379. https://doi.org/https://journal.mediapublikasi.id/index.php/logic/article/view/2772

Ilmawan, L. B., Mude, M. A. (2020a). Perbandingan Metode Klasifikasi Support Vector Machine dan Naïve Bayes untuk Analisis Sentimen pada Ulasan Tekstual di Google Play Store. *ILKOM Jurnal Ilmiah*, *12*(2), 154–161. https://doi.org/10.33096/ilkom.v12i2.597.154-161

Ilmawan, L. B., Mude, M. A. (2020b). Perbandingan Metode Klasifikasi Support Vector Machine dan Naïve Bayes untuk Analisis Sentimen pada Ulasan Tekstual di Google Play Store. *ILKOM Jurnal Ilmiah*, *12*(2), 154–161. https://doi.org/10.33096/ilkom.v12i2.597.154-161

Indrayuni, E. (2018). Komparasi Algoritma Naive Bayes Dan Support Vector Machine Untuk Analisa Sentimen Review Film. *Jurnal PILAR Nusa Mandiri*, *14*(2),175–179.

https://doi.org/https://ejournal.nusamandiri.ac.id/index.php/pilar/article/view/36

Kusnia, U., Kurniawan, F., Artikel, S. (2022). Analisis Sentimen Review Aplikasi Media Berita Online Pada Google Play menggunakan Metode Algoritma Support Vector Machines (SVM) Dan Naive Bayes. *Jurnal Keilmuan Dan Aplikasi Teknik Informatika*, 24–28. https://doi.org/10.35891/explorit

Maulana, B. A., Fahmi, M. J., Imran, A. M., Hidayati, N. (2024). Analisis Sentimen Terhadap Aplikasi Pluang Menggunakan Algoritma Naive Bayes dan Support Vector Machine (SVM). *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, *4*(2), 375–384. https://doi.org/10.57152/malcom.v4i2.1206

Md, R., Restiyan, R. D., Irsyad, H. (2024). Analisis Sentimen Masyarakat terhadap Perilaku Lawan Arah yang diunggah pada Media Sosial Youtube Menggunakan Naïve Bayes. In *BANDWIDTH: Journal of Informatics and Computer Engineering* (Vol. 02, Issue 02).

Miftahusalam, A., Pratiwi, H., Slamet, I. (2023). Perbandingan Metode Random Forest dan Naive Bayes pada Analisis Sentimen Review Aplikasi BCA Mobile. *SIPTEK : Seminar Nasional Inovasi Dan Pengembangan Teknologi Pendidikan*,*1*(1),1–8. https://doi.org/https://proceeding.unesa.ac.id/index.php/siptek/article/view/184

Muttaqin, M. N., Kharisudin, I. (2021). Analisis Sentimen Pada Ulasan Aplikasi Gojek Menggunakan Metode Support Vector Machine dan K Nearest Neighbor. *UNNES Journal of Mathematics*, *10*(2), 22–27. https://doi.org/http://journal.unnes.ac.id/sju/index.php/ujm

Nishfi, D., Huda, I., Prianto, C., Awangga, R. M. (2023). Dellavianti Nishfi Ilmiah Huda Analisis Sentimen Perbandingan Layanan Jasa Pengiriman Kurir Pada Ulasan Play Store Menggunakan Metode Random Forest dan Descision Tree. *jurnal ilmiah informatika*, *11*(2), 150–158. https://doi.org/https://ejournal.upbatam.ac.id/index.php/jif/article/view/7952

Nurtikasari, Y., Syariful Alam., Teguh Iman Hermanto. (2022). Analisis Sentimen Opini Masyarakat Terhadap Film Pada Platform Twitter Menggunakan Algoritma Naive Bayes. *INSOLOGI: Jurnal Sains Dan Teknologi*, *1*(4), 411–423. https://doi.org/10.55123/insologi.v1i4.770

Oktafiandi, H., Raziq Olajuwon, S. M. (2023). Perbandingan Algoritma untuk Analisis Sentimen Terhadap Google Play Store Menggunakan Machine Learning. *jurnal ekonomi dan teknik informatika*, *11*(2), 16–21. https://doi.org/https://e-journal.polsa.ac.id/index.php/jneti/article/view/234

Pamungkas, F. S., Kharisudin, I. (2021). Analisis Sentimen dengan SVM, NAIVE BAYES dan KNN untuk Studi Tanggapan Masyarakat Indonesia Terhadap Pandemi Covid-19 pada Media Sosial Twitter. *Journal UNNES*, *4*(1), 628–634. https://doi.org/https://journal.unnes.ac.id/sju/index.php/prisma/article/view/450

Purnajaya, A. R., Lieputra, V., Tayanto, V., Salim, J. G. (2022). Implementasi Text Mining untuk Mengetahui Opini Masyarakat Tentang Climate Change. *Journal of Information System and Technology*, *03*(03), 320–328. https://doi.org/https://journal.uib.ac.id/index.php/joint/article/view/7337

Putri Gabriella, Y. A. (2023). Optimasi Penerimaan Siswa Baru Dengan Penerapan Algortima Text Mining Dan Tf-Idf. *Journal of Computing and Informatics Research*, *2*(3), 110–117. https://doi.org/10.47065/comforch.v2i3.941

Putri, M. I., Kharisudin, I. (2022). Penerapan Synthetic Minority Oversampling Technique (SMOTE) Terhadap Analisis Sentimen Data Review Pengguna Aplikasi Marketplace Tokopedia. *PRISMA, Prosiding Seminar Nasional Matematika*, *5*, 759–766. https://journal.unnes.ac.id/sju/index.php/prisma/

Rahayu, A. S., Fauzi, A., Rahmat, R. (2022). Komparasi Algoritma Naïve Bayes Dan Support Vector Machine (SVM) Pada Analisis Sentimen Spotify. *Jurnal Sistem Komputer Dan Informatika (JSON)*, *4*(2), 349. https://doi.org/10.30865/json.v4i2.5398

Resa, M., Yudianto, A., Rahim, A., Sukmasetya, P., Hasani, R. A. (2022). Perbandingan Metode Support Vector Machine Dengan Metode Lexicon Dalam Analisis Sentimen Bahasa Indonesia. *Jurnal Teknologi Informasi)*, *6*(1), 7–13. https://doi.org/https://jurnal.una.ac.id/index.php/jurti/article/view/2537

Ridwansyah, T. (2022). KLIK: Kajian Ilmiah Informatika dan Komputer Implementasi Text Mining Terhadap Analisis Sentimen Masyarakat Dunia Di Twitter Terhadap Kota Medan Menggunakan K-Fold Cross Validation Dan Naïve Bayes Classifier. *Media Online*, *2*(5), 178–185. https://doi.org/https://djournals.com/klik

Rokhman, K., Arsi, P., Berlilana. (2021). Perbandingan Metode Support Vector Machine dan Decision Tree Untuk Analisis Sentimen Review Komentar Pada aplikasi Transportasi Online. *jurnal of information system management*, *2*(2), 1–7. https://doi.org/https://jurnal.amikom.ac.id/index.php/joism/article/view/341

Saron Tandiapa, S., Caren Rorimpandey, G. (2023). Analisis Sentimen Ulasan Pengguna Pada Aplikasi Threads Dengan Metode Lexicon Based Dan Naive Bayes Classifier. *Jurnal Cahaya Mandalika*, *3*(1), 339–344.

https://doi.org/https://ojs.cahayamandalika.com/index.php/jcm/article/view/2585

Sayuti Rahman, Arnes Sembiring, Dodi Siregar, Husnul Khair, I Gusti Prahmana, Ratih Puspadini., Muhammad Zen. (2023). *Dasar dan Pemrograman Berorientasi Objek* (tahta Media, Ed.; 1st ed.). Tahta Media.

Septiani, D., Isabela, I. (2022). Analisis Term Frequency Inverse Document Frequency (TF-IDF) Dalam Temu Kembali Informasi Pada Dokumen Teks. *Jurnal Sistem Dan Teknologi Informasi Indonesia* , *1*(2), 81–88. https://doi.org/https://journal.unj.ac.id/unj/index.php/SINTESIA/article/view/3936

Simbolon, A. S., Pangaribuan, N. I., Aruan, N. M. (2021). Analisis Sentimen Aplikasi E-Learning Selama Pandemi Covid-19 Dengan Menggunakan Metode Support Vector Machine Dan Convolutional Neural Network. *SEMINASTIKA*, *3*(1), 16–25. https://doi.org/10.47002/seminastika.v3i1.236

LAMPIRAN LISTING PROGRAM

Akuisisi Data:

```
!pip install google-play-scraper
from google_play_scraper import app
import pandas as pd
import numpy as np
from google_play_scraper import Sort, reviews
result, continuation_token = reviews(
    'com.kai.kaiticketing',
    lang='id',
    country='id',
    sort=Sort.MOST_RELEVANT,
)
reviews_df = pd.DataFrame(result)
reviews_df['at'] = pd.to_datetime(reviews_df['at'])
start_date = '2023-03-13'
end_date = '2023-08-10'
filtered_reviews_df = reviews_df[(reviews_df['at'] >= start_date) & (reviews_df['at'] <= end_date)]
df_busu = pd.DataFrame(np.array(result),columns=['review'])
df_busu = df_busu.join(pd.DataFrame(df_busu.pop('review').tolist()))
df_busu.head()
len(df_busu.index)
df_busu[['at', 'content']].head()
new_df = df_busu[['at', 'content']]
sorted_df = new_df.sort_values(by='at', ascending=False).
sorted_df.head()
my_df = sorted_df[['at', 'content']]
my_df.head()
my_df.to_csv("scrapped_data.csv", index = False)
```

Preprocessing :

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
import numpy as np
df = pd.read_csv("/content/drive/MyDrive/Acces by KAI 3587.csv", encoding='ISO-8859-1')\
```

tokenizing:

```
df['content_tokens'] = df['content'].apply(word_tokenize_wrapper)
print('Preprocessing Result : \n')
print(df['content_tokens'].head())
```

Case folding:

```
import pandas as pd
df = pd.read_csv("/content/drive/MyDrive/Acces by KAI 3587.csv", encoding='ISO-8859-1')
df['content'] = df['content'].str.lower()
print('Case Folding Result : \n')
print(df['content'].head(5))
```

cleaning:

```
import string
import re
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import nltk
nltk.download('punkt')
import pandas as pd
import numpy as np
import re
import string
from nltk.tokenize import word_tokenize
def replace_comma_with_space(text):
    return text.replace(",", " ")
def remove_tweet_special(text):
    text = text.replace('\\t', " ").replace('\\n', " ").replace('\\', "")
    text = text.encode('ascii', 'replace').decode('ascii')
    text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)", " ", text).split())
    return text.replace("http://", " ").replace("https://", " ")
```

```
def remove_number(text):
    return re.sub(r"\d+", "", text)
def remove_punctuation(text):
    return text.translate(str.maketrans("", "", string.punctuation))
def remove_whitespace_LT(text):
    return text.strip()
def remove_whitespace_multiple(text):
    return re.sub('\s+', ' ', text)
def remove_single_char(text):
    return re.sub(r"\b[a-zA-Z]\b", "", text)
def remove_repeated_letters(text):
    return re.sub(r'(.)\1+', r'\1', text)
def case_folding(text):
    return text.lower()
def word_tokenize_wrapper(text):
    return word_tokenize(text)
def preprocess(text):
    text = replace_comma_with_space(text)
    text = remove_tweet_special(text)
    text = remove_number(text)
    text = remove_punctuation(text)
    text = remove_whitespace_LT(text)
    text = remove_whitespace_multiple(text)
    text = remove_single_char(text)
    text = remove_repeated_letters(text)
    text = case_folding(text)
    return text
df = pd.read_csv("/content/drive/MyDrive/Acces by KAI 3587.csv", encoding='ISO-8859-1')
df['content'] = df['content'].apply(preprocess)
import re
def remove_repeated_letters(text):
    return re.sub(r'(.)\1+', r'\1', text)
df['content'] = df['content'].apply(remove_repeated_letters)
print(df['content'].head())
Stopwords removal:
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
list_stopwords = stopwords.words('indonesian')
list_stopwords.extend(["yg", "dg", "rt", "dgn", "ny", "d", 'klo',
                       'kalo', 'amp', 'biar', 'bikin', 'bilang',
                       'gak', 'ga', 'krn', 'nya', 'nih', 'sih',
                       'si', 'tau', 'tdk', 'tuh', 'utk', 'ya',
                       'jd', 'jgn', 'sdh', 'aja', 'n', 't',
                       'nyg', 'hehe', 'pen', 'u', 'nan', 'loh', 'rt',
                       '&amp', 'yah', 'nya', 'untuk', 'di', 'pada', 'ini'])
txt_stopword = pd.read_csv("stopwords.txt", names= ["stopwords"], header = None)
list_stopwords.extend(txt_stopword["stopwords"][0].split(' '))
list_stopwords = set(list_stopwords)
def stopwords_removal(words):
    return [word for word in words if word not in list_stopwords]
df['content_tokens_WSW'] = df['content_tokens'].apply(stopwords_removal)
print(df['content_tokens_WSW'].head())
Normalizad:
normalizad_word = pd.read_csv("colloquial-indonesian-lexicon.csv")
normalizad_word_dict = {}
for index, row in normalizad_word.iterrows():
    if row[0] not in normalizad_word_dict:
        normalizad_word_dict[row[0]] = row[1]
def normalized_term(document):
    return [normalizad_word_dict[term] if term in normalizad_word_dict else term for term in document]
df['content_normalized'] = df['content_tokens_WSW'].apply(normalized_term)
df['content_normalized'].head(10)
Stemmer:
pip install swifter
pip install sastrawi
import pandas as pd
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import swifter
```

```
import ast
def convert_tokenizing_to_list(text):
    try:
        # Menggunakan ast.literal_eval untuk mengonversi string menjadi list
        text_list = ast.literal_eval(text)
        return text_list
    except Exception as e:
        print("Error:", e)
        return []
def stemmed_wrapper(term):
    return stemmer.stem(term)
df = pd.read_csv("Text_normalized.csv")
factory = StemmerFactory()
stemmer = factory.create_stemmer()
df['content_normalized_list'] = df['content_normalized'].apply(convert_tokenizing_to_list)
term_dict = {}
for document in df['content_normalized_list']:
    for term in document:
        if term not in term_dict:
            term_dict[term] = ' '
print(len(term_dict))
print("------------------------")
    term_dict[term] = stemmed_wrapper(term)
    print(term, ":", term_dict[term])
print("------------------------")
def get_stemmed_term(document):
    return [term_dict[term] for term in document]
df['content_tokens_stemmed'] = df['content_normalized_list'].swifter.apply(get_stemmed_term)
print(df['content_tokens_stemmed'])
TF-IDF:
import pandas as pd
import numpy as np
df = pd.read_csv("Text_Preprocessing.csv", usecols=["tweet_tokens_stemmed"])
df.columns = ["content"]
df.head()
import ast
def convert_text_list(texts):
    texts = ast.literal_eval(texts)
    return [text for text in texts]
df["content_list"] = df["content"].apply(convert_text_list)
print(df["content_list"][90])
print("\ntype : ", type(df["content_list"][90]))
def calc_TF(document):
    # Counts the number of times the word appears in review
    TF_dict = {}
    for term in document:
        if term in TF_dict:
            TF_dict[term] += 1
        else:
            TF_dict[term] = 1
    for term in TF_dict:
        TF_dict[term] = TF_dict[term] / len(document)
    return TF_dict
df["TF_dict"] = df['content_list'].apply(calc_TF)
df["TF_dict"].head()
index = 90
print('%20s' % "term", "\t", "TF\n")
for key in df["TF_dict"][index]:
    print('%20s' % key, "\t", df["TF_dict"][index][key])
def calc_DF(tfDict):
    count_DF = {}
    for document in tfDict:
        for term in document:
            if term in count_DF:
                count_DF[term] += 1
            else:
                count_DF[term] = 1
    return count_DF
DF = calc_DF(df["TF_dict"])
```

```
n_document = len(df)
def calc_IDF(__n_document, __DF):
    IDF_Dict = {}
    for term in __DF:
        IDF_Dict[term] = np.log(__n_document / (__DF[term] + 1))
    return IDF_Dict
DF = calc_IDF(n_document, DF)
def calc_TF_IDF(TF):
    TF_IDF_Dict = {}
        TF_IDF_Dict[key] = TF[key] * IDF[key]
    return TF_IDF_Dict
df["TF-IDF_dict"] = df["TF_dict"].apply(calc_TF_IDF)
index = 90
print('%20s' % "term", "\t", '%10s' % "TF", "\t", '%20s' % "TF-IDF\n")
for key in df["TF-IDF_dict"][index]:
    print('%20s' % key, "\t", df["TF_dict"][index][key] ,"\t" , df["TF-IDF_dict"][index][key])
def calc_TF_IDF_Vec(__TF_IDF_Dict):
    TF_IDF_vector = [0.0] * len(unique_term)
        if term in __TF_IDF_Dict:
            TF_IDF_vector[i] = __TF_IDF_Dict[term]
    return TF_IDF_vector
df["TF_IDF_Vec"] = df["TF-IDF_dict"].apply(calc_TF_IDF_Vec)
print("print first row matrix TF_IDF_Vec Series\n")
print(df["TF_IDF_Vec"][0])
print("\nmatrix size : ", len(df["TF_IDF_Vec"][0]))
TF_IDF_Vec_List = np.array(df["TF_IDF_Vec"].to_list())
sums = TF_IDF_Vec_List.sum(axis=0)
data = []
for col, term in enumerate(unique_term):
    data.append((term, sums[col]))
ranking = pd.DataFrame(data, columns=['term', 'rank'])
ranking.sort_values('rank', ascending=False)
Labeling:
import pandas as pd
import ast
df = pd.read_csv("Text_Preprocessing.csv", usecols=["tweet_tokens_stemmed"])
df["tweet_tokens_stemmed"] = df["tweet_tokens_stemmed"].apply(lambda x: ast.literal_eval(x))
print("Tipe data setelah diubah:", type(df["tweet_tokens_stemmed"][0]))
import pandas as pd
import numpy as np
lexicon = pd.read_csv("modified_full_lexicon.csv", encoding='ISO-8859-1')
lexicon = lexicon.drop(lexicon[lexicon['word'].isin(["bukan", "tidak", "ga", "gk"])].index)
lexicon_word = lexicon['word'].tolist()
lexicon_weights = lexicon['weight']
def calculate_sentiment(text):
    sentiment = 0
    words = text.split()
    for word in words:
        if word in lexicon_word:
            sentiment += lexicon_weights[lexicon_word.index(word)]
    return sentiment
df = pd.read_csv("Text_Preprocessing_modified.csv")
df['sentiment'] = df['tweet_tokens_stemmed'].apply(calculate_sentiment)
df['label'] = np.where(df['sentiment'] >= 0, 1, 0)
df.to_csv("labeled_Acces_by_KAI_3587.csv", index=False)
Splitting Data:
import pandas as pd
df = pd.read_csv('dataset_berlabel.csv')
train_percentages = [0.6, 0.7, 0.8]  # Contoh nilai persentase untuk train_size
accuracy_result = []
from sklearn.model_selection import train_test_split
for percentage in train_percentages:
    train_size = int(len(df) * percentage)
    Train_X, Test_X, Train_Y, Test_Y = train_test_split(df['content'], df['label'], train_size=train_size, random_state=8)
from sklearn.feature_extraction.text import TfidfVectorizer
Tfidf_vect = TfidfVectorizer(max_features=5000)
Train_X_Tfidf = Tfidf_vect.fit_transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)
print("Train_X_Tfidf:")
```

```
print(Train_X_Tfidf)
print("\nTest_X_Tfidf:")
print(Test_X_Tfidf)
import pandas as pd
from sklearn.model_selection import train_test_split
train_percentages = [0.6, 0.7, 0.8]  # Contoh nilai persentase untuk train_size
dataframes = []
for percentage in train_percentages:
    train_size = int(len(df) * percentage)
    Train_X, Test_X, Train_Y, Test_Y = train_test_split(df['content'], df['label'], train_size=train_size, random_state=8)
    train_df = pd.DataFrame({'Train_X': Train_X, 'Train_Y': Train_Y})
    test_df = pd.DataFrame({'Test_X': Test_X, 'Test_Y': Test_Y})
    dataframes.append((train_df, test_df))
for idx, (train_df, test_df) in enumerate(dataframes):
    print(f"Train Test Split {idx+1}")
    print("Train Set:")
    print(train_df.head())
    print("\nTest Set:")
    print(test_df.head())
    print("\n")
```

Support Vector Machine:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
train_df = pd.read_csv('train_data_1.csv')
test_df = pd.read_csv('test_data_1.csv')
X_train = train_df['content']
y_train = train_df['label']
X_test = test_df['content']
y_test = test_df['label']
train_df = train_df.dropna(subset=['label'])
test_df = test_df.dropna(subset=['label'])
X_train = train_df['content']
y_train = train_df['label']
X_test = test_df['content']
y_test = test_df['label']
classifier = SVC(kernel='rbf', C=100, gamma=1)
count_vectorizer = CountVectorizer(binary=True)
train_counts = count_vectorizer.fit_transform(X_train)
tfidf_transformer = TfidfTransformer(use_idf=True)
tfidf_train = tfidf_transformer.fit_transform(train_counts)
classifier.fit(tfidf_train, y_train)
test_counts = count_vectorizer.transform(X_test)
tfidf_test = tfidf_transformer.transform(test_counts)
predictions = classifier.predict(tfidf_test)
print("Accuracy: ", accuracy_score(y_test, predictions))
print("Confusion Matrix: \n", confusion_matrix(y_test, predictions))
print("Classification Report: \n", classification_report(y_test, predictions))
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:")
print(conf_matrix)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negatif', 'Positif'], yticklabels=['Negatif',
'Positif'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

Naive Bayes:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
train_data = pd.read_csv('train_data_1.csv')
test_data = pd.read_csv('test_data_1.csv')
Train_X = train_data['content']  # Kolom teks dari data latih
Train_Y = train_data['label']  # Kolom label dari data latih
Test_X = test_data['content']  # Kolom teks dari data uji
Test_Y = test_data['label']  # Kolom label dari data uji
tfidf_vectorizer = TfidfVectorizer(max_features=500)  # Anda dapat mengubah jumlah fitur maksimal sesuai kebutuhan
Train_X_Tfidf = tfidf_vectorizer.fit_transform(Train_X)
Test_X_Tfidf = tfidf_vectorizer.transform(Test_X)
model_nb = MultinomialNB()
print(f"\nTraining Naive Bayes (Train Size: {len(Train_X)} samples):")
model_nb.fit(Train_X_Tfidf, Train_Y)
predictions_nb = model_nb.predict(Test_X_Tfidf)
accuracy_nb = accuracy_score(Test_Y, predictions_nb)
conf_mat_nb = confusion_matrix(Test_Y, predictions_nb)
classification_rep = classification_report(Test_Y, predictions_nb)
print(f"Naive Bayes Accuracy: {accuracy_nb}")
print("Naive Bayes Confusion Matrix:")
print(conf_mat_nb)
print("\nClassification Report:")
print(classification_rep)
sns.heatmap(conf_mat_nb, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```